R Syntax to Accompany Cohen, Cohen, Aiken & West's (2003) *Applied multiple regression/correlation analysis for the behavioral sciences (3rd ed.)*

A. Alexander Beaujean
Baylor University
Alex_Beaujean@Baylor.edu

Xiao Qiu
Baylor University

BPL–2012-CCAW-1
Last updated: March 13, 2017

Contents

List of Figures

List of Tables

Preface

This book is designed to demonstrate how to conduct the analyses in Cohen, Cohen, West, and Aiken (2003) using `R` (R Development Core Team, 2015). It was written in LaTeX, using the **`knitr()`** package. For information on **R** syntax for the chapters not shown, contact Alex Beaujean (Alex_Beaujean@baylor.edu)

Chapter 1

Introduction

There are no data to analyze in this chapter

Chapter 2

Bivariate Correlation and Regression

2.1 Tabular and Graphic Representation of Variables

The data in CCAW Table 2.1.1 are not given in the accompanying data CD, so have to be entered "by hand" into R.

```
#Table 2.1.1 data
Vocabulary<-c(5,8,7,9,10,8,6,6,10,9,7,7,9,6,8)
DigitSymbol<-c(12,15,14,18,19,18,14,17,20,17,15,16,16,13,16)
VocabDS.data<-data.frame(cbind(Vocabulary,DigitSymbol))
```

To create a scatterplot from this data, use the syntax below. The resulting figure is shown in Figure 2.1.

```
#Figure 2.1.1
plot(VocabDS.data$Vocabulary, VocabDS.data$DigitSymbol, xlab="Vocabulary", ylab="Digit-symbol", pch=16)
```

Import the income and major household appliance data.

```
#Income and appliance data (Table 2.2.1)
Income.data<-read.table("C0201DT.txt", header=TRUE, sep="\t")
```

The `header=TRUE` argument is used because the file has variable names at the top. The `sep="\t"` argument is used because the variables are separated by tabs.

To get the rank order of a variable, use the `rank()` function.

```
rank(Income.data$INCOME) #rank order

## [1] 1 3 2 4
```

To get put a variable in the Z-score metric, use the `scale()` function.

```
Income.data$Income.Z<-scale(Income.data$INCOME)[,1] #Z-scores
Income.data$Applianc.Z<-scale(Income.data$APPLIANC)[,1] #Z-scores
```

The `[,1]` is appended to the function because only the first column of the output has the transformed variable values.

2.2 Correlations

The `cor()` function will calculate correlations for a two or more variables.

To import the PhD-publications data, use the following syntax.

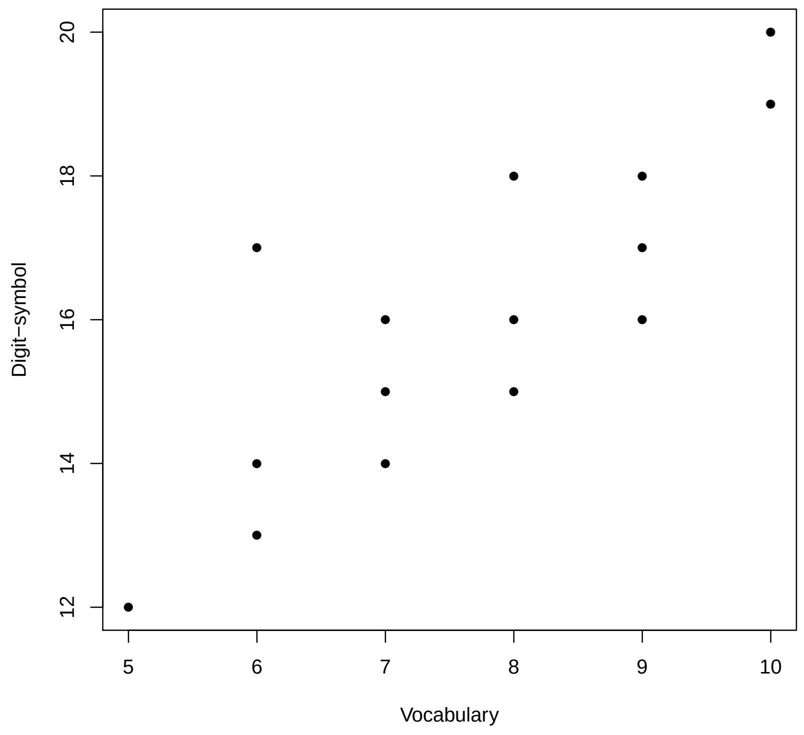


Figure 2.1: Scatterplot

```
#PhD/Publication data (Table 2.2.2)
PhDPub.data<-read.table("C0202DT.txt", header=TRUE, sep="\t")
```

This is how to calculate the Pearson correlation between time since PhD and number of publications.

```
##Correlation
cor(PhDPub.data)

##       TIME  PUBS
## TIME 1.000 0.657
## PUBS 0.657 1.000
```

To import the stimulus-task score data, use the following syntax.

```
# Stimulus data
Stimulus.data<-read.table("C0203DT.txt", header=TRUE, sep="\t")
```

The *stimulus condition* variable is coded using text, so it needs to be re-coded. This can be done in multiple ways. First, you can make a new variable that recodes the the variable numerically. Second, you can use the `as.numeric()` function within the `cor()` function.

```
# Method 1: Recode qualitative variable to 0-1
Stimulus.data$STIMULUS.ReCoded<-ifelse(Stimulus.data$STIMULUS=="None", 0, 1)
##Point-biserial correlation
cor(Stimulus.data$STIMULUS.ReCoded, Stimulus.data$TASK)

## [1] -0.707

# Method two: Use as.numeric() function
cor(as.numeric(Stimulus.data$STIMULUS), Stimulus.data$TASK)

## [1] -0.707
```

To calculate the biserial correlation between a continuous variable and a categorical variable that has an underlying continuous distribution.

```
library(psych)
#biserial correlation
biserial(Stimulus.data$TASK, Stimulus.data$STIMULUS.ReCoded)
```

```
##        [,1]
## [1,] -0.825
```

Import the homeowner data and calculate the ϕ coefficient.

```
# Stimulus data
Homeowner.data<-read.table("C0204DT.txt", header=TRUE, sep="\t")
```

```
##Phi Coefficient
cor(Homeowner.data)

##          HOMEOWN CANDIDAT
## HOMEOWN    1.000   -0.272
## CANDIDAT  -0.272    1.000
```

A better correlation to use when the the variables are dichotomous, but represent underlying continuous variables is the tetrachoric correlation (Harris, 1988).

```
library(psych)
#tetrachoric correlation
tetrachoric(Homeowner.data)
```

```
## Call: tetrachoric(x = Homeowner.data)
## tetrachoric correlation
##          HOMEO CANDI
## HOMEOWN   1.00
## CANDIDAT -0.43  1.00
##
##  with tau of
##  HOMEOWN CANDIDAT
##    -0.27    -0.18
```

The rank order data is not in the data CD, so have to be entered “by hand.”

```
#Table 2.3.3 data
X<-c(4,2,3,5,1)
Y<-c(2,1,4,3,5)
Rank.data<-data.frame(X,Y)
```

To calculate the Spearman rank-order correlation, use the `cor()` function with the `type="spearman"` argument.

```
cor(Rank.data, method = "spearman")

##      X    Y
## X  1.0 -0.3
## Y -0.3  1.0
```

2.2.1 Correcting Correlation Coefficients

To correct a correlation for unreliability, there are two methods. The first is to use a path model that accounts for the measured variables unreliability/error. More will discussed about this after introducing path models. The second is the correction for attenuation (Spearman, 1904). This is done via the `correct.cor()` function in the `psych` package.

```
##Correcting correlation for unreliability
library(psych)
cor<-matrix(c(1,.44,.44,1), ncol=2, nrow=2)
rel<-c(.8, .8)
correct.cor(cor, rel)

##      [,1] [,2]
## [1,] 0.80 0.55
## [2,] 0.44 0.80
```

The coefficients in the upper triangle are the corrected correlations.
To correct for range restriction, use the `rangeCorrection()` function in the psych `package`.

```
#Correcting for range restriction
library(psych)
rangeCorrection(.25, 12,5)

## [1] 0.527
```

2.3 Regression Analysis

In `R` the `lm()` handles most "typical" regression analyses.

```
##Unstandardized regression
PhD.fit<-lm(PUBS~TIME, data=PhDPub.data)
summary(PhD.fit)

##
## Call:
## lm(formula = PUBS ~ TIME, data = PhDPub.data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -13.628  -8.645   0.303   5.846  23.440
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    4.731      5.591    0.85   0.4128
## TIME           1.983      0.632    3.14   0.0078 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.8 on 13 degrees of freedom
## Multiple R-squared:  0.431,Adjusted R-squared:  0.387
## F-statistic: 9.85 on 1 and 13 DF,  p-value: 0.00783
```

There is no function to obtain standardized regression coefficients. To obtain them, you need to use standardized variables in the `lm()` function.

```
##Standardized
PhD.std.fit<-lm(scale(PUBS)~scale(TIME), data=PhDPub.data)
summary(PhD.std.fit)

##
## Call:
## lm(formula = scale(PUBS) ~ scale(TIME), data = PhDPub.data)
##
```

```
## Residuals:
##    Min     1Q Median     3Q    Max
## -0.986 -0.625  0.022  0.423  1.696
##
## Coefficients:
##                           Estimate            Std. Error t value Pr(>|t|)
## (Intercept) 0.0000000000000000323 0.2020823192367780796    0.00   1.0000
## scale(TIME) 0.6566545933266347834 0.2091750729859912228    3.14   0.0078 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.783 on 13 degrees of freedom
## Multiple R-squared:  0.431,Adjusted R-squared:  0.387
## F-statistic: 9.85 on 1 and 13 DF,  p-value: 0.00783
```

Alternatively, you could create a standardized data set.

```
library(plyr)
PhDPub.dataStd <- numcolwise(scale)(PhDPub.data)
PhD.std.fit.alt<-lm(PUBS~TIME, data=PhDPub.dataStd)
summary(PhD.std.fit.alt)

##
## Call:
## lm(formula = PUBS ~ TIME, data = PhDPub.dataStd)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -0.986 -0.625  0.022  0.423  1.696
##
## Coefficients:
##                           Estimate            Std. Error t value Pr(>|t|)
## (Intercept) 0.0000000000000000323 0.2020823192367780796    0.00   1.0000
## TIME        0.6566545933266347834 0.2091750729859912228    3.14   0.0078 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.783 on 13 degrees of freedom
## Multiple R-squared:  0.431,Adjusted R-squared:  0.387
## F-statistic: 9.85 on 1 and 13 DF,  p-value: 0.00783
```

To add a regression line to a scatterplot, use the `abline()` function after creating the scatterplot. The scatterplot with regression line is shown in Figure 2.2.

```
plot(PhDPub.data, pch=16)
# regression line (y~x)
abline(PhD.fit, col="red")
```

2.4 Confidence Interval

To obtain a confidence interval for a regression parameter, use the `confint()` function.

```
##confidence Intervals
confint(PhD.fit, parm = "TIME", level=.95) #Regression slope

##      2.5 % 97.5 %
## TIME 0.618   3.35

confint(PhD.fit, parm = "(Intercept)", level=.95) #Regression Intercept

##             2.5 % 97.5 %
## (Intercept) -7.35   16.8
```

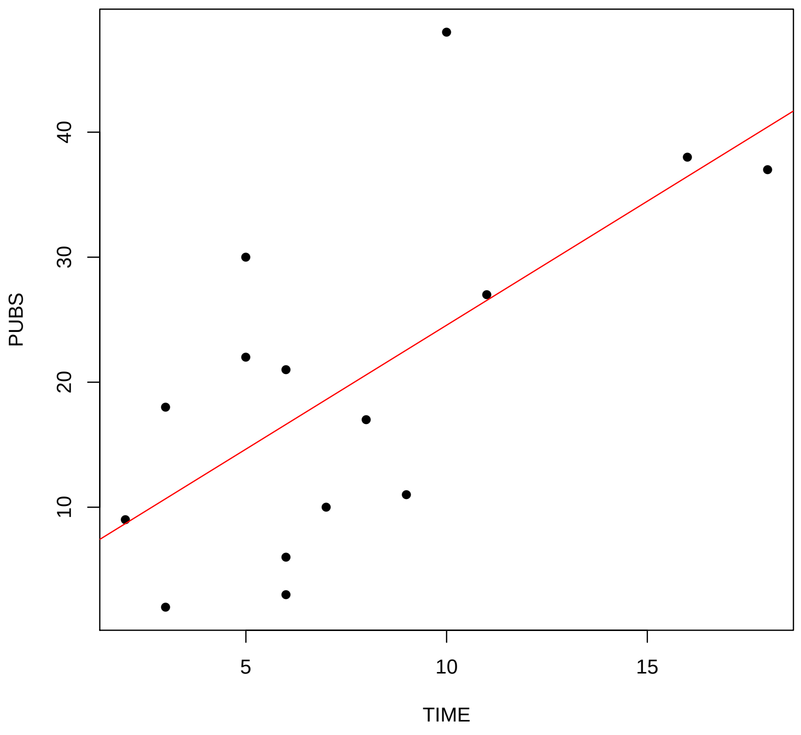


Figure 2.2: Scatterplot with Regression Line (CCAW Figure 2.4.1)

To obtain a confidence interval for the predicted value requires two steps. First, predict the $\hat{Y}$ value for a specific value of X. Second, obtain a confidence interval around that predicted value. Fortunately in R this can be done in a single step using the `predict()` function.

```
predict(PhD.fit, data.frame(TIME = 9), interval="confidence", level=.95)

##    fit  lwr  upr
## 1 22.6 16.3 28.9
```

To compare the difference in regression coefficients, there is no native function in R. Consequently, you have to write a function to do the calculations.

```
# CI for difference in regression coefficients
CI_bDif<-function(b1=NULL, b2=NULL, se1=NULL, se2=NULL, level=.95){
bDif<-b1-b2
bDif_SE<-sqrt(se1^2 + se2^2 )
multiplier<-abs(qnorm((1-level)/2))
bDifUB<-bDif + (multiplier*bDif_SE)
bDifLB<-bDif - (multiplier*bDif_SE)
me<-multiplier*bDif_SE
out<-list(lower.bound=bDifLB, upper.bound=bDifUB)
unlist(out)
}
```

To compare the regression coefficients using the `CI_bDif()` function, use this syntax.

```
CI_bDif(b1=1.98, b2=1.70, se1=.294, se2=.301, level=.955)

## lower.bound upper.bound
##      -0.563       1.123
```

To obtain a confidence interval for a correlation coefficient

```
#Confidence Interval for correlation
library(psychometric) # load package with function
CIr(r=.657, n = 15, level = .95)

## [1] 0.218 0.875
```

There is no function to obtain the analytic confidence interval for the difference in correlations. It can be obtained, however, by writing a function or by using bootstrapping (Efron & Tibshirani, 1994). Here is a function that will calculate the confidence interval analytically (Olkin & Finn, 1995).

```
## CI for difference in correlations
CI_corDif<-function(n1=NULL, n2=NULL, r1=NULL, r2=NULL, level=.95){
corDif<-r1-r2
corDif_SE<-sqrt((1-r1^2)/n1 + (1-r2^2)/n2 )
multiplier<-abs(qnorm((1-level)/2))
corDifUB<-corDif + (multiplier*corDif_SE)
corDifLB<-corDif - (multiplier*corDif_SE)
me<-multiplier*corDif_SE
out<-list(lower.bound=corDifLB, upper.bound=corDifUB)
unlist(out)
}
```

Here is an example of using the function (note that $.657 - .430 = .227$, not $.277$)

```
CI_corDif(62,143, .657,.430, level=.95)

## lower.bound upper.bound
##      -0.012       0.466
```

Bootstrapping requires having access to raw data. So lets add a third variable to the PhD-Publications data:

```
# Add sex variable to PhDPub.data data
PhDPub.data$Sex<-as.factor(c(0,0,0,0,0,0,0,1,1,1,1,1,1,1,1))
```

Now estimate the bootstrapped CI.

```
#bootstrapped CI
library(bootES)

## Error in library(bootES): there is no package called 'bootES'

#Bootstrapped CI
bootES(PhDPub.data[c("TIME","PUBS","Sex")], group.col = "Sex", R=1000, effect.type="r")

## Error in eval(expr, envir, enclos):  could not find function "bootES"
```

2.5 Hypothesis Testing

Null hypothesis testing is part of the `summary()` output from the `lm()` function. It is located in the `t value` column.

To do null hypothesis testing for a correlation, use the `r.test()` function in the `psych` package.

```
library(psych)
r.test(n=15, r12=.657)
```

```
## Correlation tests
## Call:r.test(n = 15, r12 = 0.657)
## Test of significance of a  correlation
##  t value 3.14    with probability < 0.0078
##  and confidence interval  0.22 0.87
```

```
r.test(n=62, r12=.657, r34=.430, n2=143)

## Correlation tests
## Call:r.test(n = 62, r12 = 0.657, r34 = 0.43, n2 = 143)
## Test of difference between two independent correlations
##  z value 2.11    with probability  0.03
```

2.6 Power Analysis

Cohen (1988) uses the f^2 effect size, which can be calculated from a variety of other linear model effect sizes.

$$f^2 = \frac{R^2}{1 - R^2} \tag{2.1}$$

In R, the pwr package has multiple functions for quick power analysis.

```
library(pwr)

## Error in library(pwr):  there is no package called 'pwr'

# Regression
#Sample size needed for a small-sized regression with 3 predictors
pwr.f2.test(u=3,f2=cohen.ES(test="f2",size="small")$effect.size,
power=0.80,sig.level=0.05) ### u = number of predictors

## Error in eval(expr, envir, enclos):  could not find function "pwr.f2.test"
```

To get the estimated sample size needed, combine the v and u parameters.

Chapter 3

Multiple Regression

3.1 Two Predictor Variables

Import the data shown in Table 3.2.1 in CCAW.

```
#PhD/Publication data (Table 3.2.1)
PhDPubSal.data<-read.table("C0301DT.txt", header=TRUE, sep="\t")
```

We can obtain the Mean and SD on single variables by using the `mean()` and `sd()` functions included in **R**.

```
mean(PhDPubSal.data$TIME)

## [1] 7.67

sd(PhDPubSal.data$TIME)

## [1] 4.58
```

We can also obtain the Mean and SD of *all* the variables (columns) at once by using the `colMeans()` and `sapply(*,sd)` functions.

```
colMeans(PhDPubSal.data)

##     TIME     PUBS   SALARY
##     7.67    19.93 53045.60

sapply(PhDPubSal.data,sd)

##    TIME    PUBS  SALARY
##    4.58   13.82 7889.77
```

There is also a `describe()` function in the `psych` package can generate similar descriptive table.

```
library(psych)
describe(PhDPubSal.data)

##        vars  n     mean      sd median  trimmed     mad   min   max range  skew kurtosis      se
## TIME      1 15     7.67    4.58      6     7.31    4.45     2    18    16  0.90    -0.19    1.18
## PUBS      2 15    19.93   13.82     18    19.15   13.34     2    48    46  0.45    -1.03    3.57
## SALARY    3 15 53045.60 7889.77  52926 53087.46 8735.48 39115 66432 27317 -0.03    -1.18 2037.13
```

Based on the provided data, we can form our regression equation with two independent variables: (a) Time since Ph.D and (b) No. of Publications.

```
#Multiple Regression
PhDPubSal.fit<-lm(SALARY~TIME+PUBS, data=PhDPubSal.data)
summary(PhDPubSal.fit) #R2 and coefficients

##
## Call:
## lm(formula = SALARY ~ TIME + PUBS, data = PhDPubSal.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -12065  -3522   -342   3324   8847
##
## Coefficients:
##             Estimate Std. Error t value     Pr(>|t|)
## (Intercept)    43082       3100   13.90 0.0000000093 ***
## TIME             983        452    2.17         0.05 .
## PUBS             122        150    0.81         0.43
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5840 on 12 degrees of freedom
## Multiple R-squared:  0.53,Adjusted R-squared:  0.452
## F-statistic: 6.78 on 2 and 12 DF,  p-value: 0.0107
```

We can use the `anova()` function to extract the SS and MS values from the regression equation.

```
anova(PhDPubSal.fit)

## Analysis of Variance Table
##
## Response: SALARY
##           Df    Sum Sq   Mean Sq F value Pr(>F)
## TIME       1 439746525 439746525   12.90 0.0037 **
## PUBS       1  22572295  22572295    0.66 0.4317
## Residuals 12 409159359  34096613
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We can use the `scale()` function to obtain standardized regression coefficients

```
#Standardizd
PhDPubSal.fit.stand<-lm(scale(SALARY)~0+scale(TIME)+scale(PUBS),data=PhDPubSal.data)
summary(PhDPubSal.fit.stand)

##
## Call:
## lm(formula = scale(SALARY) ~ 0 + scale(TIME) + scale(PUBS), data = PhDPubSal.data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.5293 -0.4465 -0.0433  0.4213  1.1213
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## scale(TIME)    0.570      0.252    2.26    0.041 *
## scale(PUBS)    0.213      0.252    0.85    0.412
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.711 on 13 degrees of freedom
## Multiple R-squared:  0.53,Adjusted R-squared:  0.458
## F-statistic: 7.34 on 2 and 13 DF,  p-value: 0.00734
```

The `0` at the beginning of the equation (i.e., `0`) in the `lm()` function removes the intercept, which is not needed when all the variables are continuous and standardized.

An alternative to using the `scale()` function is to use the `lm.beta()` function in the `QuantPsyc` package.

```
library(QuantPsyc)

## Error in library(QuantPsyc):  there is no package called 'QuantPsyc'

lm.beta(PhDPubSal.fit)

## Error in eval(expr, envir, enclos):  could not find function "lm.beta"
```

To obtain predicted values, we can use the `predict()` function. The difference between the actual and predicted outcome values is called the *residual*, which can be obtained by using the `resid()` function.

```
#This adds the predicted values to the PhDPubSal.data data set
PhDPubSal.data$Salary.pred<-predict(PhDPubSal.fit)
#This adds the residuals to the PhDPubSal.data data set
PhDPubSal.data$Resid<-resid(PhDPubSal.fit)

head(PhDPubSal.data)

##   TIME PUBS SALARY Salary.pred Resid
## 1    3   18  51876       48223  3653
## 2    6    3  54511       49345  5166
## 3    3    2  53425       46275  7150
## 4    8   17  61863       53016  8847
## 5    9   11  52926       53268  -342
## 6    6    6  47034       49710 -2676
```

The `ppcor` package in `R` efficiently calculates partial and semi-partial/part correlations.

```
library(ppcor)
# Semi-partial
spcor(PhDPubSal.data[c("TIME","PUBS","SALARY")])

## $estimate
##         TIME  PUBS SALARY
## TIME    1.00 0.296  0.401
## PUBS    0.34 1.000  0.172
## SALARY 0.43 0.161  1.000
##
## $p.value
##          TIME  PUBS SALARY
## TIME   0.000 0.305  0.155
## PUBS   0.235 0.000  0.556
## SALARY 0.125 0.583  0.000
##
## $statistic
##         TIME  PUBS SALARY
## TIME    0.00 1.072  1.516
## PUBS    1.25 0.000  0.606
## SALARY 1.65 0.565  0.000
##
## $n
## [1] 15
##
## $gp
## [1] 1
```

```
##
## $method
## [1] "pearson"

# Partial
pcor(PhDPubSal.data[c("TIME","PUBS","SALARY")])

## $estimate
##         TIME  PUBS SALARY
## TIME   1.000 0.420  0.532
## PUBS   0.420 1.000  0.229
## SALARY 0.532 0.229  1.000
##
## $p.value
##          TIME  PUBS SALARY
## TIME   0.0000 0.135 0.0504
## PUBS   0.1350 0.000 0.4317
## SALARY 0.0504 0.432 0.0000
##
## $statistic
##        TIME  PUBS SALARY
## TIME   0.00 1.603  2.174
## PUBS   1.60 0.000  0.814
## SALARY 2.17 0.814  0.000
##
## $n
## [1] 15
##
## $gp
## [1] 1
##
## $method
## [1] "pearson"
```

3.2 Multiple Predictor Variables

Import the data shown in Table 3.5.1 in CCAW.

```
#PhD/Publication data (Table 3.2.1)
PhDPubSalSex.data<-read.table("C0302DT.txt", header=TRUE, sep="\t")
```

Use the `cor()` function to obtain the correlations among the variables. A scatterplot matrix is shown in Figure 3.1.

```
cor(PhDPubSalSex.data)

##            CASE   TIME    PUBS    CITS   SALARY FEMALE
## CASE    1.00000 -0.081 -0.0618  0.0268  0.00493  0.130
## TIME   -0.08103  1.000  0.6505  0.3729  0.60790 -0.210
## PUBS   -0.06180  0.651  1.0000  0.3334  0.50615 -0.159
## CITS    0.02683  0.373  0.3334  1.0000  0.54977 -0.149
## SALARY  0.00493  0.608  0.5061  0.5498  1.00000 -0.201
## FEMALE  0.12997 -0.210 -0.1588 -0.1492 -0.20096  1.000
```

```
# Scatterplot Matrix with Histograms on the Diagonal
panel.hist <- function(x){
    usr <- par("usr"); on.exit(par(usr))
    par(usr = c(usr[1:2], 0, 1.5) )
    h <- hist(x, plot = FALSE)
```

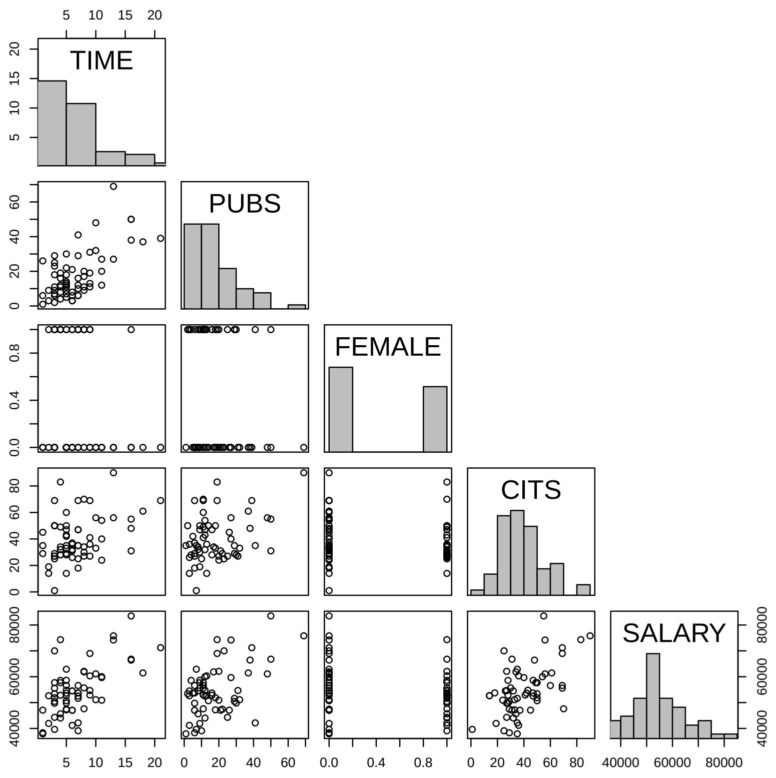


Figure 3.1: Scatterplot Matrix for Academic Salary Example Data, with Histograms on the Diagonal

```
    breaks <- h$breaks; nB <- length(breaks)
    y <- h$counts; y <- y/max(y)
    rect(breaks[-nB], 0, breaks[-1], y, col="gray")
}
pairs(PhDPubSalSex.data[c("TIME", "PUBS","FEMALE","CITS","SALARY")],
diag.panel=panel.hist,upper.panel=NULL)
```

The four-predictor regression equation the new data set is basically the same as with two predictors

```
PhDPubSalSex.fit<-lm(SALARY~TIME+PUBS+CITS+FEMALE, data=PhDPubSalSex.data)
# Regression Coefficinets, standard errors, R^2, adjusted R^2 and F
summary(PhDPubSalSex.fit)

##
## Call:
## lm(formula = SALARY ~ TIME + PUBS + CITS + FEMALE, data = PhDPubSalSex.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -13377  -4482   -990   4316  20671
##
## Coefficients:
##             Estimate Std. Error t value             Pr(>|t|)
## (Intercept)  39587.3     2717.5   14.57 < 0.0000000000000002 ***
## TIME           857.0      287.9    2.98              0.00428 **
## PUBS            92.7       85.9    1.08              0.28498
## CITS           201.9       57.5    3.51              0.00088 ***
## FEMALE        -917.8     1859.9   -0.49              0.62360
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 7080 on 57 degrees of freedom
## Multiple R-squared:  0.503,Adjusted R-squared:  0.468
## F-statistic: 14.4 on 4 and 57 DF,  p-value: 0.0000000336
```

```
#  For standardized coefficients

PhDPubSalSex.stand.data<-data.frame(scale(PhDPubSalSex.data))
PhDPubSalSex.stand.fit<-lm(SALARY~0+TIME+PUBS+CITS+FEMALE, data=PhDPubSalSex.stand.data)
summary(PhDPubSalSex.stand.fit)

##
## Call:
## lm(formula = SALARY ~ 0 + TIME + PUBS + CITS + FEMALE, data = PhDPubSalSex.stand.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -1.378 -0.462 -0.102  0.445  2.130
##
## Coefficients:
##        Estimate Std. Error t value Pr(>|t|)
## TIME     0.3777     0.1258    3.00  0.00395 **
## PUBS     0.1338     0.1229    1.09  0.28076
## CITS     0.3573     0.1009    3.54  0.00079 ***
## FEMALE  -0.0473     0.0950   -0.50  0.62054
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.723 on 58 degrees of freedom
## Multiple R-squared:  0.503,Adjusted R-squared:  0.469
## F-statistic: 14.7 on 4 and 58 DF,  p-value: 0.000000024
```

```
# Calculate the Predicted Salaries and add them to the data
PhDPubSalSex.data$Salary.pred<-predict(PhDPubSalSex.fit)
```

Now we can re-create Table 3.5.1 in CCAW

```
head(PhDPubSalSex.data[c(2:3,6,4:5,7)])

##   TIME PUBS FEMALE CITS SALARY Salary.pred
## 1    3   18      1   50  51876       53007
## 2    6    3      1   26  54511       49340
## 3    3    2      1   50  53425       51523
## 4    8   17      0   34  61863       54886
## 5    9   11      1   41  52926       55682
## 6    6    6      0   37  47034       52757

describe(PhDPubSalSex.data[c(2:3,6,4:5,7)])

##             vars  n     mean      sd median  trimmed     mad   min   max range skew kurtosis
## TIME           1 62     6.79    4.28      6     6.22    2.97     1    21    20 1.23     1.29
## PUBS           2 62    18.18   14.00     13    16.28   10.38     1    69    68 1.31     1.61
## FEMALE         3 62     0.44    0.50      0     0.42    0.00     0     1     1 0.25    -1.97
## CITS           4 62    40.23   17.17     35    39.08   14.08     1    90    89 0.65     0.32
## SALARY         5 62 54815.76 9706.02  53681 54226.10 9119.47 37939 83503 45564 0.61     0.25
## Salary.pred    6 62 54815.76 6885.43  52972 54061.91 4789.28 43010 75302 32292 1.08     1.06
##                 se
## TIME          0.54
## PUBS          1.78
## FEMALE        0.06
## CITS          2.18
## SALARY     1232.67
## Salary.pred  874.45
```

The semi-partial and partial correlations:

```
spcor(PhDPubSalSex.data[c("TIME", "PUBS","FEMALE","CITS","SALARY")])

## $estimate
##           TIME    PUBS   FEMALE    CITS   SALARY
## TIME    1.0000  0.3904 -0.06193  0.0133  0.2698
## PUBS    0.4252  1.0000 -0.00808  0.0405  0.1066
## FEMALE -0.0880 -0.0105  1.00000 -0.0393 -0.0636
## CITS    0.0162  0.0452 -0.03362  1.0000  0.3869
## SALARY  0.2778  0.1008 -0.04606  0.3278  1.0000
##
## $p.value
##            TIME    PUBS FEMALE   CITS   SALARY
## TIME   0.000000 0.00224  0.641 0.9202 0.03879
## PUBS   0.000788 0.00000  0.952 0.7607 0.42179
## FEMALE 0.507431 0.93682  0.000 0.7676 0.63246
## CITS   0.903092 0.73385  0.800 0.0000 0.00247
## SALARY 0.033116 0.44765  0.729 0.0113 0.00000
##
## $statistic
##          TIME    PUBS FEMALE   CITS SALARY
## TIME    0.000  3.2012 -0.468  0.101  2.115
## PUBS    3.547  0.0000 -0.061  0.306  0.809
## FEMALE -0.667 -0.0796  0.000 -0.297 -0.481
## CITS    0.122  0.3417 -0.254  0.000  3.168
## SALARY  2.184  0.7646 -0.348  2.620  0.000
##
## $n
## [1] 62
##
## $gp
## [1] 3
##
## $method
## [1] "pearson"

pcor(PhDPubSalSex.data[c("TIME", "PUBS","FEMALE","CITS","SALARY")])

## $estimate
##           TIME    PUBS  FEMALE    CITS  SALARY
## TIME    1.0000  0.4955 -0.0901  0.0195  0.3667
## PUBS    0.4955  1.0000 -0.0108  0.0543  0.1415
## FEMALE -0.0901 -0.0108  1.0000 -0.0404 -0.0652
## CITS    0.0195  0.0543 -0.0404  1.0000  0.4217
## SALARY  0.3667  0.1415 -0.0652  0.4217  1.0000
##
## $p.value
##             TIME      PUBS FEMALE     CITS   SALARY
## TIME   0.0000000 0.0000661  0.497 0.883665 0.004276
## PUBS   0.0000661 0.0000000  0.935 0.683152 0.284979
## FEMALE 0.4972225 0.9350394  0.000 0.761391 0.623597
## CITS   0.8836650 0.6831524  0.761 0.000000 0.000879
## SALARY 0.0042759 0.2849794  0.624 0.000879 0.000000
##
## $statistic
##          TIME    PUBS  FEMALE   CITS SALARY
## TIME    0.000  4.3067 -0.6832  0.147  2.976
## PUBS    4.307  0.0000 -0.0819  0.410  1.079
## FEMALE -0.683 -0.0819  0.0000 -0.305 -0.493
## CITS    0.147  0.4103 -0.3051  0.000  3.511
## SALARY  2.976  1.0793 -0.4934  3.511  0.000
##
```

```
## $n
## [1] 62
##
## $gp
## [1] 3
##
## $method
## [1] "pearson"
```

The 80% and 95% confidence intervals for the regression coefficients can be obtained using the

```
# Unstandardized
confint(PhDPubSalSex.fit,level = 0.80)

##                10 %  90 %
## (Intercept) 36063.9 43111
## TIME          483.7  1230
## PUBS          -18.7   204
## CITS          127.4   276
## FEMALE      -3329.3  1494

confint(PhDPubSalSex.fit,level = 0.95)

##               2.5 % 97.5 %
## (Intercept) 34145.7  45029
## TIME          280.4   1434
## PUBS          -79.3    265
## CITS           86.8    317
## FEMALE      -4642.2   2807

# Standardized
confint(PhDPubSalSex.stand.fit, level=.80)

##           10 %   90 %
## TIME    0.2146 0.5408
## PUBS   -0.0255 0.2931
## CITS    0.2265 0.4880
## FEMALE -0.1704 0.0758

confint(PhDPubSalSex.stand.fit, level=.95)

##         2.5 % 97.5 %
## TIME    0.126  0.630
## PUBS   -0.112  0.380
## CITS    0.155  0.559
## FEMALE -0.237  0.143
```

Use the `CI.Rsqlm()` function in the `psychometric` package to get the standard error and confidence interval for the R^2

```
library(psychometric)
# 80% CI
CI.Rsqlm(PhDPubSalSex.fit, level=.80)

##     Rsq  SErsq LCL   UCL
## 1 0.503 0.0804 0.4 0.606

# 95% CI
CI.Rsqlm(PhDPubSalSex.fit, level=.95)

##     Rsq  SErsq   LCL   UCL
## 1 0.503 0.0804 0.346 0.661
```

The correlation of the predictor variables with the predicted value of the outcome is called the *structure coefficient*.

The `yhat` package will estimate the structure coefficients.

```
library(yhat)
regr(PhDPubSalSex.fit)$Structure_Coefficients

##       TIME  PUBS  CITS FEMALE
## [1,] 0.857 0.713 0.775 -0.283
```

Chapter 4

Data Visualization Exploration, and Assumption Checking: Diagnosing and Solving Regression Problems 1

Import the data PhD-Publications data.

```
#PhD/Publication data
PhDPubSalSex.data <- read.table("C04e01dt1.txt")
names(PhDPubSalSex.data) <- c("CASE", "CASE2", "TIME", "PUBS", "FEMALE", "CITS", "SALARY")
```

4.1 Useful Graphical Displays

4.1.1 Univariate Displays

4.1.1.1 Histograms

Histograms can be obtained by using the `hist()` function. On the x-axis, the number of intervals of equal width is controlled by the `breaks` argument. Therefore, by assigning different values to `breaks`, we can present data with different number of bins.

```
hist(PhDPubSalSex.data$TIME, breaks=5, xlab="Years since Ph.D.", ylab="", main="Five bins")
hist(PhDPubSalSex.data$TIME, breaks=20, xlab="Years since Ph.D.", ylab="", main="Twenty bins", xlim=c(0,25))
```

The univariate histograms of Years since Ph.D. are shown in Figure 4.1

4.1.1.2 Stem-and-Leaf Displays

A *stem-and-leaf plot* is another way to represent the frequency pattern of data. Use the `stem()` function to generate a stem and leaf plot. The interval width is controlled by the `scale` argument.

```
stem(PhDPubSalSex.data$TIME,scale=0.5)

##
##   The decimal point is 1 digit(s) to the right of the |
##
##   0 | 11122333333333444444
##   0 | 5555555555666666777777788889999
##   1 | 0011133
##   1 | 6668
##   2 | 1
```

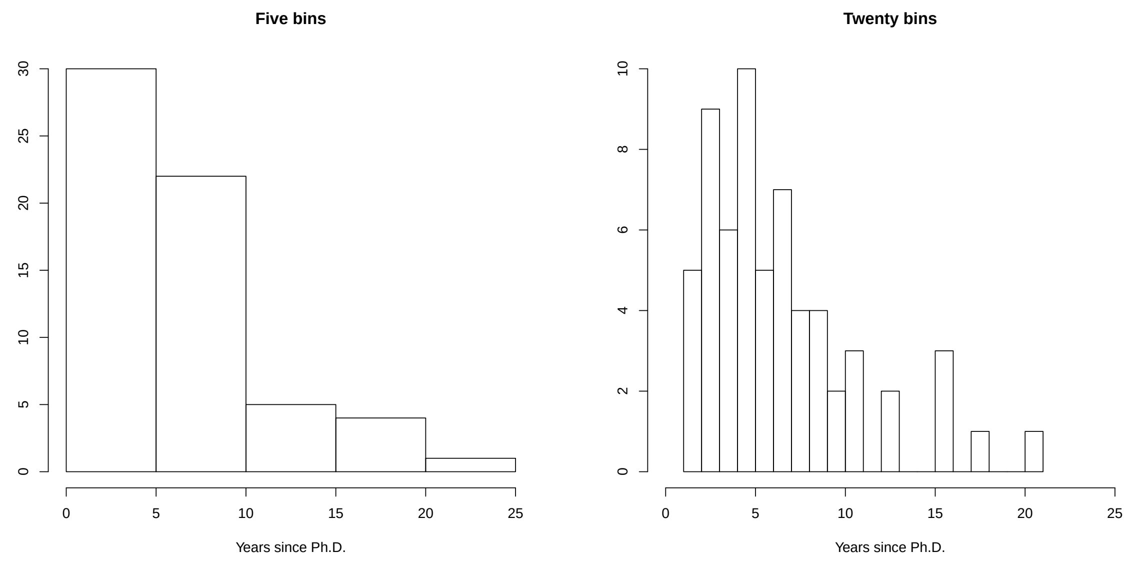


Figure 4.1: Histograms of Years since Ph.D

4.1.1.3 Kernel Density Plots

Another way to visually present frequency distributions is via kernel density plots, which is done via the `plot(density())` function in **R**. The degree of smoothing is controlled by the `bw` argument, which represents bandwidth. Some example kernel density plots are given in Figure 4.2.

```
plot(density(PhDPubSalSex.data$TIME,bw=4), xlim=c(0,25), ylim=c(0,0.15), xlab="Year since Ph.D")
```

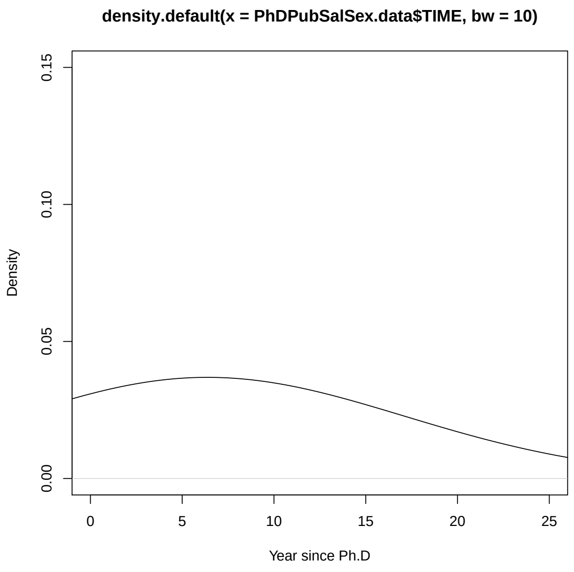
We can also superimpose a density plot on top of a histogram by using the `lines()` function. An example is given in Figure 4.3.

```
hist(PhDPubSalSex.data$TIME, breaks=5, probability=TRUE, xlab="Year since Ph.D")
lines(density(PhDPubSalSex.data$TIME,bw=4))
```

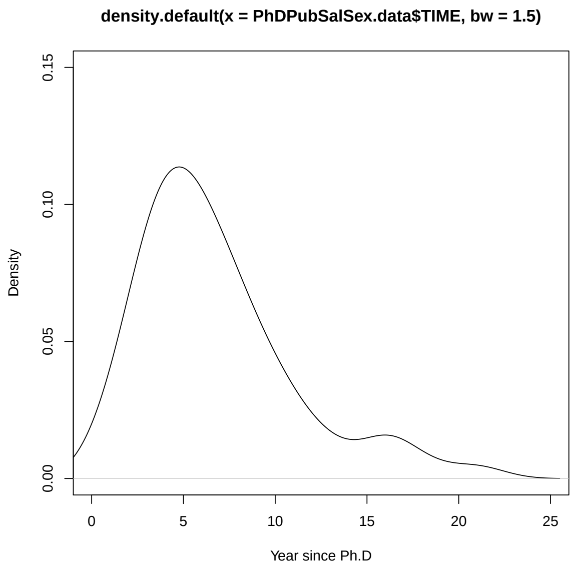
4.1.1.4 Boxplot

We can use the `boxplot()` function for creating boxplots (aka box-and-whisker plots).

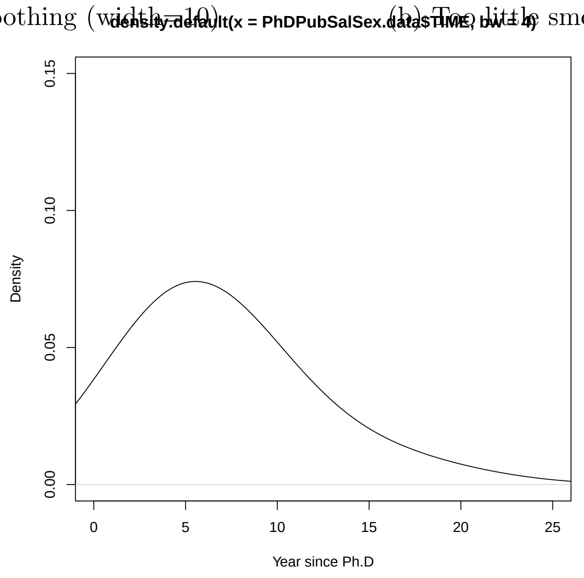
```
boxplot(PhDPubSalSex.data$TIME,  ylab="Year since Ph.D")
```



(a) Too much smoothing (width=10).



(b) Too little smoothing (width=1.5).



(c) Appropriate smoothing (width=4).

Figure 4.2: Kernel density plots.

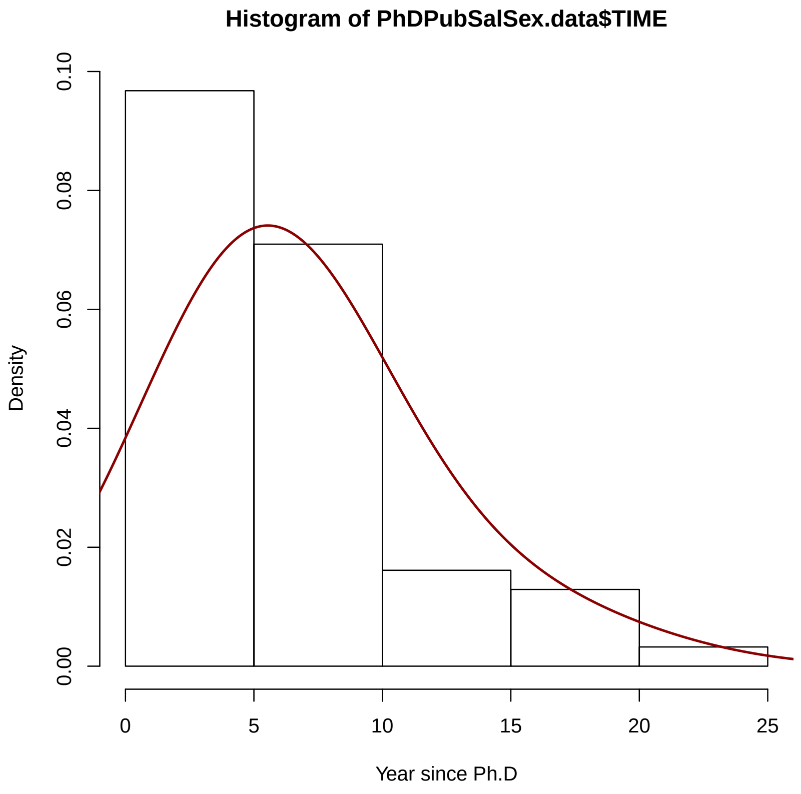
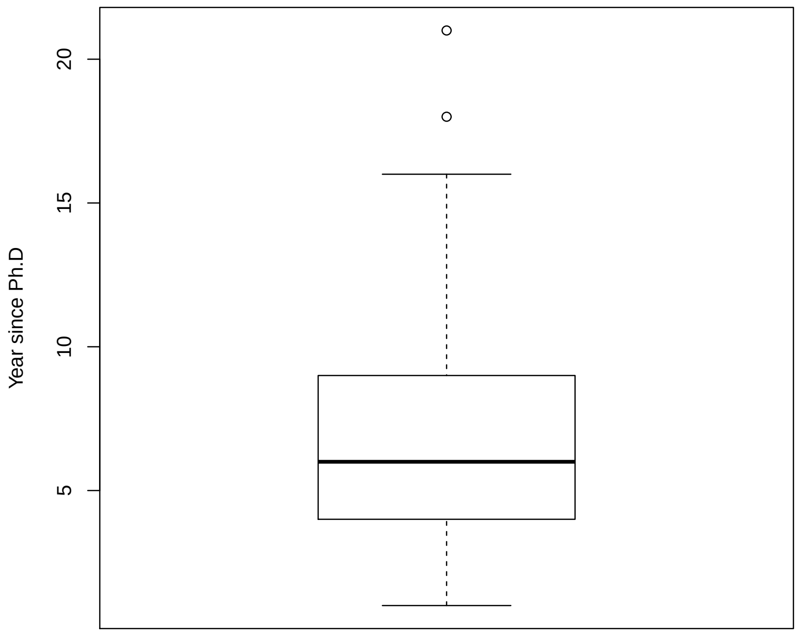
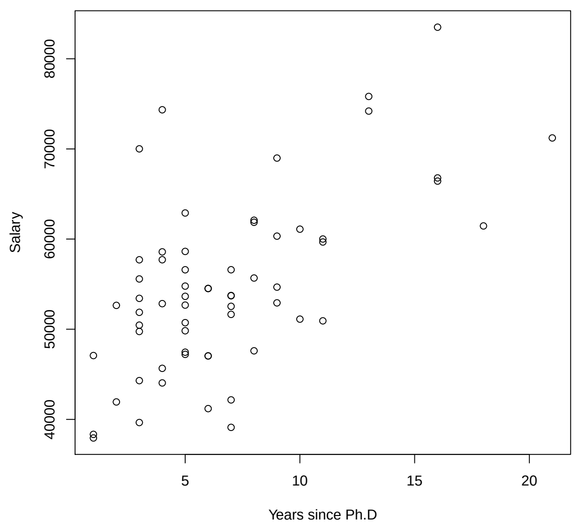


Figure 4.3: Histogram with superimposed kernel density plot.

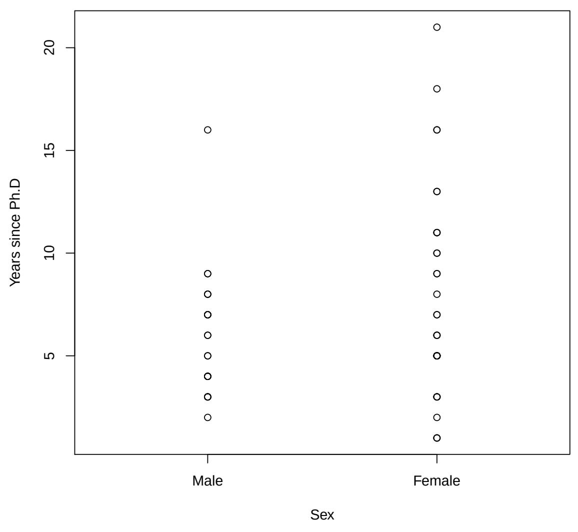


4.1.2 Bivariate Displays

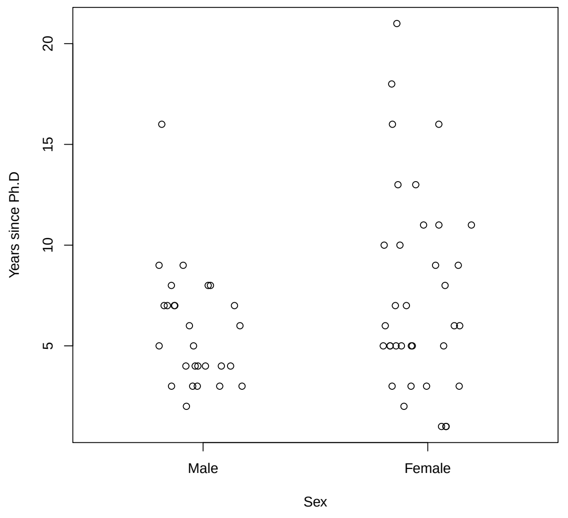
When examining the relationship between two variables, we can use `plot()` function to create scatterplots, while the first argument represents the variable on the x axis and the second



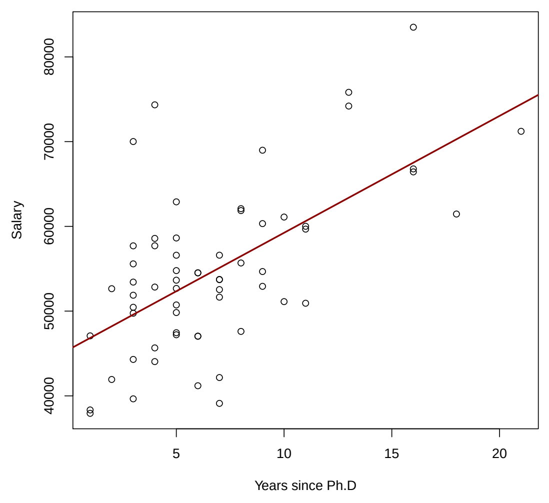
(a) Salary vs. Years since Ph.D



(b) Sex vs. Years since Ph.D



(c) Jittered scatterplot: Sex vs. Years since Ph.D



(d) Salary vs. Years since Ph.D. with Superimposed Regression Line

Figure 4.4: Scatterplots.

argument represents the variable on the y axis. Some examples are shown in Figure 4.4.

A *jittered* scatterplot can be obtained by using the `jitter()` function on each variable. An example is given in Figure 4.4c.

We can superimpose the regression line to the scatterplot by using `abline()` function. An example is shown in Figure 4.4d.

```
# Scatterplot with continuous variables
plot(PhDPubSalSex.data$TIME, PhDPubSalSex.data$SALARY, xlab="Years since Ph.D", ylab="Salary")
# Scatterplot with a categorical variable
plot(PhDPubSalSex.data$FEMALE, PhDPubSalSex.data$TIME, xlim=c(-.5,1.5), xlab="Sex",
ylab="Years since Ph.D",xaxt = "n")
axis(1, at=0:1, labels=c("Male","Female"))
# Scatterplot with a jittered categorical variable
plot(jitter(PhDPubSalSex.data$FEMALE), PhDPubSalSex.data$TIME, xlim=c(-.5,1.5), xlab="Sex",
ylab="Years since Ph.D", xaxt = "n")
axis(1, at=0:1, labels=c("Male","Female"))
```

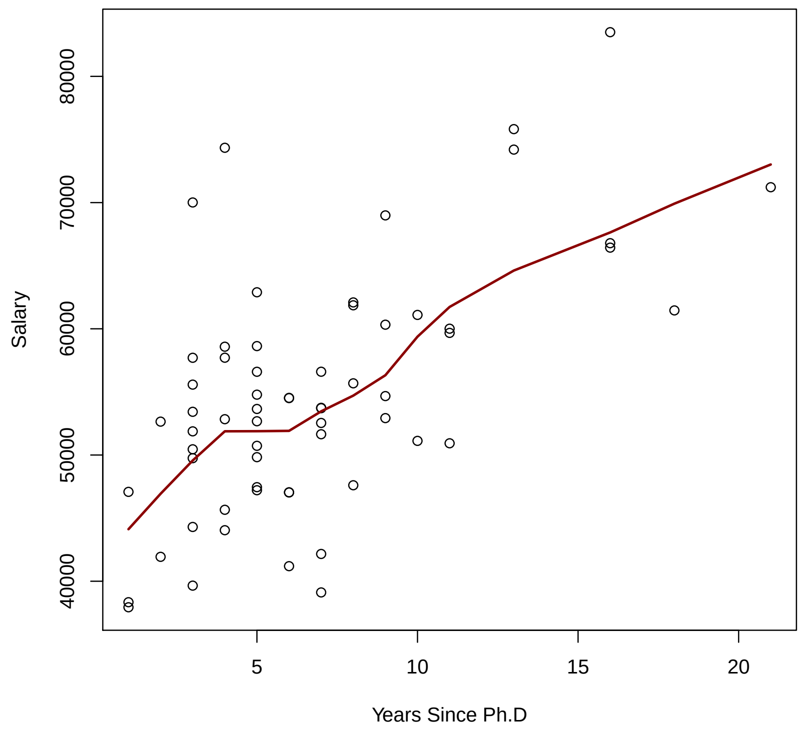


Figure 4.5: Superimposed LOWESS fit: Salary vs. Years since Ph.D

```
# Scatterplot with a regression line
plot(PhDPubSalSex.data$TIME, PhDPubSalSex.data$SALARY, xlab="Years since Ph.D", ylab="Salary")
abline(lm(SALARY~TIME, data=PhDPubSalSex.data))
```

Locally weighted scatterplot smoothing (LOWESS) regression is a non-parametric regression method that combines multiple regression models using small subsets of the data. For a readable introduction to LOWESS regressions, see Trexler and Travis (1993). The `lowess()` function, used in conjunction with the `lines()` function, can be used for superimposing LOWESS lines onto plots. The degree of LOWESS smoothing is controlled by the `alpha` argument. An example is given in Figure 4.5

```
plot(PhDPubSalSex.data$TIME, PhDPubSalSex.data$SALARY, xlab="Years Since Ph.D", ylab="Salary")
lines(lowess(PhDPubSalSex.data$TIME, PhDPubSalSex.data$SALARY))
```

4.1.3 Correlation and Scatterplot Matrices

The correlation table matrix between multiple variables can be obtained by using the `cor()` function. Moreover, using `[ ]` and the variable names (or location), we can subset of data set to include just the variables of interest.[1]

```
cor(PhDPubSalSex.data[c("TIME", "PUBS", "FEMALE", "CITS", "SALARY")])
```

[1]To make Table 4.1, I used the `xtable()` function in the `xtable` package.

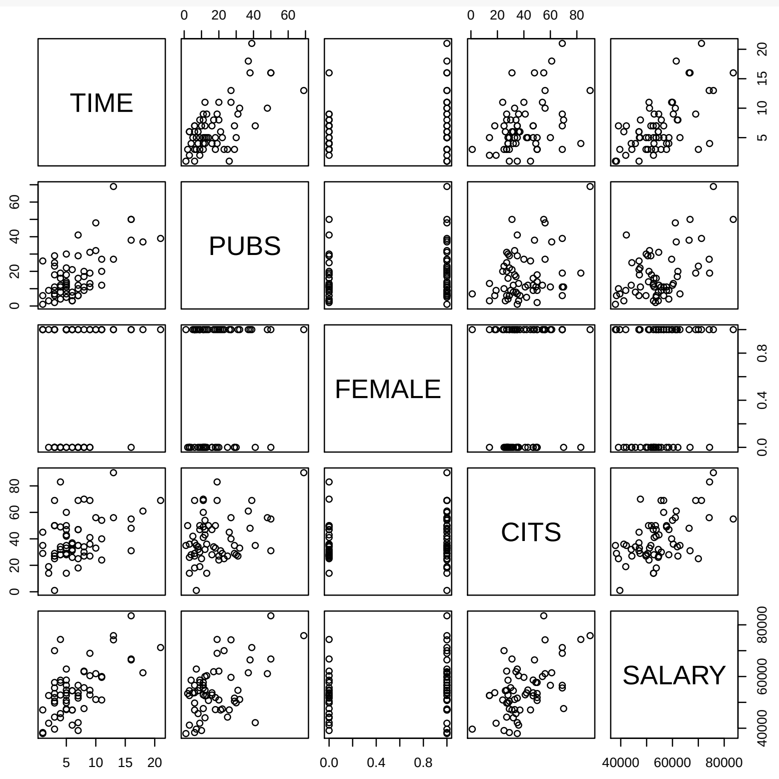
```
xtable(cor(PhDPubSalSex.data[c("TIME", "PUBS", "FEMALE", "CITS", "SALARY")]),
caption="Correlation Matrix for Faculty Salary Example.", label="tab:FacSalaryCorrelationTable")
```

Table 4.1: Correlation Matrix for Faculty Salary Example.

	TIME	PUBS	FEMALE	CITS	SALARY
TIME	1.00	0.65	0.21	0.37	0.61
PUBS	0.65	1.00	0.16	0.33	0.51
FEMALE	0.21	0.16	1.00	0.15	0.20
CITS	0.37	0.33	0.15	1.00	0.55
SALARY	0.61	0.51	0.20	0.55	1.00

Instead of creating multiple individual scatterplots between each pair of variables, we can create a matrix of scatterplots for multiple bivariate pairs using the `pairs()` function.

```r
pairs(~TIME+PUBS+FEMALE+CITS+SALARY,data=PhDPubSalSex.data)
```



4.2 Detecting Violations of Assumptions

4.2.1 Form of the relationship

The linearity assumption is examined by comparing the LOWESS line of fit to the mean of residuals line in a scatterplot. In order to overlay the mean of residuals line, we can use the `abline()` function, using the `h=0` argument to request a horizontal line at $y = 0$. Note that the native `resid()` and `predict()` functions, when used on a `lm` object, will produce the residuals and predicted values, respectively, for each observation in the data set.

Before graphing anything, we need to fit the regression model.

```r
PhD.full.fit <- lm(SALARY ~ TIME + PUBS + FEMALE + CITS, data=PhDPubSalSex.data)
```

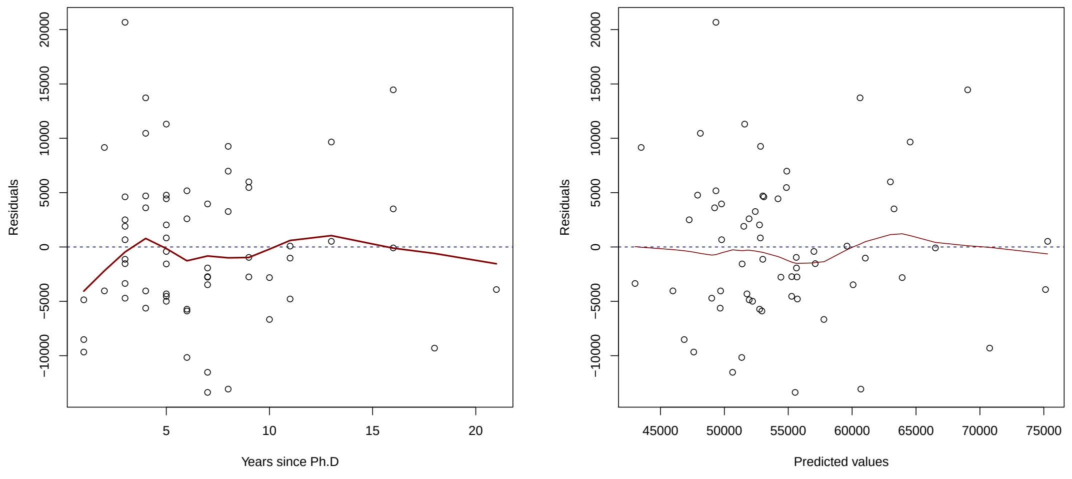


Figure 4.6: Residual Plots

```
# Variable vs. residuals
plot(PhDPubSalSex.data$TIME,residuals(PhD.full.fit),xlab="Years since Ph.D", ylab="Residuals")
#Add a 0-line (Mean of residuals line)
abline(h=0, col="blue4", lty=2)
#Add a LOWESS fit line
lines(lowess(PhDPubSalSex.data$TIME,residuals(PhD.full.fit)), lwd=2, col="red4")
# Predicted vs. residuals
plot(predict(PhD.full.fit),residuals(PhD.full.fit), xlab="Predicted values",ylab="Residuals")
abline(h=0, col="blue4", lty=2)
lines(lowess(predict(PhD.full.fit),residuals(PhD.full.fit)), col="red4")
```

4.2.2 Omitted Independent Variable

In order to examine the influence from each potential predictors, added variable plots can be obtained by using the `avPlots()` function from `car` package. An example is given in Figure 4.7

```
library(car)
avPlots(PhD.full.fit, main="", col.lines="red4", lwd=2, grid=FALSE)
```

4.2.3 Homoscedasticity of Residuals

Various residual plots are given in Figure 4.8. To examine the pattern of residuals, we can add several LOWESS lines to define a confidence intervals (really, a *confidence band*) . There are multiple ways to accomplish this. First, use the `loess.sd()` function in the `msir` package, which will allow us to add lines ±1 SD. An example plot is given in Figure 4.8c.

```
# Residuals vs. a variable
plot(resid(PhD.full.fit)~PhDPubSalSex.data$TIME, xlab="Years since Ph.D", ylab="Residuals")
abline(h=0,col="blue4", lty=2)
# Residuals vs. predicted values
plot(resid(PhD.full.fit)~predict(PhD.full.fit), xlab="Predicted values", ylab="Residuals")
abline(h=0, col="blue4", lty=2)
# Plot with LOWESS lines +/- 1 SD
library(msir)
PhD.full.loess <- loess.sd(resid(PhD.full.fit)~PhDPubSalSex.data$TIME, degree=1)
```

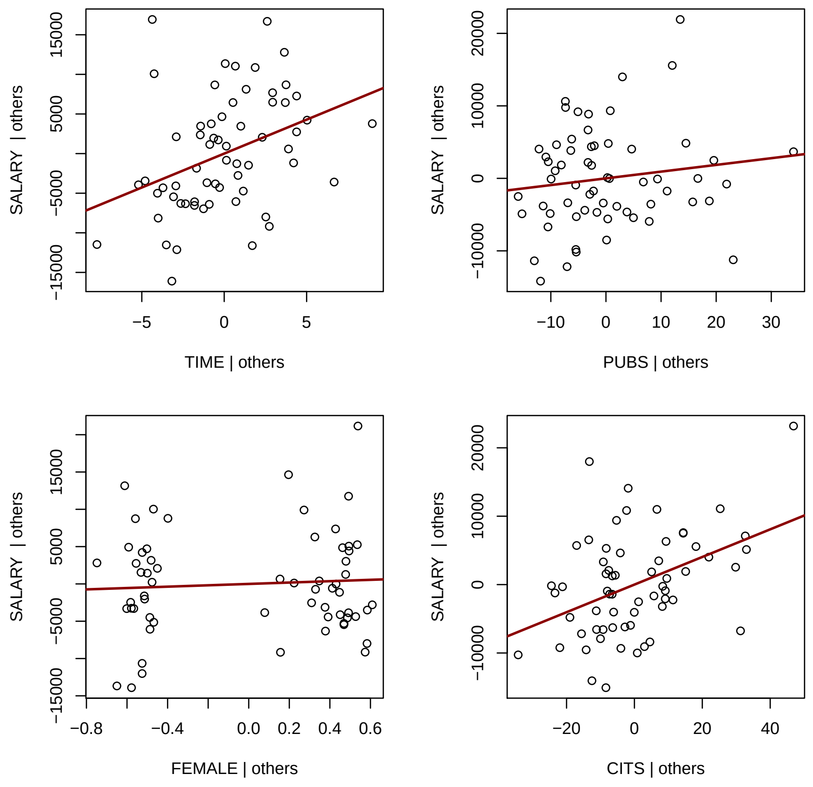


Figure 4.7: Added Variable Plots

```
plot(resid(PhD.full.fit)~PhDPubSalSex.data$TIME,  ylim=c(-20000, 30000), xlim=c(0,25),
xlab="Years since Ph.D", ylab="Residuals")
lines(PhD.full.loess, col="red4", lwd=2)
lines(PhD.full.loess$x, PhD.full.loess$upper, lty=2)
lines(PhD.full.loess$x, PhD.full.loess$lower, lty=2)
```

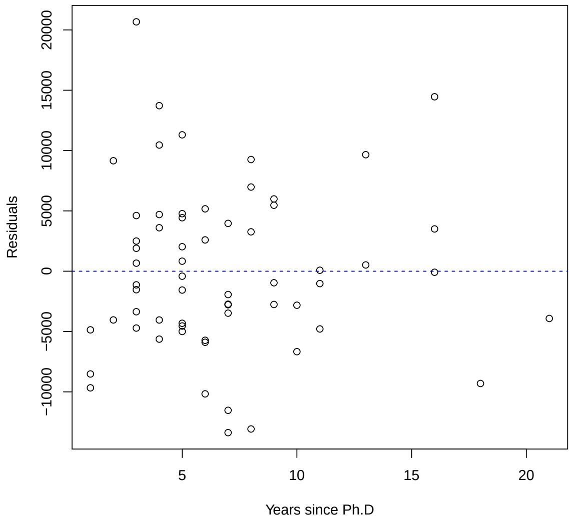
An alternative way to plot a LOWESS line with a confidence band is to use the `ggplot2` package. The `ggplot2` syntax is different from many of other functions in **R**, so the author of the package has written an entire book on how to use it (Wickham, 2009), as well as supports an website with many examples [http://ggplot2.org]. For our purposes, the `geom_smooth()` function will add a shadowed area representing a ±1 SD confidence band around the LOWESS fit line.

```
library(ggplot2)
qplot (x = TIME, y = resid(PhD.full.fit), data = PhDPubSalSex.data) +
geom_abline(slope = 0, intercept = 0) + geom_smooth() + ylab("Residuals")
```

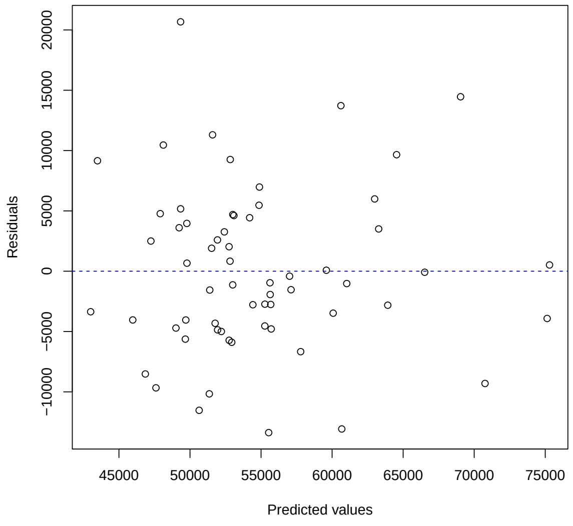
4.2.4 Nonindependence of Residuals

```
# Import non-independent data
cluster.data <- read.table("C04e01dt2.txt")
names(cluster.data) <- c("CASE", "CLUSTER", "VAR1", "VAR2", "VAR3")
```

```
# Regression of non-independent data
cluster.fit <- lm(VAR2~VAR1 + VAR3 , data=cluster.data)
```

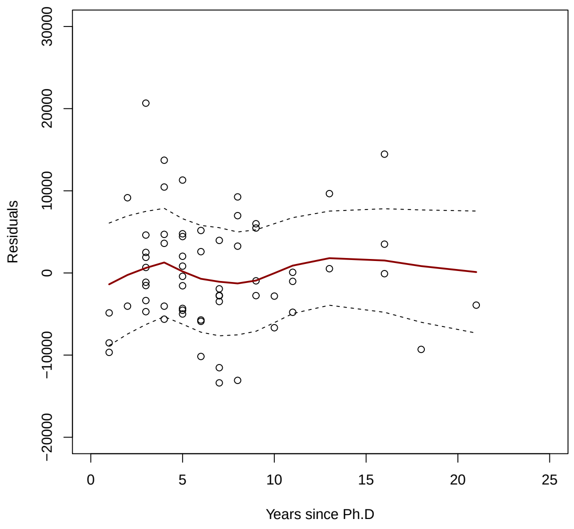


(a) Residuals vs. Years since Ph.D.



(b) Residuals vs. Predicted values.

```
## Warning in rgl.init(initValue, onlyNULL):
      RGL: unable to open X11 display
## Warning:  'rgl_init' failed, running with
            rgl.useNULL = TRUE
```



(c) Residuals vs. years since Ph.D. (LOWESS fit added).

Figure 4.8: Plots of residuals

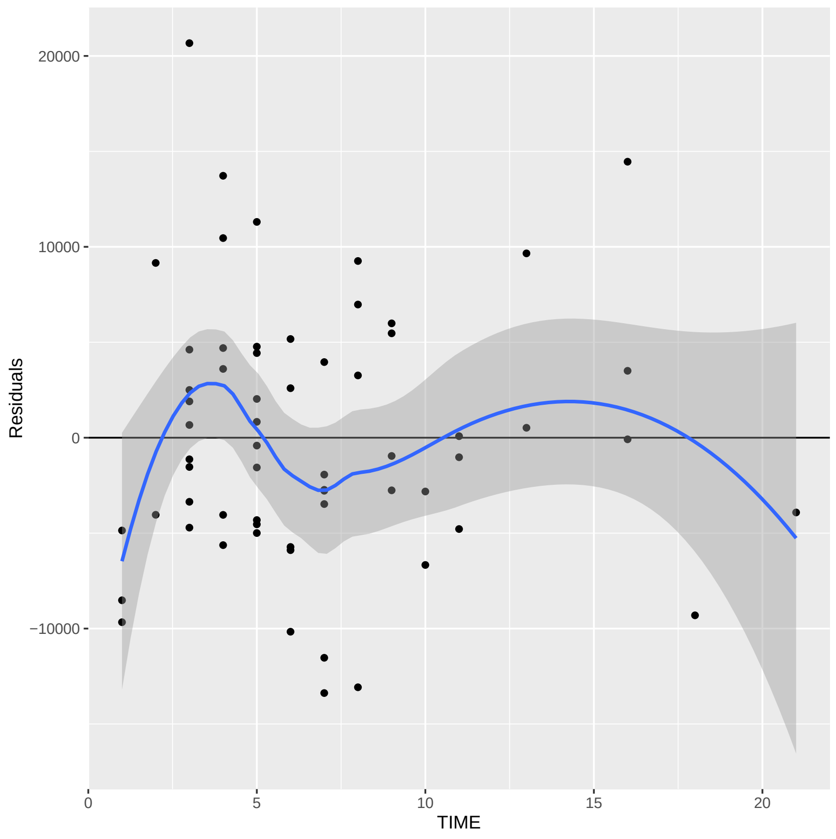


Figure 4.9: Residuals vs. years since Ph.D. with LOWESS fit added (ggplot2 version)).

One way of assessing this assumption is to plot the residuals against the ID number. An example is given in Figure 4.10.

```
pchs <- c(3,5,4,0,2,6,1,7,8,9)
plot(cluster.data$CASE,resid(cluster.fit),  xlab="Case Number", ylab="Residuals", ylim=c(-8,8), pch=pchs[cluste
abline(h=0, lty=2, col="blue4")
```

Another way is to plot the data by clusters (e.g., school, lab where collected data), if they are known. An example is given in Figure 4.11.

```
boxplot(resid(cluster.fit)~cluster.data$CLUSTER, xlab="Cluster", ylab="Residuals",ylim=c(-8,8))
```

4.2.5 Normality of the Residuals

One way to examine the normality of residuals is by using the histogram overlaid with a normal curve. In **R**, the `curve()` function can be used to plott curved lines and `dnorm()` will produce the density of a point that follows a normal distribution. An example is given in Figure 4.12.

```
hist(resid(PhD.full.fit), probability=TRUE,breaks=14, xlab="Values of residuals")
# Mean and SD of residuals to use for the location and dispersion of the normal curve
m<-mean(resid(PhD.full.fit)); std<-sd(resid(PhD.full.fit))
curve(dnorm(x, mean=m, sd=std), col="red4", lwd=2, add=TRUE)
```

Another way to examine the normality of residuals is using q-q plots. In `R` there are multiple packages offering this function. The `car` packages has the `qqPlot()` function, and the confidence interval is controlled by the `envelope` argument.

```
library(car)
qqPlot(resid(PhD.full.fit), ylab="Residuals", xlab="Normal Quantiles",col.lines="red4", grid=FALSE)
```

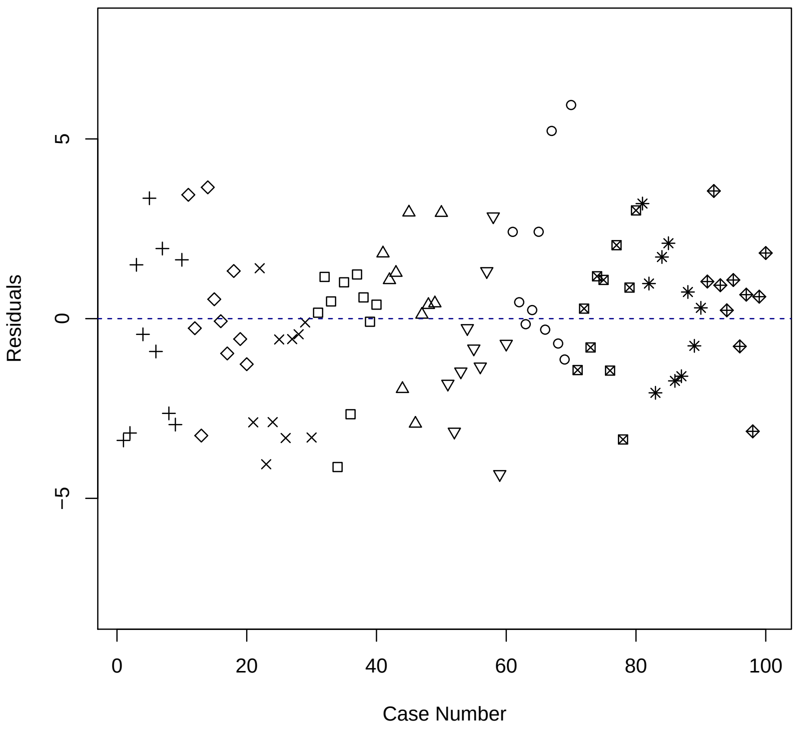


Figure 4.10: Scatterplot of residuals vs. observations using a different symbol for each cluster

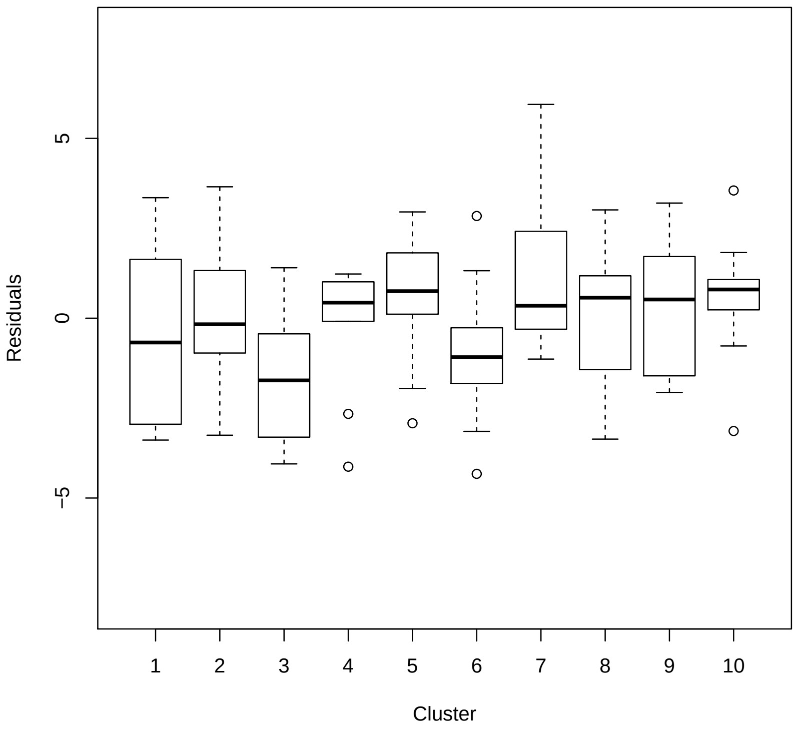


Figure 4.11: Side by side boxplots of the 10 clusters

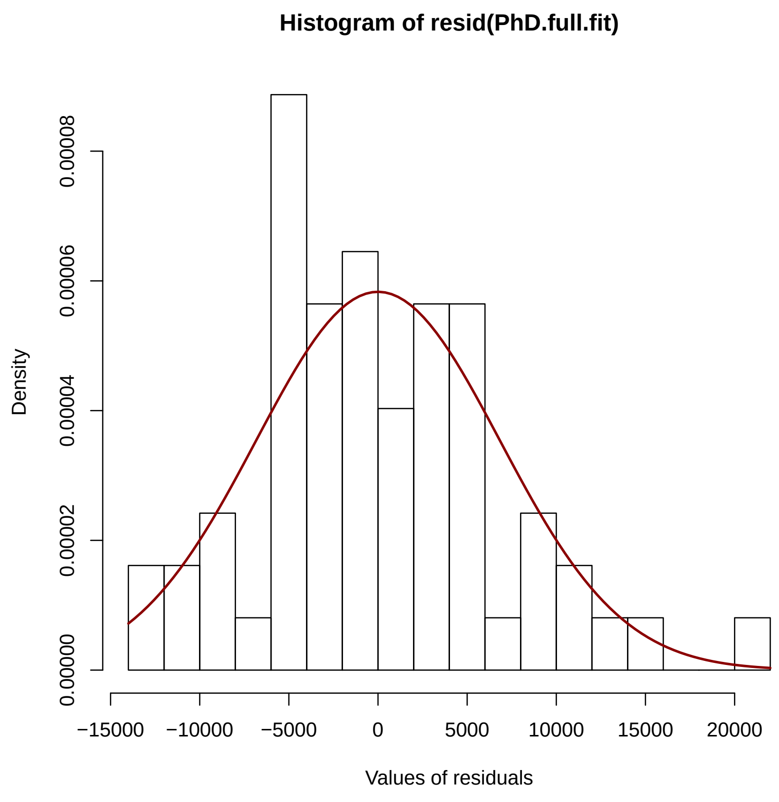


Figure 4.12: Histogram of residuals with normal curve overlay

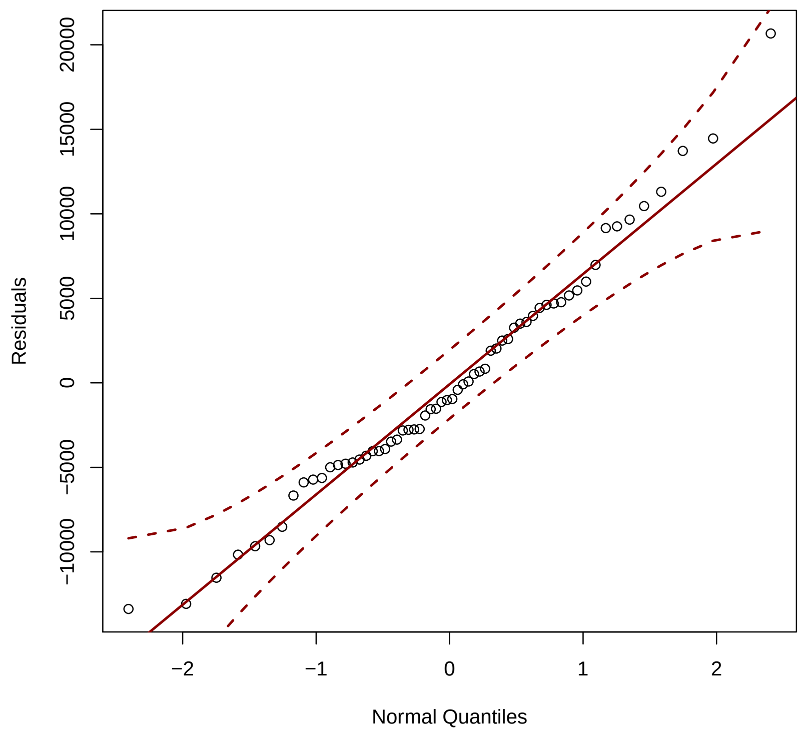


Figure 4.13: Normal q-q plot of residuals with a 95 confidence band

Chapter 7

Interactions among Continuous Variables

7.1 A Numerical Example

```
# Data for example in CCAW section 7.4.5
mod.data <- read.table("C07E01DT.txt", header=FALSE)
```

```
# Name the variables
colnames(mod.data)<-c("case","x","z","y")
# X = Age
# Y = Endurance
#  Z = Amount of exercise
head(mod.data)

##   case  x  z  y
## 1    1 60 10 18
## 2    2 40  9 36
## 3    3 29  2 51
## 4    4 47 10 18
## 5    5 48  9 23
## 6    6 42  6 30
```

In order to perform regression with interactions, we need to mean center the continuous predictors. This can be done two different ways.

1. Create new variables in the dataset
2. Use the `scale()` function within the `lm()` function, using the following arguments: `center = TRUE` and `scale = FALSE`.

To create the mean centered variables, just subtract the raw score from the mean

```
mod.data$x.c <- mod.data$x - mean(mod.data$x)
mod.data$z.c <- mod.data$z - mean(mod.data$z)
```

Likewise, creating the interaction terms can be done in two different ways.

1. Create new variables in the dataset
2. Use the `:` function within the `lm()` function (e.g., `lm(Y X+Y+X:Y)`).

To create the interaction from the *mean-centered* variables, just multiply the terms together.

```
mod.data$xz.c <- mod.data$x.c*mod.data$z.c
```

The summary statistics of centered data are

```
# Subset of variables of interest
mod.vars <- c("x.c","z.c","xz.c", "y")

# means
colMeans(mod.data[mod.vars])

##                  x.c                  z.c                 xz.c                    y
## -0.000000000000000870  0.000000000000000363 13.586505622657226056 26.530612244897959329

# SDs
sapply(mod.data[mod.vars],sd)

##   x.c   z.c  xz.c     y
## 10.11  4.78 46.01 10.82
```

To loose some of the zeros in the means calculations, use the `sprintf()` function.

```
sprintf("%.2f", colMeans(mod.data[mod.vars]))

## [1] "-0.00" "0.00"  "13.59" "26.53"
```

```
# Correlations
cor(mod.data[mod.vars])

##          x.c    z.c    xz.c      y
## x.c   1.0000  0.283 -0.0137 -0.126
## z.c   0.2827  1.000 -0.1179  0.337
## xz.c -0.0137 -0.118  1.0000  0.154
## y    -0.1259  0.337  0.1537  1.000
```

The full moderation model is

```
mod.fit <- lm(y ~ x.c + z.c + xz.c, data=mod.data)
summary(mod.fit)

##
## Call:
## lm(formula = y ~ x.c + z.c + xz.c, data = mod.data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -21.165  -6.939   0.269   6.299  21.299
##
## Coefficients:
##             Estimate Std. Error t value             Pr(>|t|)
## (Intercept)  25.8887     0.6466   40.04 < 0.0000000000000002 ***
## x.c          -0.2617     0.0641   -4.08       0.000060074745 ***
## z.c           0.9727     0.1365    7.12       0.000000000012 ***
## xz.c          0.0472     0.0136    3.48               0.0006 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.7 on 241 degrees of freedom
## Multiple R-squared:  0.206,Adjusted R-squared:  0.196
## F-statistic: 20.9 on 3 and 241 DF,  p-value: 0.00000000000476
```

```
# Equivalent ways of specifying the full moderation model
lm(y~x.c+z.c+x.c:z.c, data=mod.data)
lm(y~x.c*z.c, data=mod.data)
lm(y ~ scale(x, scale=FALSE) + scale(z, scale=FALSE) +
scale(x, scale=FALSE):scale(z, scale=FALSE), data=mod.data)
```

To get the covariance matrix for the regression coefficients use the `vcov()` function.[1]

```
vcov(mod.fit)

##             (Intercept)       x.c       z.c      xz.c
## (Intercept)    0.418115  0.000244 -0.002997 -0.002510
## x.c            0.000244  0.004104 -0.002476 -0.000018
## z.c           -0.002997 -0.002476  0.018641  0.000221
## xz.c          -0.002510 -0.000018  0.000221  0.000185
```

There is not an **R** package that gives the computes the simple slopes standard errors an CIs. Fortunately, this is calculated easily with only only one predictor and one moderator. The full regression equation is given in Equation 7.1.

$$Y = b_0 + b_1X + b_2Z + b_3XZ \tag{7.1}$$

Setting Z at an arbitrary value, Z_0, Equation 7.1 becomes

$$\begin{aligned} Y &= b_0 + b_1X + b_2Z_0 + b_3XZ_0 \\ &= (b_0 + b_2Z_0) + (b_1 + b_3Z_0)X \\ &= b_0^\dagger + b_1^\dagger X \end{aligned} \tag{7.2}$$

The standard error for $b_1^\dagger$ is

$$s_{b_1^\dagger} = \sqrt{s_{b_1}^2 + 2Z_0 s_{b_1b_3} + Z_0^2 s_{b_3}^2} \tag{7.3}$$

where
$s_{b_1}^2$ and $s_{b_3}^2$ are the variances of b_1 and b_3, respectively, from Equation 7.1, and
$s_{b_1b_3}$ is the covariance between b_1 and b_3.

```
# Values of z.c
z.low <- mean(mod.data$z.c) - sd(mod.data$z.c)
z.mean <- mean(mod.data$z.c)
z.high <- mean(mod.data$z.c) + sd(mod.data$z.c)

# For z.c=z.low

# Intercept
b0.zLow <- coef(mod.fit)["(Intercept)"]+coef(mod.fit)["z.c"]*z.low
b0.zLow

## (Intercept)
##        21.2

# Slope
b1.zLow <- coef(mod.fit)["x.c"]+coef(mod.fit)["xz.c"]*z.low
b1.zLow

##    x.c
## -0.487
```

[1]Note. These are **not** the covariances of the variables. They are the covariances of the *coefficients*.

```
# Standard Error
b1SE.zLow <- sqrt(vcov(mod.fit)["x.c","x.c"] + 2*z.low*vcov(mod.fit)["x.c","xz.c"] + z.low^2*vcov(mod.fit)["xz.
b1SE.zLow

## [1] 0.0921

# t-value
t.zLow <- b1.zLow/b1SE.zLow
t.zLow

##   x.c
## -5.29

# 95% CI, df=241, which I got from summary(mod.fit)
b1.zLowCIL <- b1.zLow - qt(.975,241)*b1SE.zLow
b1.zLowCIU <- b1.zLow + qt(.975,241)*b1SE.zLow
cbind(Lower=b1.zLowCIL,Upper=b1.zLowCIU)

##      Lower  Upper
## x.c -0.669 -0.306
```

If we set X at an arbitrary value, X_0, Equation 7.2 becomes

$$\begin{aligned} Y &= b_0 + b_1X_0 + b_2 + b_3X_0Z \\ &= (b_0 + b_1X_0) + (b_2 + b_3X_0)Z \\ &= b_0^{\ddagger} + b_1^{\ddagger}Z \end{aligned} \tag{7.4}$$

The standard error for $b_1^{\ddagger}$ is

$$s_{b_1^{\ddagger}} = \sqrt{s_{b_2}^2 + 2Z_0 s_{b_2b_3} + Z_0^2 s_{b_3}^2} \tag{7.5}$$

where
$s_{b_2}^2$ and $s_{b_3}^2$ are the variances of b_2 and b_3, respectively, from Equation 7.1, and $s_{b_2b_3}$ is the covariance between b_2 and b_3.

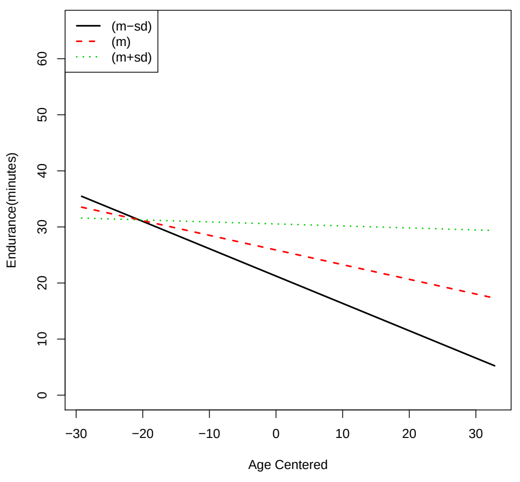
```
# Values of x.c
x.low <- mean(mod.data$x.c) - sd(mod.data$x.c)
x.mean <- mean(mod.data$x.c)
x.high <- mean(mod.data$x.c) + sd(mod.data$x.c)

# For x.c=x.low

# Intercept
b0.xLow <- coef(mod.fit)["(Intercept)"]+coef(mod.fit)["x.c"]*x.low
b0.xLow

## (Intercept)
##        28.5

# Slope
b1.xLow <- coef(mod.fit)["z.c"]+coef(mod.fit)["xz.c"]*x.low
b1.xLow

##   z.c
## 0.495

# Standard Error
b1SE.xLow <- sqrt(vcov(mod.fit)["z.c","z.c"] + 2*x.low*vcov(mod.fit)["z.c","xz.c"] + x.low^2*vcov(mod.fit)["xz.
b1SE.xLow
```

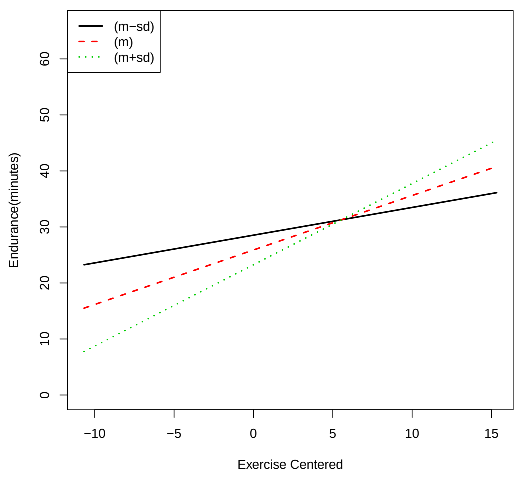
```
## [1] 0.182

# t-value
t.xLow <- b1.xLow/b1SE.xLow
t.xLow

##  z.c
## 2.72

# 95% CI, df=241, which I got from summary(mod.fit)
b1.xLowCIL <- b1.xLow - qt(.975,241)*b1SE.xLow
b1.xLowCIU <- b1.xLow + qt(.975,241)*b1SE.xLow
cbind(Lower=b1.xLowCIL,Upper=b1.xLowCIU)

##     Lower Upper
## z.c 0.137 0.853

dim(mod.data)

## [1] 245   7
```

There are multiple functions in **R** that will plot interactions. We will just show the one from the `rockchalk` package.

In the `rockchalk` package, use the `plotSlopes()` function. In the `plotSlopes()` function, specify the predictor using the `plotx` argument, specify the moderator using the `modx` argument, and specify the levels of a continuous moderator using the `modxVals` argument. To hide the scatterplot values, change the default setting of the `plotPoints` argument to `FALSE`.

```
library(rockchalk)
# Must fit a lm() model creating the interaction in the funciton
mod.fit2 <- lm(y ~ x.c*z.c, data=mod.data)
# Age as predictor
plotSlopes(mod.fit2, plotx="x.c", modx="z.c", modxVals="std.dev.", xlab="Age Centered",
ylab="Endurance(minutes)", plotPoints=FALSE)
# Exercise as predictor
plotSlopes(mod.fit2, plotx="z.c", modx="x.c", modxVals="std.dev.", xlab="Exercise Centered",
ylab="Endurance(minutes)", plotPoints=FALSE)
```





7.2 Standardized Estimates for Equations Containing Interactions

To create standardized coefficients, either transform all the continuous variables to a Z-scale or use the `scale()` function on each continuous predictor and the outcome in the `lm()` function.

```
mod.data$y.std <- scale(mod.data$y)
mod.data$x.std <- scale(mod.data$x)
mod.data$z.std <- scale(mod.data$z)
mod.data$xz.std <- mod.data$x.std*mod.data$z.std
```

```
modSt.fit<-lm(y.std~x.std + z.std + xz.std, data=mod.data)
# This is equivalnet to the above regression function
modSt.fit<-lm(scale(y)~scale(x)*scale(z), data=mod.data)
summary(modSt.fit)
```

```
##
## Call:
## lm(formula = y.std ~ x.std + z.std + xz.std, data = mod.data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.9563 -0.6414  0.0249  0.5823  1.9687
##
## Coefficients:
##             Estimate Std. Error t value       Pr(>|t|)
## (Intercept)  -0.0593     0.0598   -0.99         0.3219
## x.std        -0.2445     0.0598   -4.08 0.000060074745 ***
## z.std         0.4293     0.0603    7.12 0.000000000012 ***
## xz.std        0.2108     0.0606    3.48         0.0006 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.897 on 241 degrees of freedom
## Multiple R-squared:  0.206,Adjusted R-squared:  0.196
## F-statistic: 20.9 on 3 and 241 DF,  p-value: 0.00000000000476
```

The summary statistics of standardized data are

```
# Subset of variables of interest
modSt.vars <- c("x.std","z.std","xz.std", "y.std")
# means
sprintf("%.2f", colMeans(mod.data[modSt.vars]))

## [1] "-0.00" "0.00"  "0.28"  "-0.00"

# SDs
apply(mod.data[modSt.vars],2,sd)

##  x.std  z.std xz.std  y.std
##  1.000  1.000  0.953  1.000
```

```
# Correlations
cor(mod.data[modSt.vars])

##          x.std  z.std  xz.std  y.std
## x.std   1.0000  0.283 -0.0137 -0.126
## z.std   0.2827  1.000 -0.1179  0.337
## xz.std -0.0137 -0.118  1.0000  0.154
## y.std  -0.1259  0.337  0.1537  1.000
```

7.3 Curvilinear by Linear Interactions

```
# Data for example in CCAW section 7.9
curv.data <- read.table("C07E02DT.txt", header=FALSE)
```

To

```
# Name the variables
colnames(curv.data)<-c("case","x","z","y")
```

Using polynomial terms in the regression equations can be done two different ways.

1. Create new variables in the dataset
2. Use the `I()` function within the `lm()` function.

```
# Mean center, square, and create interactions for the variables
curv.data$x.c  <- curv.data$x - mean(curv.data$x)
curv.data$z.c  <- curv.data$z - mean(curv.data$z)
curv.data$x2.c  <- curv.data$x.c^2
curv.data$xz.c  <- curv.data$x.c*curv.data$z.c
curv.data$x2z.c  <- curv.data$x2.c*curv.data$z.c
```

Once the terms are created, estimating the curvilinear regressions with and without interactions are just like any other regression in **R**.

```
# Centered curvilinear regression with no interaction effect
curv.fit1 <- lm(y ~ x.c + x2.c + z.c, data=curv.data)
# This is equivalnet to the above equation
curv.fit1 <- lm(y ~ scale(x, scale=FALSE) + I(scale(x, scale=FALSE)^2) + scale(z, scale=FALSE), data=curv.data)
summary(curv.fit1)
```

```
##
## Call:
## lm(formula = y ~ x.c + x2.c + z.c, data = curv.data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.4260 -0.4388 -0.0316  0.3979  1.8519
##
## Coefficients:
##             Estimate Std. Error t value             Pr(>|t|)
## (Intercept)  3.64213    0.04746   76.74 < 0.0000000000000002 ***
## x.c          0.22362    0.03375    6.63        0.00000000022 ***
## x2.c        -0.00813    0.01908   -0.43                 0.67
## z.c          0.61984    0.05292   11.71 < 0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.605 on 246 degrees of freedom
## Multiple R-squared:  0.511,Adjusted R-squared:  0.505
## F-statistic: 85.6 on 3 and 246 DF,  p-value: <0.0000000000000002
```

```
# Centered regression with curvilinear and linear interaction
curv.fit2 <- lm(y ~ x.c + x2.c + z.c + xz.c + x2z.c, data=curv.data)
curv.fit2 <- lm(y ~ scale(x, scale=FALSE) + I(scale(x, scale=FALSE)^2) + scale(z, scale=FALSE) + scale(x, scale
summary(curv.fit2)
```

```
##
## Call:
## lm(formula = y ~ scale(x, scale = FALSE) + I(scale(x, scale = FALSE)^2) +
##     scale(z, scale = FALSE) + scale(x, scale = FALSE):scale(z,
##     scale = FALSE) + I(scale(x, scale = FALSE)^2):scale(z, scale = FALSE),
##     data = curv.data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.4804 -0.4188 -0.0626  0.3842  1.6273
##
## Coefficients:
##                                                        Estimate Std. Error t value
## (Intercept)                                              3.6511     0.0462   79.09
## scale(x, scale = FALSE)                                  0.1783     0.0377    4.73
## I(scale(x, scale = FALSE)^2)                            -0.0520     0.0216   -2.41
## scale(z, scale = FALSE)                                  0.5509     0.0624    8.83
## scale(x, scale = FALSE):scale(z, scale = FALSE)          0.1641     0.0472    3.48
## I(scale(x, scale = FALSE)^2):scale(z, scale = FALSE)    0.0648     0.0287    2.26
##                                                                    Pr(>|t|)
## (Intercept)                                          < 0.0000000000000002 ***
## scale(x, scale = FALSE)                                         0.0000038 ***
## I(scale(x, scale = FALSE)^2)                                      0.01691 *
## scale(z, scale = FALSE)                              < 0.0000000000000002 ***
## scale(x, scale = FALSE):scale(z, scale = FALSE)                   0.00059 ***
## I(scale(x, scale = FALSE)^2):scale(z, scale = FALSE)              0.02470 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.588 on 244 degrees of freedom
## Multiple R-squared:  0.542,Adjusted R-squared:  0.532
## F-statistic: 57.7 on 5 and 244 DF,  p-value: <0.0000000000000002
```

```
##
## Call:
## lm(formula = y ~ x.c + x2.c + z.c + xz.c + x2z.c, data = curv.data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.4804 -0.4188 -0.0626  0.3842  1.6273
##
## Coefficients:
##             Estimate Std. Error t value             Pr(>|t|)
## (Intercept)   3.6511     0.0462   79.09 < 0.0000000000000002 ***
## x.c           0.1783     0.0377    4.73            0.0000038 ***
## x2.c         -0.0520     0.0216   -2.41              0.01691 *
## z.c           0.5509     0.0624    8.83 < 0.0000000000000002 ***
## xz.c          0.1641     0.0472    3.48              0.00059 ***
## x2z.c         0.0648     0.0287    2.26              0.02470 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.588 on 244 degrees of freedom
## Multiple R-squared:  0.542,Adjusted R-squared:  0.532
## F-statistic: 57.7 on 5 and 244 DF,  p-value: <0.0000000000000002
```

```
# Values of z.c
z.low <- mean(curv.data$z.c) -  sd(curv.data$z.c)
z.mean <- mean(curv.data$z.c)
z.high <- mean(curv.data$z.c) +  sd(curv.data$z.c)
```

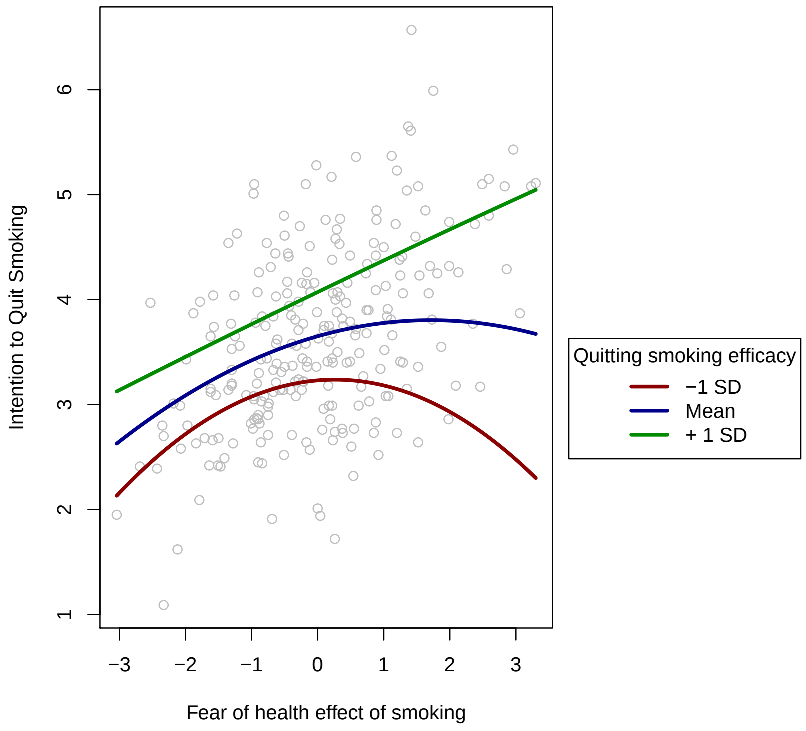


Figure 7.1: Curvilinear by linear interaction for the quitting smoking data.

```
# For z.c=z.low

# Intercept
b0.zLow <- coef(curv.fit2)["(Intercept)"]+coef(curv.fit2)["z.c"]*z.low
b0.zLow
# Slope for x
b1.xLow <- coef(curv.fit2)["x.c"]+coef(curv.fit2)["xz.c"]*z.low
b1.xLow
# Slope for x2
b2.xLow <- coef(curv.fit2)["x2.c"]+coef(curv.fit2)["x2z.c"]*z.low
b2.xLow
```

Plotting the interaction is a little trickier than in the case where there is no curvilinear relationship. We will use the native `curve()` function to do this. The result is shown in Figure 7.1.

```
# Simple  Regressions
par(mar=par()$mar+c(0,0,0,10))
plot(curv.data$x.c, curv.data$y, ylab="Intention to Quit Smoking",
xlab="Fear of health effect of smoking", col="gray")
curve (cbind (1,x, x*x, z.low, x*z.low,x*x*z.low) %*% coef(curv.fit2), add=TRUE, col="red4", lwd=3)
curve (cbind (1,x, x*x, z.mean, x*z.mean,x*x*z.mean) %*% coef(curv.fit2), add=TRUE, col="blue4",
lwd=3)
curve (cbind (1,x, x*x, z.high, x*z.high,x*x*z.high) %*% coef(curv.fit2), add=TRUE, col="green4",
lwd=3)
legend(max(curv.data$x.c)+.5, mean(curv.data$y),c("-1 SD", "Mean", "+ 1 SD"),
title="Quitting smoking efficacy", lwd = 3, col=c("red4", "blue4", "green4"),
xpd=TRUE)
```

Chapter 8

Categorical or Nominal Independent Variables

8.1 Dummy-Variable Coding

First, import the abortion attitude data shown in Table 8.2.3 in CCAW. Since the dataset included with the book put the case number and group into a single variable, we need to do some extra work while importing data into **R**. Instead of using the `read.table()` function, we can use `read.fwf()` function to import data with fixed widths.

```
#Religion data (Table 8.2.3)
Abortion.data <- read.fwf("C08e01dt.txt", widths=c(3,1,5))
```

The dataset did not include variable names, we can add the names on top of each column by using the `colnames()` function.

```
colnames(Abortion.data)<-c("Case", "Group", "ATA")
head(Abortion.data)

##   Case Group ATA
## 1    1     c  61
## 2    2     o  78
## 3    3     p  47
## 4    4     c  65
## 5    5     c  45
## 6    6     o 106
```

In order to do the data analyses, the categorical predictor (religious groups) needs to be dummy coded. There are multiple ways to do dummy coding in **R**, but the easiest is by using the `contr.treatment()` function. It takes as its main arguments the number of groups and the group you want to be the reference group. By default, **R** using the first group as the comparison group (cf. Table 8.2.1.A in CCAW).

```
contrasts(Abortion.data$Group)

##   j o p
## c 0 0 0
## j 1 0 0
## o 0 1 0
## p 0 0 1
```

We can make the last group (in this case Protestants) the reference group (cf. Table 8.2.1.D and Table 8.2.3 in CCAW)

```
# Make the fourth group the comparison group
contrasts(Abortion.data$Group) <- contr.treatment(4, base = 4, contrasts=TRUE)
contrasts(Abortion.data$Group)

##   1 2 3
## c 1 0 0
## j 0 1 0
## o 0 0 1
## p 0 0 0
```

A more difficult, but sometimes necessary, way of creating the dummy codes is doing it by hand. That is, creating each dummy code manually. This is most easily done by using the **ifelse()** function.

```
# Catholic
Abortion.data$C1<-ifelse(Abortion.data$Group=="c", 1, 0)
# Jewish
Abortion.data$C2<-ifelse(Abortion.data$Group=="j", 1, 0)
# Other
Abortion.data$C3<-ifelse(Abortion.data$Group=="o", 1, 0)
head(Abortion.data)

##   Case Group ATA C1 C2 C3
## 1    1     c  61  1  0  0
## 2    2     o  78  0  0  1
## 3    3     p  47  0  0  0
## 4    4     c  65  1  0  0
## 5    5     c  45  1  0  0
## 6    6     o 106  0  0  1
```

After creating the dummy coded variables, then we can use the **cor()** and **colMeans()** functions to estimate the correlations and means given in Table 8.2.4. To estimate the SD, we have to use the **sapply()** and **sd()** functions. To get the multiple R^2, we have to use the **lm()** function, which we will do later.

```
# Create variables-of-interest set
Abortion.vars <- c("ATA", "C1", "C2", "C3")
cor(Abortion.data[Abortion.vars])

##         ATA     C1     C2     C3
## ATA  1.000 -0.442  0.355 -0.225
## C1  -0.442  1.000 -0.258 -0.309
## C2   0.355 -0.258  1.000 -0.239
## C3  -0.225 -0.309 -0.239  1.000

colMeans(Abortion.data[Abortion.vars])

##    ATA     C1     C2     C3
## 81.694  0.250  0.167  0.222

sapply(Abortion.data[Abortion.vars], sd)

##    ATA     C1     C2     C3
## 27.880  0.439  0.378  0.422
```

To put all the results in a single table, use the **rbind()** function.

```
Abortion.descTable <- rbind(cor(Abortion.data[Abortion.vars]), colMeans(Abortion.data[Abortion.vars]), sapply(A
rownames(Abortion.descTable) <- c("ATA", "C1", "C2", "C3", "Mean", "SD")
Abortion.descTable
```

```
##         ATA     C1     C2     C3
## ATA   1.000 -0.442  0.355 -0.225
## C1   -0.442  1.000 -0.258 -0.309
## C2    0.355 -0.258  1.000 -0.239
## C3   -0.225 -0.309 -0.239  1.000
## Mean 81.694  0.250  0.167  0.222
## SD   27.880  0.439  0.378  0.422
```

To print a pretty table in LaTeX, use the `xtable()` function in the `xtable` package.

```
library(xtable)
xtable(Abortion.descTable,
caption="Correlations, Means, and Standard Deviations from CCAW (p. 307)")
```

Table 8.1: Correlations, Means, and Standard Deviations from CCAW (p. 307)

	ATA	C1	C2	C3
ATA	1.00	-0.44	0.35	-0.22
C1	-0.44	1.00	-0.26	-0.31
C2	0.35	-0.26	1.00	-0.24
C3	-0.22	-0.31	-0.24	1.00
Mean	81.69	0.25	0.17	0.22
SD	27.88	0.44	0.38	0.42

The partial and semi-partial correlations are calculated via the `ppcor()` function in the `ppcor` package.

```
library(ppcor)
# partial
pcor(Abortion.data[Abortion.vars])

## $estimate
##        ATA     C1     C2     C3
## ATA  1.000 -0.494  0.154 -0.363
## C1  -0.494  1.000 -0.233 -0.499
## C2   0.154 -0.233  1.000 -0.263
## C3  -0.363 -0.499 -0.263  1.000
##
## $p.value
##         ATA      C1    C2      C3
## ATA 0.00000 0.00298 0.384 0.03491
## C1  0.00298 0.00000 0.185 0.00266
## C2  0.38444 0.18543 0.000 0.13221
## C3  0.03491 0.00266 0.132 0.00000
##
## $statistic
##        ATA    C1     C2    C3
## ATA  0.000 -3.21  0.882 -2.20
## C1  -3.214  0.00 -1.353 -3.26
## C2   0.882 -1.35  0.000 -1.54
## C3  -2.203 -3.26 -1.545  0.00
##
## $n
## [1] 36
##
## $gp
## [1] 2
```

```
##
## $method
## [1] "pearson"

# semi-partial/part
spcor(Abortion.data[Abortion.vars])

## $estimate
##        ATA     C1     C2     C3
## ATA  1.000 -0.456  0.125 -0.313
## C1  -0.439  1.000 -0.185 -0.445
## C2   0.140 -0.214  1.000 -0.245
## C3  -0.324 -0.479 -0.227  1.000
##
## $p.value
##         ATA      C1    C2      C3
## ATA 0.00000 0.00667 0.480 0.07169
## C1  0.00948 0.00000 0.296 0.00844
## C2  0.43114 0.22383 0.000 0.16343
## C3  0.06178 0.00417 0.197 0.00000
##
## $statistic
##        ATA    C1     C2    C3
## ATA  0.000 -2.90  0.714 -1.86
## C1  -2.760  0.00 -1.063 -2.81
## C2   0.797 -1.24  0.000 -1.43
## C3  -1.936 -3.09 -1.319  0.00
##
## $n
## [1] 36
##
## $gp
## [1] 2
##
## $method
## [1] "pearson"
```

The regression coefficients and their standard error are calculated as with a typical regression.

```
# Regression coefficinets
Abortion.fit <- lm(ATA ~ C1 + C2 + C3, data=Abortion.data)
summary(Abortion.fit)

##
## Call:
## lm(formula = ATA ~ C1 + C2 + C3, data = Abortion.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -46.31 -13.90  -3.99  18.42  46.69
##
## Coefficients:
##             Estimate Std. Error t value           Pr(>|t|)
## (Intercept)     93.3        6.5   14.37 0.0000000000000017 ***
## C1             -32.6       10.2   -3.21              0.003 **
## C2              10.2       11.6    0.88              0.384
## C3             -23.2       10.5   -2.20              0.035 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 23.4 on 32 degrees of freedom
## Multiple R-squared:  0.355,Adjusted R-squared:  0.294
## F-statistic: 5.87 on 3 and 32 DF,  p-value: 0.0026

# Standardized Regression coefficinets, only standardizing the continuous variables
Abortion.fitStd <- lm(scale(ATA) ~ C1 + C2 + C3, data=Abortion.data)
summary(Abortion.fitStd)

##
## Call:
## lm(formula = scale(ATA) ~ C1 + C2 + C3, data = Abortion.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -1.661 -0.498 -0.143  0.661  1.675
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.417      0.233    1.79    0.083 .
## C1            -1.171      0.364   -3.21    0.003 **
## C2             0.366      0.415    0.88    0.384
## C3            -0.832      0.377   -2.20    0.035 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.84 on 32 degrees of freedom
## Multiple R-squared:  0.355,Adjusted R-squared:  0.294
## F-statistic: 5.87 on 3 and 32 DF,  p-value: 0.0026

# Standardized Regression coefficinets uisng CCAW's method
Abortion.fitStd2 <- lm(scale(ATA) ~ scale(C1) + scale(C2) + scale(C3), data=Abortion.data)
summary(Abortion.fitStd2)

##
## Call:
## lm(formula = scale(ATA) ~ scale(C1) + scale(C2) + scale(C3),
##     data = Abortion.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -1.661 -0.498 -0.143  0.661  1.675
##
## Coefficients:
##                           Estimate             Std. Error t value Pr(>|t|)
## (Intercept)  0.0000000000000000651  0.1399929900965807272    0.00    1.000
## scale(C1)   -0.5141457123833444998  0.1599535244454814986   -3.21    0.003 **
## scale(C2)    0.1381745514846636214  0.1566888233313864232    0.88    0.384
## scale(C3)   -0.3505961682397592538  0.1591436279784460839   -2.20    0.035 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.84 on 32 degrees of freedom
## Multiple R-squared:  0.355,Adjusted R-squared:  0.294
## F-statistic: 5.87 on 3 and 32 DF,  p-value: 0.0026
```

To obtain the predicted values, use the `predict()` function using only values of interest for the predictors

```
# Catholic
Cath.data <- data.frame("C1"=1, "C2"=0, "C3"=0)
predict(Abortion.fit,newdata=Cath.data)

##    1
## 60.7

# Protestant
Pros.data <- data.frame("C1"=0, "C2"=0, "C3"=0)
predict(Abortion.fit,newdata=Pros.data)

##    1
## 93.3

# Jewish
Jew.data <- data.frame("C1"=0, "C2"=1, "C3"=0)
predict(Abortion.fit,newdata=Jew.data)

##   1
## 104

# Other
Other.data <- data.frame("C1"=0, "C2"=0, "C3"=1)
predict(Abortion.fit,newdata=Other.data)

##    1
## 70.1
```

The residual error can be obtained from the regression results.

```
summary(Abortion.fit)$sigma^2

## [1] 548
```

To test the “significance” of the R^2 examine the bottom of the `summary()` output.

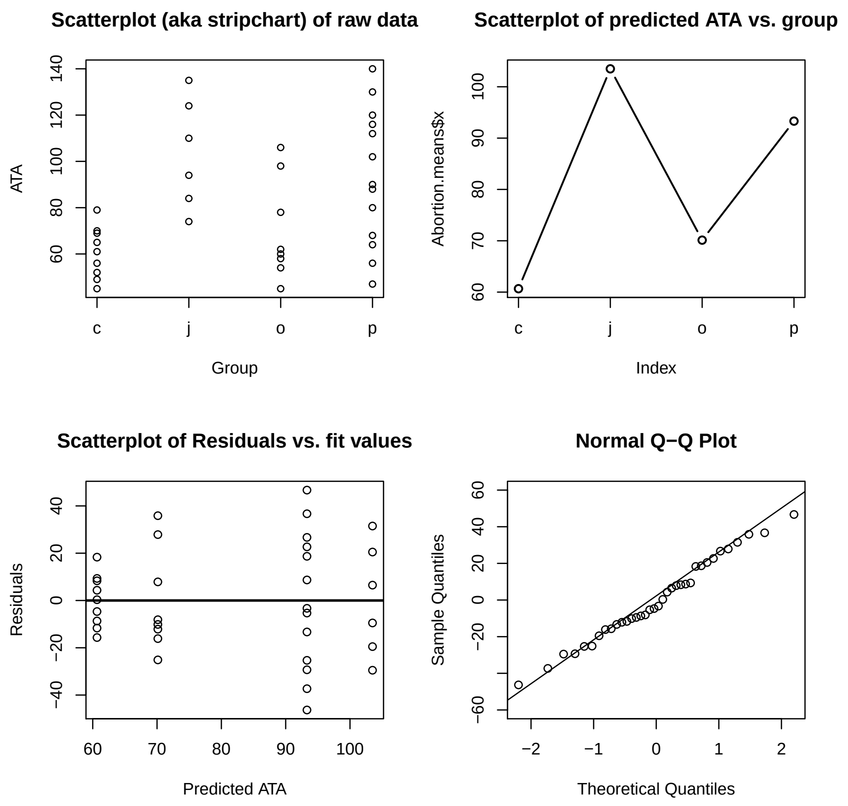
Use the `CI.Rsqlm()` function in the `psychometric` package to get the standard error and confidence interval for the R^2

```
library(psychometric)
CI.Rsqlm(Abortion.fit)

##     Rsq SErsq  LCL   UCL
## 1 0.355 0.109 0.14 0.569
```

Note. some editions of CCAW have the value of 0.136 as the SE of R^2. This is wrong, as they do not square the second term in their equation.

```
par(mfrow=c(2,2))
# Scatterplot 1
stripchart(ATA~Group,data = Abortion.data, vertical = TRUE,pch = 21, xlab="Group", main="Scatterplot (aka strip

# Scatterplot 2
# Calculate group means
Abortion.means <- aggregate(Abortion.data$ATA, list(Group=Abortion.data$Group), mean)

plot(Abortion.means$x, type="b", xaxt="n", main="Scatterplot of predicted ATA vs. group")
axis(1, at=1:4, labels=levels(Abortion.means$Group))
lines(Abortion.means$Group, Abortion.means$x, type="b", lwd=1.5)


# Scatterplot 3
plot(predict(Abortion.fit), summary(Abortion.fit)$residuals, xlab="Predicted ATA", ylab="Residuals", main="Scat
# Horizontal line
abline(h=0, lwd=2)
```

```r
# Scatterplot 4
qqnorm(summary(Abortion.fit)$residuals, ylim=c(-60,60))
qqline(summary(Abortion.fit)$residuals)
```



To obtain the confidence intervals, use the `confint()` function.

```r
# Unstandardized
confint(Abortion.fit,level = 0.95)

##               2.5 % 97.5 %
## (Intercept)  80.1 106.54
## C1          -53.3 -11.96
## C2          -13.4  33.74
## C3          -44.6  -1.75
```

ANOVA table for regression model

```r
anova(Abortion.fit)

## Analysis of Variance Table
##
## Response: ATA
##           Df Sum Sq Mean Sq F value Pr(>F)
## C1         1   5306    5306    9.68 0.0039 **
## C2         1   1689    1689    3.08 0.0889 .
## C3         1   2662    2662    4.85 0.0349 *
## Residuals 32  17549     548
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

8.2 Effects Coding

The easiest way to do effects coding is to use the `contr.sum()` function.

```
 #Effect coding of the group differences
contrasts(Abortion.data$Group)<-contr.sum
contrasts(Abortion.data$Group)

##   [,1] [,2] [,3]
## c    1    0    0
## j    0    1    0
## o    0    0    1
## p   -1   -1   -1
```

A more difficult, but sometimes necessary, way of creating the effects codes is doing it by hand. That is, creating each effects code manually. This is most easily done by using the `ifelse()` function.

```
# Catholic
Abortion.data$E1<-ifelse(Abortion.data$Group=="c", 1, ifelse(Abortion.data$Group=="p", -1,0))
# Jewish
Abortion.data$E2<-ifelse(Abortion.data$Group=="j", 1, ifelse(Abortion.data$Group=="p", -1,0))
# Other
Abortion.data$E3<-ifelse(Abortion.data$Group=="o", 1, ifelse(Abortion.data$Group=="p", -1,0))
head(Abortion.data)

##   Case Group ATA C1 C2 C3 E1 E2 E3
## 1    1     c  61  1  0  0  1  0  0
## 2    2     o  78  0  0  1  0  0  1
## 3    3     p  47  0  0  0 -1 -1 -1
## 4    4     c  65  1  0  0  1  0  0
## 5    5     c  45  1  0  0  1  0  0
## 6    6     o 106  0  0  1  0  0  1
```

After creating the effects coded variables, then we can use the `cor()` and `colMeans()` functions to estimate the correlations and means given in Table 8.3.3. To estimate the SD, we have to use the `sapply()` and `sd()` functions. To get the multiple R^2, we have to use the `lm()` function, which we will do later.

```
# Create variables-of-interest set
Abortion.vars2 <- c("ATA", "E1", "E2", "E3")
cor(Abortion.data[Abortion.vars2])

##         ATA     E1      E2     E3
## ATA  1.0000 -0.444 -0.0291 -0.328
## E1  -0.4443  1.000  0.6268  0.595
## E2  -0.0291  0.627  1.0000  0.636
## E3  -0.3277  0.595  0.6355  1.000

colMeans(Abortion.data[Abortion.vars2])

##    ATA     E1     E2     E3
## 81.694 -0.111 -0.194 -0.139

sapply(Abortion.data[Abortion.vars2], sd)

##    ATA     E1     E2     E3
## 27.880  0.785  0.710  0.762
```

To put all the results in a single table, use the `rbind()` function.

```
Abortion.descTable2 <- rbind(cor(Abortion.data[Abortion.vars2]), colMeans(Abortion.data[Abortion.vars2]), sappl
rownames(Abortion.descTable2) <- c("ATA", "E1", "E2", "E3", "Mean", "SD")
Abortion.descTable2
```

```
##          ATA     E1      E2     E3
## ATA   1.0000 -0.444 -0.0291 -0.328
## E1   -0.4443  1.000  0.6268  0.595
## E2   -0.0291  0.627  1.0000  0.636
## E3   -0.3277  0.595  0.6355  1.000
## Mean 81.6944 -0.111 -0.1944 -0.139
## SD   27.8802  0.785  0.7099  0.762
```

To print a pretty table in LaTeX, use the `xtable()` function in the `xtable` package.

```
library(xtable)
xtable(Abortion.descTable2,
caption="Correlations, Means, and Standard Deviations from CCAW (p. 324)", digits=3)
```

Table 8.2: Correlations, Means, and Standard Deviations from CCAW (p. 324)

	ATA	E1	E2	E3
ATA	1.000	-0.444	-0.029	-0.328
E1	-0.444	1.000	0.627	0.595
E2	-0.029	0.627	1.000	0.636
E3	-0.328	0.595	0.636	1.000
Mean	81.694	-0.111	-0.194	-0.139
SD	27.880	0.785	0.710	0.762

The partial and semi-partial correlations are calculated via the `ppcor()` function in the `ppcor` package.

```
library(ppcor)
# partial
pcor(Abortion.data[Abortion.vars2])

## $estimate
##        ATA     E1    E2     E3
## ATA  1.000 -0.481 0.436 -0.281
## E1  -0.481  1.000 0.526  0.140
## E2   0.436  0.526 1.000  0.485
## E3  -0.281  0.140 0.485  1.000
##
## $p.value
##         ATA      E1      E2      E3
## ATA 0.00000 0.00402 0.00996 0.10803
## E1  0.00402 0.00000 0.00140 0.42950
## E2  0.00996 0.00140 0.00000 0.00368
## E3  0.10803 0.42950 0.00368 0.00000
##
## $statistic
##       ATA   E1   E2    E3
## ATA  0.00 -3.1 2.74 -1.65
## E1  -3.10  0.0 3.50  0.80
## E2   2.74  3.5 0.00  3.13
## E3  -1.65  0.8 3.13  0.00
##
## $n
## [1] 36
##
## $gp
## [1] 2
```

```
##
## $method
## [1] "pearson"

# semi-partial/part
spcor(Abortion.data[Abortion.vars2])

## $estimate
##        ATA      E1    E2      E3
## ATA  1.000 -0.4402 0.389 -0.2347
## E1  -0.354  1.0000 0.399  0.0914
## E2   0.308  0.3936 1.000  0.3527
## E3  -0.205  0.0991 0.388  1.0000
##
## $p.value
##       ATA      E1     E2     E3
## ATA 0.000 0.00919 0.0230 0.1814
## E1  0.040 0.00000 0.0193 0.6074
## E2  0.076 0.02129 0.0000 0.0408
## E3  0.245 0.57719 0.0233 0.0000
##
## $statistic
##       ATA     E1   E2     E3
## ATA  0.00 -2.773 2.39 -1.366
## E1  -2.14  0.000 2.46  0.519
## E2   1.83  2.422 0.00  2.132
## E3  -1.18  0.563 2.38  0.000
##
## $n
## [1] 36
##
## $gp
## [1] 2
##
## $method
## [1] "pearson"
```

The regression coefficients and their standard error are calculated as with a typical regression.

```
# Regression coefficinets
Abortion2.fit <- lm(ATA ~ E1 + E2 + E3, data=Abortion.data)
summary(Abortion2.fit)

##
## Call:
## lm(formula = ATA ~ E1 + E2 + E3, data = Abortion.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -46.31 -13.90  -3.99  18.42  46.69
##
## Coefficients:
##             Estimate Std. Error t value            Pr(>|t|)
## (Intercept)    81.90       4.05   20.20 <0.0000000000000002 ***
## E1            -21.23       6.85   -3.10               0.004 **
## E2             21.60       7.88    2.74               0.010 **
## E3            -11.77       7.12   -1.65               0.108
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 23.4 on 32 degrees of freedom
## Multiple R-squared:  0.355,Adjusted R-squared:  0.294
## F-statistic: 5.87 on 3 and 32 DF,  p-value: 0.0026

# Standardized Regression coefficinets, only standardizing the continuous variables
Abortion2.fitStd <- lm(scale(ATA) ~ E1 + E2 + E3, data=Abortion.data)
summary(Abortion2.fitStd)

##
## Call:
## lm(formula = scale(ATA) ~ E1 + E2 + E3, data = Abortion.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -1.661 -0.498 -0.143  0.661  1.675
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.00737    0.14544    0.05    0.960
## E1          -0.76159    0.24566   -3.10    0.004 **
## E2           0.77475    0.28275    2.74    0.010 **
## E3          -0.42234    0.25544   -1.65    0.108
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.84 on 32 degrees of freedom
## Multiple R-squared:  0.355,Adjusted R-squared:  0.294
## F-statistic: 5.87 on 3 and 32 DF,  p-value: 0.0026

# Standardized Regression coefficinets uisng CCAW's method
Abortion2.fitStd2 <- lm(scale(ATA) ~ scale(E1) + scale(E2) + scale(E3), data=Abortion.data)
summary(Abortion2.fitStd2)

##
## Call:
## lm(formula = scale(ATA) ~ scale(E1) + scale(E2) + scale(E3),
##     data = Abortion.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -1.661 -0.498 -0.143  0.661  1.675
##
## Coefficients:
##                          Estimate            Std. Error t value Pr(>|t|)
## (Intercept)  0.000000000000000281  0.139992990096580727    0.00    1.000
## scale(E1)   -0.597674938399394229  0.192787902059495392   -3.10    0.004 **
## scale(E2)    0.550000546408802138  0.200725309218371589    2.74    0.010 **
## scale(E3)   -0.321686605933545589  0.194561957462719731   -1.65    0.108
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.84 on 32 degrees of freedom
## Multiple R-squared:  0.355,Adjusted R-squared:  0.294
## F-statistic: 5.87 on 3 and 32 DF,  p-value: 0.0026
```

To obtain the predicted values, use the `predict()` function using only values of interest for the predictors

```
# Catholic
Cath.data2 <- data.frame("E1"=1, "E2"=0, "E3"=0)
predict(Abortion2.fit,newdata=Cath.data2)

##    1
## 60.7

# Protestant
Pros.data2 <- data.frame("E1"=-1, "E2"=-1, "E3"=-1)
predict(Abortion2.fit,newdata=Pros.data2)

##    1
## 93.3

# Jewish
Jew.data2 <- data.frame("E1"=0, "E2"=1, "E3"=0)
predict(Abortion2.fit,newdata=Jew.data2)

##   1
## 104

# Other
Other.data2 <- data.frame("E1"=0, "E2"=0, "E3"=1)
predict(Abortion2.fit,newdata=Other.data2)

##    1
## 70.1
```

The residual error can be obtained from the regression results

```
summary(Abortion2.fit)$sigma^2

## [1] 548
```

To obtain the confidence intervals, use the `confint()` function.

```
# Unstandardized
confint(Abortion2.fit,level = 0.95)

##              2.5 % 97.5 %
## (Intercept)  73.64  90.16
## E1          -35.18  -7.28
## E2            5.54  37.66
## E3          -26.28   2.73
```

8.2.1 Weighted Effects Coding

Weighted effects coding is similar to effects coding, but accounts for differences in group sample sizes. It has to be done manually in **R**.[1]

```
# Sample size for each group
Abortion.group.n <- unlist(lapply(split(Abortion.data$Group,f=Abortion.data$Group),length))

# Sample size for Catholic
Abortion.c.n <- Abortion.group.n["c"]

# Sample size for Protestants
Abortion.p.n <- Abortion.group.n["p"]

# Sample size for Jews
Abortion.j.n <- Abortion.group.n["j"]
```

[1]You *can* do it via the `contrasts()` function, but you still have to manually figure out the proportions. Consequently, it is easier to create the weighted effects variables manually.

```
# Sample size for Others
Abortion.o.n <- Abortion.group.n["o"]

# Weighted Effects
# Catholic
Abortion.data$WE1<-ifelse(Abortion.data$Group=="c", 1, ifelse(Abortion.data$Group=="p",
 -Abortion.c.n/Abortion.p.n,0))
# Jewish
Abortion.data$WE2<-ifelse(Abortion.data$Group=="j", 1, ifelse(Abortion.data$Group=="p",
 -Abortion.j.n/Abortion.p.n,0))
# Other
Abortion.data$WE3<-ifelse(Abortion.data$Group=="o", 1, ifelse(Abortion.data$Group=="p",
 -Abortion.o.n/Abortion.p.n,0))
head(Abortion.data)

##   Case Group ATA C1 C2 C3 E1 E2 E3    WE1    WE2    WE3
## 1    1     c  61  1  0  0  1  0  0  1.000  0.000  0.000
## 2    2     o  78  0  0  1  0  0  1  0.000  0.000  1.000
## 3    3     p  47  0  0  0 -1 -1 -1 -0.692 -0.462 -0.615
## 4    4     c  65  1  0  0  1  0  0  1.000  0.000  0.000
## 5    5     c  45  1  0  0  1  0  0  1.000  0.000  0.000
## 6    6     o 106  0  0  1  0  0  1  0.000  0.000  1.000
```

The regression coefficients and their standard error are calculated as with a typical regression.

```
# Regression coefficinets
Abortion3.fit <- lm(ATA ~ WE1 + WE2 + WE3, data=Abortion.data)
summary(Abortion3.fit)

##
## Call:
## lm(formula = ATA ~ WE1 + WE2 + WE3, data = Abortion.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -46.31 -13.90  -3.99  18.42  46.69
##
## Coefficients:
##             Estimate Std. Error t value            Pr(>|t|)
## (Intercept)    81.69       3.90   20.93 <0.0000000000000002 ***
## WE1           -21.03       6.76   -3.11              0.0039 **
## WE2            21.81       8.73    2.50              0.0178 *
## WE3           -11.57       7.30   -1.58              0.1229
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 23.4 on 32 degrees of freedom
## Multiple R-squared:  0.355,Adjusted R-squared:  0.294
## F-statistic: 5.87 on 3 and 32 DF,  p-value: 0.0026

# Standardized Regression coefficinets, only standardizing the continuous variables
Abortion3.fitStd <- lm(scale(ATA) ~ WE1 + WE2 + WE3, data=Abortion.data)
summary(Abortion3.fitStd)

##
## Call:
## lm(formula = scale(ATA) ~ WE1 + WE2 + WE3, data = Abortion.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -1.661 -0.498 -0.143  0.661  1.675
```

```
##
## Coefficients:
##                           Estimate             Std. Error t value Pr(>|t|)
## (Intercept)  0.0000000000000000534  0.1399929900965807272    0.00   1.0000
## WE1         -0.7542198119170729909  0.2424749715507644454   -3.11   0.0039 **
## WE2          0.7821169780117606107  0.3130338422294093870    2.50   0.0178 *
## WE3         -0.4149703456584693528  0.2619029027457212555   -1.58   0.1229
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.84 on 32 degrees of freedom
## Multiple R-squared:  0.355,Adjusted R-squared:  0.294
## F-statistic: 5.87 on 3 and 32 DF,  p-value: 0.0026

# Standardized Regression coefficinets uisng CCAW's method
Abortion3.fitStd2 <- lm(scale(ATA) ~ scale(WE1) + scale(WE2) + scale(WE3), data=Abortion.data)
summary(Abortion3.fitStd2)

##
## Call:
## lm(formula = scale(ATA) ~ scale(WE1) + scale(WE2) + scale(WE3),
##     data = Abortion.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -1.661 -0.498 -0.143  0.661  1.675
##
## Coefficients:
##                           Estimate             Std. Error t value Pr(>|t|)
## (Intercept) -0.0000000000000000246  0.1399929900965807272    0.00   1.0000
## scale(WE1)  -0.4975363698413171609  0.1599535244454814986   -3.11   0.0039 **
## scale(WE2)   0.3914879877503840699  0.1566888233313864787    2.50   0.0178 *
## scale(WE3)  -0.2521540831323890175  0.1591436279784461394   -1.58   0.1229
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.84 on 32 degrees of freedom
## Multiple R-squared:  0.355,Adjusted R-squared:  0.294
## F-statistic: 5.87 on 3 and 32 DF,  p-value: 0.0026
```

To obtain the predicted values, use the `predict()` function using only values of interest for the predictors

```
# Catholic
Cath.data3 <- data.frame("WE1"=1, "WE2"=0, "WE3"=0)
predict(Abortion3.fit,newdata=Cath.data3)

##    1
## 60.7

# Protestant
Pros.data3 <- data.frame("WE1"=-Abortion.c.n/Abortion.p.n, "WE2"=-Abortion.j.n/Abortion.p.n,
"WE3"=-Abortion.o.n/Abortion.p.n)
predict(Abortion3.fit,newdata=Pros.data3)

##    c
## 93.3

# Jewish
Jew.data3 <- data.frame("WE1"=0, "WE2"=1, "WE3"=0)
predict(Abortion3.fit,newdata=Jew.data3)
```

```
##   1
## 104

# Other
Other.data3 <- data.frame("WE1"=0, "WE2"=0, "WE3"=1)
predict(Abortion3.fit,newdata=Other.data3)

##    1
## 70.1
```

The residual error can be obtained from the regression results

```
summary(Abortion2.fit)$sigma^2

## [1] 548
```

To obtain the confidence intervals, use the `confint()` function.

```
# Unstandardized
confint(Abortion3.fit,level = 0.95)

##              2.5 % 97.5 %
## (Intercept)  73.74  89.64
## WE1         -34.80  -7.26
## WE2           4.03  39.58
## WE3         -26.44   3.30
```

8.3 Contrast Coding

Contrast coding is dependent on the user's hypotheses. It has to be done manually in **R**.[2] Note that the `ifelse()` function can take logical operators, some of which are shown in Table 8.3.

Table 8.3: Some logical operators to use in R

Operator	Description
`==`	Equal to
`!=`	Not equal to
`x | y`	x **or** y
`x & y`	x **and** y

```
# Contrast Coding, majority vs. minority religions
# Catholic/Prostestant vs. Jewish/other
Abortion.data$Con1<-ifelse(Abortion.data$Group=="c" | Abortion.data$Group=="p", .5, -.5)
# Cathooic vs. Protestant
Abortion.data$Con2<-ifelse(Abortion.data$Group=="c", .5, ifelse(Abortion.data$Group=="p", -.5,0))
# Jewish vs. Other
Abortion.data$Con3<-ifelse(Abortion.data$Group=="j", .5, ifelse(Abortion.data$Group=="o", -.5,0))
head(Abortion.data)

##   Case Group ATA C1 C2 C3 E1 E2 E3   WE1   WE2   WE3 Con1 Con2 Con3
## 1    1     c  61  1  0  0  1  0  0 1.000 0.000 0.000  0.5  0.5  0.0
```

[2]You *can* do it via the `contrasts()` function, but you still have to manually figure out the proportions. Consequently, it is easier to create the weighted effects variables manually.

```
## 2    2     o  78  0  0  1  0  0  1  0.000  0.000  1.000 -0.5  0.0 -0.5
## 3    3     p  47  0  0  0 -1 -1 -1 -0.692 -0.462 -0.615  0.5 -0.5  0.0
## 4    4     c  65  1  0  0  1  0  0  1.000  0.000  0.000  0.5  0.5  0.0
## 5    5     c  45  1  0  0  1  0  0  1.000  0.000  0.000  0.5  0.5  0.0
## 6    6     o 106  0  0  1  0  0  1  0.000  0.000  1.000 -0.5  0.0 -0.5
```

The partial and semi-partial correlations are calculated via the `ppcor()` function in the `ppcor` package.

```
# Create variables-of-interest set
Abortion.vars3 <- c("ATA", "Con1", "Con2", "Con3")

library(ppcor)
# partial
pcor(Abortion.data[Abortion.vars3])

## $estimate
##         ATA   Con1   Con2  Con3
## ATA   1.000 -0.209 -0.494 0.423
## Con1 -0.209  1.000 -0.200 0.187
## Con2 -0.494 -0.200  1.000 0.209
## Con3  0.423  0.187  0.209 1.000
##
## $p.value
##          ATA  Con1    Con2   Con3
## ATA  0.00000 0.235 0.00298 0.0127
## Con1 0.23456 0.000 0.25618 0.2890
## Con2 0.00298 0.256 0.00000 0.2359
## Con3 0.01274 0.289 0.23585 0.0000
##
## $statistic
##        ATA  Con1  Con2 Con3
## ATA   0.00 -1.21 -3.21 2.64
## Con1 -1.21  0.00 -1.16 1.08
## Con2 -3.21 -1.16  0.00 1.21
## Con3  2.64  1.08  1.21 0.00
##
## $n
## [1] 36
##
## $gp
## [1] 2
##
## $method
## [1] "pearson"

# semi-partial/part
spcor(Abortion.data[Abortion.vars3])

## $estimate
##         ATA   Con1   Con2  Con3
## ATA   1.000 -0.172 -0.456 0.375
## Con1 -0.207  1.000 -0.197 0.184
## Con2 -0.491 -0.177  1.000 0.184
## Con3  0.420  0.172  0.192 1.000
##
## $p.value
##          ATA  Con1    Con2  Con3
## ATA  0.00000 0.331 0.00667 0.029
## Con1 0.24072 0.000 0.26338 0.298
## Con2 0.00321 0.318 0.00000 0.296
## Con3 0.01338 0.332 0.27585 0.000
##
```

```
## $statistic
##        ATA   Con1  Con2 Con3
## ATA   0.00 -0.988 -2.90 2.29
## Con1 -1.20  0.000 -1.14 1.06
## Con2 -3.19 -1.015  0.00 1.06
## Con3  2.62  0.986  1.11 0.00
##
## $n
## [1] 36
##
## $gp
## [1] 2
##
## $method
## [1] "pearson"
```

The regression coefficients and their standard error are calculated as with a typical regression.

```
# Regression coefficinets
Abortion4.fit <- lm(ATA ~ Con1 + Con2 + Con3, data=Abortion.data)
summary(Abortion4.fit)

##
## Call:
## lm(formula = ATA ~ Con1 + Con2 + Con3, data = Abortion.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -46.31 -13.90  -3.99  18.42  46.69
##
## Coefficients:
##             Estimate Std. Error t value            Pr(>|t|)
## (Intercept)    81.90       4.05   20.20 <0.0000000000000002 ***
## Con1           -9.83       8.11   -1.21               0.235
## Con2          -32.64      10.15   -3.21               0.003 **
## Con3           33.37      12.65    2.64               0.013 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 23.4 on 32 degrees of freedom
## Multiple R-squared:  0.355,Adjusted R-squared:  0.294
## F-statistic: 5.87 on 3 and 32 DF,  p-value: 0.0026

# Standardized Regression coefficinets, only standardizing the continuous variables
Abortion4.fitStd <- lm(scale(ATA) ~ Con1 + Con2 + Con3, data=Abortion.data)
summary(Abortion4.fitStd)

##
## Call:
## lm(formula = scale(ATA) ~ Con1 + Con2 + Con3, data = Abortion.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -1.661 -0.498 -0.143  0.661  1.675
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.00737    0.14544    0.05    0.960
## Con1        -0.35241    0.29088   -1.21    0.235
## Con2        -1.17076    0.36423   -3.21    0.003 **
## Con3         1.19709    0.45363    2.64    0.013 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.84 on 32 degrees of freedom
## Multiple R-squared:  0.355,Adjusted R-squared:  0.294
## F-statistic: 5.87 on 3 and 32 DF,  p-value: 0.0026

# Standardized Regression coefficinets uisng CCAW's method
Abortion4.fitStd2 <- lm(scale(ATA) ~ scale(Con1) + scale(Con2) + scale(Con3), data=Abortion.data)
summary(Abortion4.fitStd2)

##
## Call:
## lm(formula = scale(ATA) ~ scale(Con1) + scale(Con2) + scale(Con3),
##     data = Abortion.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -1.661 -0.498 -0.143  0.661  1.675
##
## Coefficients:
##                            Estimate            Std. Error t value Pr(>|t|)
## (Intercept)  0.000000000000000294  0.139992990096580727    0.00    1.000
## scale(Con1) -0.174237395132303213  0.143814566404295407   -1.21    0.235
## scale(Con2) -0.459392548598772266  0.142919517721424416   -3.21    0.003 **
## scale(Con3)  0.377047066197536918  0.142879747209633950    2.64    0.013 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.84 on 32 degrees of freedom
## Multiple R-squared:  0.355,Adjusted R-squared:  0.294
## F-statistic: 5.87 on 3 and 32 DF,  p-value: 0.0026
```

To obtain the predicted values, use the `predict()` function using only values of interest for the predictors

```
# Catholic
Cath.data4 <- data.frame("Con1"=.5, "Con2"=.5, "Con3"=0)
predict(Abortion4.fit,newdata=Cath.data4)

##    1
## 60.7

# Protestant
Pros.data4 <- data.frame("Con1"=.5, "Con2"=-.5, "Con3"=0)
predict(Abortion4.fit,newdata=Pros.data4)

##    1
## 93.3

# Jewish
Jew.data4<- data.frame("Con1"=-.5, "Con2"=0, "Con3"=.5)
predict(Abortion4.fit,newdata=Jew.data4)

##   1
## 104

# Other
Other.data4 <- data.frame("Con1"=-.5, "Con2"=0, "Con3"=-.5)
predict(Abortion4.fit,newdata=Other.data4)

##    1
## 70.1
```

The residual error can be obtained from the regression results

```
summary(Abortion4.fit)$sigma^2

## [1] 548
```

8.4 Coding Schemes in the Context of Other Independent Variables

Import the city dweller data

```
#City dweller data (Table 8.7.1)
Dweller.data <- read.table("C08e02dt.txt")
```

Name the variables

```
# the .c suffix means the variable is a mean-centered version of the original variable
colnames(Dweller.data)<-c("smallTown","City","Rural","Altruism","SES","Neuroticism","SES.c","Neuroticism.c" )
```

The descriptive statistics in Table 8.7.1 and Table 8.7.2 can be acquired by using the `describe()` and `describeBy()` functions from the `psych` package.

```
library(psych)
# Table 8.7.1
describe(Dweller.data[c("Altruism","Neuroticism")])

##             vars   n mean    sd median trimmed   mad  min  max range skew kurtosis   se
## Altruism       1 150 46.4 14.48   47.7    46.4 15.06 11.8 84.0  72.2 0.01    -0.43 1.18
## Neuroticism    2 150 56.3  9.72   56.9    56.3  9.56 32.2 85.3  53.1 0.01    -0.09 0.79

describeBy(Dweller.data[c("Altruism","Neuroticism")],Dweller.data$City,mat=TRUE)

##              item group1 vars  n mean   sd median trimmed   mad  min  max range    skew kurtosis
## Altruism1       1      0    1 95 53.2 11.8   53.0    53.2 12.02 25.1 84.0  58.8  0.0911   -0.152
## Altruism2       2      1    1 55 34.8 10.9   33.1    34.6 10.38 11.8 57.2  45.5  0.1352   -0.687
## Neuroticism1    3      0    2 95 55.0  9.1   54.6    54.9 10.01 32.2 75.9  43.8  0.0511   -0.483
## Neuroticism2    4      1    2 55 58.4 10.4   58.9    58.7  9.58 34.2 85.3  51.1 -0.1761    0.250
##                se
## Altruism1    1.209
## Altruism2    1.473
## Neuroticism1 0.933
## Neuroticism2 1.408
```

To calculate the partialed variables, use the regular regression function, `lm()`, and the extract the residuals from the `summary()` function.

```
# Regression coefficinets
Dweller.fit1 <- lm(Altruism ~ City + Neuroticism.c, data=Dweller.data)
Dweller.fit2 <- lm(Altruism ~ Neuroticism.c, data=Dweller.data)
Dweller.fit3 <- lm(City ~ Neuroticism.c, data=Dweller.data)

# Residuals
summary(Dweller.fit2)$residuals

##        1        2        3        4        5        6        7        8        9       10       11
##  24.4443  16.1073  13.5784  19.2955  13.0234   4.8943  16.4152  19.4397  24.4265  -0.0354  13.0806
##       12       13       14       15       16       17       18       19       20       21       22
##  35.7091  22.9415   4.1399  13.2446  -3.1959   8.6714  15.0901  12.0332  29.8033   2.5780   9.8891
##       23       24       25       26       27       28       29       30       31       32       33
##   0.7263   1.4321   8.5552   9.9608  11.3585 -19.9227  26.9768  -1.8962  24.7954   3.9725   8.0659
##       34       35       36       37       38       39       40       41       42       43       44
##  19.4211  20.3259   7.5796   6.8811  21.5905  13.7457   3.4190   8.4294  -1.7940  -0.7486   4.8894
##       45       46       47       48       49       50       51       52       53       54       55
##  12.0527  -8.0178   8.4021   3.8754  16.7432  13.1882 -19.1968  16.1183   3.2473  -1.0652  -1.8672
##       56       57       58       59       60       61       62       63       64       65       66
##  -7.7597  -8.5500 -20.7523  23.1918  -4.8539  -1.3943  -9.9210   4.6581   3.7794  -7.5618  -0.7105
##       67       68       69       70       71       72       73       74       75       76       77
##  11.9357   6.3113  -3.9508  25.8511  -7.4209   3.7870   7.4971  -4.0805 -20.2344  11.9628   7.7432
##       78       79       80       81       82       83       84       85       86       87       88
##   7.6729  11.0613  19.3396  -7.2222  -1.2688  -0.0671  -6.5896  -0.4720  -8.1480  20.6374  -6.0398
##       89       90       91       92       93       94       95       96       97       98       99
##  18.3096   8.9264  -2.2538  -9.6321  20.6702 -12.6838  -2.2212 -16.2312 -17.1208   5.5385 -25.2269
##      100      101      102      103      104      105      106      107      108      109      110
## -12.1502  -3.1125 -33.0424  11.3488 -23.8788   4.6141 -12.5937 -20.5735 -18.6815 -18.5422 -12.8323
##      111      112      113      114      115      116      117      118      119      120      121
## -22.9181 -12.0188 -14.9863 -24.2874  -0.2460 -23.6856 -10.3158 -12.9703  -8.3418 -16.5709 -28.5756
##      122      123      124      125      126      127      128      129      130      131      132
## -13.1342  -3.4563 -21.7442  -1.2665   5.6091  -7.6750 -14.7601 -12.5166  -6.2845  -2.3372 -13.1900
##      133      134      135      136      137      138      139      140      141      142      143
##   8.1900 -15.2751  12.9005  -0.8770   6.5114 -17.2442  -0.9756 -16.7468 -12.3806 -14.0746 -27.4635
##      144      145      146      147      148      149      150
## -20.6562  -6.0641 -14.1707  -7.2681   0.2079  -9.8967  -2.9245
```

```r
summary(Dweller.fit3)$residuals
```

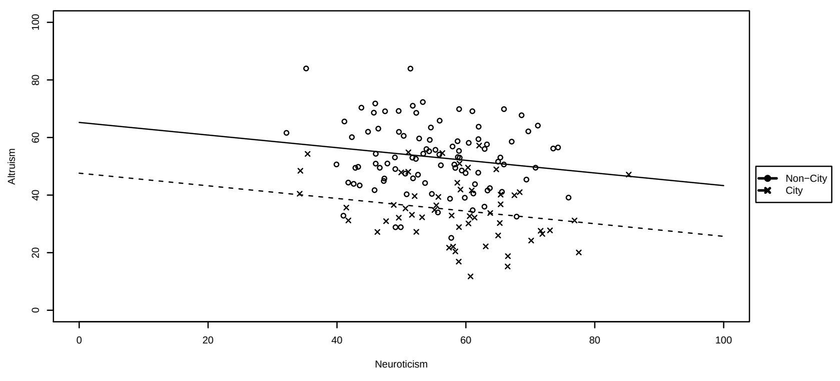
```
##      1      2      3      4      5      6      7      8      9     10     11     12     13     14
## -0.406 -0.512 -0.239 -0.364 -0.283 -0.383 -0.352 -0.292 -0.389 -0.306 -0.310 -0.325 -0.329 -0.280
##     15     16     17     18     19     20     21     22     23     24     25     26     27     28
## -0.402 -0.345 -0.346 -0.414 -0.422 -0.189 -0.398 -0.389 -0.280 -0.295 -0.249 -0.443 -0.270 -0.313
##     29     30     31     32     33     34     35     36     37     38     39     40     41     42
## -0.448 -0.253 -0.342 -0.305 -0.350 -0.415 -0.310 -0.390 -0.342 -0.279 -0.425 -0.414 -0.489 -0.229
##     43     44     45     46     47     48     49     50     51     52     53     54     55     56
## -0.410 -0.328 -0.351 -0.422 -0.440 -0.365 -0.518 -0.387 -0.238 -0.458 -0.393 -0.319 -0.445 -0.259
##     57     58     59     60     61     62     63     64     65     66     67     68     69     70
## -0.279 -0.379 -0.492 -0.291 -0.257 -0.407 -0.333 -0.384 -0.251 -0.335 -0.337 -0.163 -0.291 -0.471
##     71     72     73     74     75     76     77     78     79     80     81     82     83     84
## -0.244 -0.477 -0.363 -0.408 -0.306 -0.317 -0.448 -0.387 -0.380 -0.261 -0.377 -0.429 -0.532 -0.353
##     85     86     87     88     89     90     91     92     93     94     95     96     97     98
## -0.285 -0.321 -0.480 -0.396 -0.278 -0.358 -0.329 -0.464 -0.333 -0.361 -0.426  0.758  0.517  0.610
##     99    100    101    102    103    104    105    106    107    108    109    110    111    112
##  0.616  0.597  0.594  0.596  0.389  0.547  0.599  0.696  0.756  0.706  0.454  0.558  0.718  0.643
##    113    114    115    116    117    118    119    120    121    122    123    124    125    126
##  0.673  0.624  0.677  0.619  0.641  0.621  0.669  0.611  0.611  0.681  0.609  0.576  0.613  0.562
##    127    128    129    130    131    132    133    134    135    136    137    138    139    140
##  0.460  0.599  0.492  0.557  0.539  0.504  0.633  0.659  0.585  0.687  0.677  0.560  0.532  0.690
##    141    142    143    144    145    146    147    148    149    150
##  0.591  0.819  0.547  0.667  0.818  0.502  0.638  0.809  0.570  0.557
```

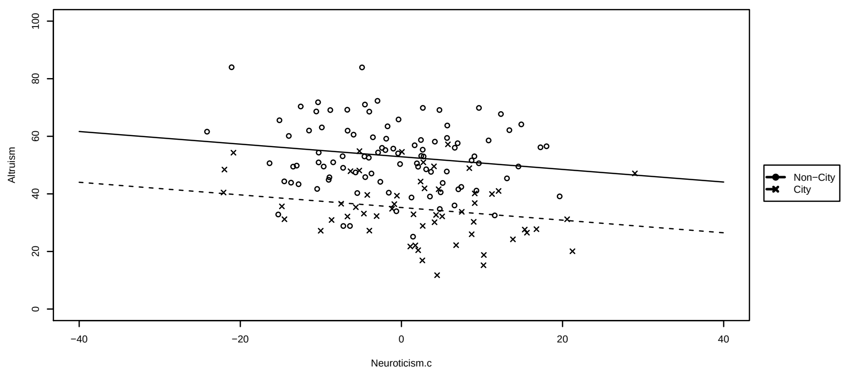
```r
# Partialed variables
Dweller.data$Altruism.Neurot.c <- mean(Dweller.data$Altruism)+summary(Dweller.fit2)$residuals
Dweller.data$City.Neurot.c <- mean(Dweller.data$City)+summary(Dweller.fit3)$residuals
```

Two make different regression lines for different groups, use the `curve()` function along with matrix multiplication `%*%`

```r
# Regressions
AltNeu.fit <- lm(Altruism ~ Neuroticism + City, data=Dweller.data)
AltNeuCent.fit <- lm(Altruism ~ Neuroticism.c + City, data=Dweller.data)

# Scatterplots with lines of best fit
# Two plots on one figure
```

```
par(mfcol=c(2,1), cex=.5)

# Leave room for legend
par(mar=par()$mar+c(0,0,0,6))


# Uncentered
# Non-City dwellers
plot(Altruism[City==0]~Neuroticism[City==0], data=Dweller.data,pch=1, xlab="Neuroticism", ylab="Altruism", xlim
curve (cbind (1, x, 1) %*% coef(AltNeu.fit), add=TRUE, lty=2)
# City dwellers
points(Altruism[City==1]~Neuroticism[City==1], data=Dweller.data,pch=4)
curve (cbind (1, x, 0) %*% coef(AltNeu.fit), add=TRUE, lty=1)

legend(105,50,c("Non-City", "City"), lwd = 2, pch=c(1,4),lty=c(1,2),xpd=TRUE)

# Centered
# Non-City dwellers
plot(Altruism[City==0]~Neuroticism.c[City==0], data=Dweller.data,pch=1, xlab="Neuroticism.c", ylab="Altruism",
curve (cbind (1, x, 1) %*% coef(AltNeuCent.fit), add=TRUE, lty=2)
# City dwellers
points(Altruism[City==1]~Neuroticism.c[City==1], data=Dweller.data,pch=4)
curve (cbind (1, x, 0) %*% coef(AltNeuCent.fit), add=TRUE, lty=1)
legend(45,50,c("Non-City", "City"), lwd = 2, pch=c(1,4),lty=c(1,2), xpd=TRUE)
```





```
library(psych)
# Table 8.7.2
describeBy(Dweller.data[c("Altruism","Neuroticism", "SES")],list(Dweller.data$City,Dweller.data$Rural),mat=TRUE

##              item group1 group2 vars  n mean    sd median trimmed   mad  min  max range    skew
## Altruism1       1      0      0    1 39 59.7 10.61   58.1    59.7 10.63 28.9 84.0  55.1 -0.0452
## Altruism2       2      1      0    1 55 34.8 10.92   33.1    34.6 10.38 11.8 57.2  45.5  0.1352
## Altruism3       3      0      1    1 56 48.6 10.37   48.2    48.4 10.82 25.1 70.4  45.2  0.1203
## Altruism4        4     1     1   NA NA   NA    NA     NA      NA    NA   NA   NA    NA      NA
## Neuroticism1     5     0     0    2 39 53.5  8.06   53.4    53.4  8.79 35.2 73.5  38.3  0.1243
## Neuroticism2     6     1     0    2 55 58.4 10.44   58.9    58.7  9.58 34.2 85.3  51.1 -0.1761
## Neuroticism3     7     0     1    2 56 56.1  9.68   56.0    56.1 10.23 32.2 75.9  43.8 -0.0780
## Neuroticism4     8     1     1   NA NA   NA    NA     NA      NA    NA   NA   NA    NA      NA
## SES1             9     0     0    3 39 46.3 11.96   46.1    45.8 13.57 26.2 77.8  51.6  0.3971
## SES2            10     1     0    3 55 51.2  9.72   50.6    50.9  9.56 24.1 74.8  50.6  0.0974
## SES3            11     0     1    3 56 44.1  9.63   43.5    44.4  9.07 19.0 61.8  42.8 -0.3120
## SES4            12     1     1   NA NA   NA    NA     NA      NA    NA   NA   NA    NA      NA
##              kurtosis   se
## Altruism1       0.724 1.70
## Altruism2      -0.687 1.47
## Altruism3      -0.499 1.39
## Altruism4          NA   NA
## Neuroticism1   -0.404 1.29
## Neuroticism2    0.250 1.41
## Neuroticism3   -0.616 1.29
## Neuroticism4       NA   NA
## SES1           -0.521 1.92
## SES2            0.345 1.31
## SES3           -0.139 1.29
## SES4               NA   NA

cor(Dweller.data[c("Altruism","City", "Rural","Neuroticism", "SES")])

##             Altruism   City   Rural Neuroticism     SES
## Altruism      1.0000 -0.613  0.1154     -0.2468 -0.0246
## City         -0.6133  1.000 -0.5873      0.1693  0.2807
## Rural         0.1154 -0.587  1.0000     -0.0143 -0.2314
## Neuroticism  -0.2468  0.169 -0.0143      1.0000  0.1183
## SES          -0.0246  0.281 -0.2314      0.1183  1.0000
```

```
# Regression coefficinets
Dweller.noCov.fit <- lm(Altruism ~ City + Rural, data=Dweller.data)
Dweller.cenCov.fit <- lm(Altruism ~ City + Rural + Neuroticism.c + SES.c, data=Dweller.data)
Dweller.Cov.fit <- lm(Altruism ~ City + Rural + Neuroticism + SES, data=Dweller.data)
```

To obtain the predicted values, use the `predict()` function using only values of interest for the predictors

```
# Centered
# Town
Town.data.c <- data.frame("City"=0, "Rural"=0, "Neuroticism.c"=mean(Dweller.data$Neuroticism.c),
"SES.c"=mean(Dweller.data$SES.c))
predict(Dweller.cenCov.fit,newdata=Town.data.c)

##    1
## 59.4

# City
City.data.c <- data.frame("City"=1, "Rural"=0, "Neuroticism.c"=mean(Dweller.data$Neuroticism.c),
"SES.c"=mean(Dweller.data$SES.c))
predict(Dweller.cenCov.fit,newdata=City.data.c)

##    1
## 34.4

# Rural
Rural.data.c <- data.frame("City"=0, "Rural"=1, "Neuroticism.c"=mean(Dweller.data$Neuroticism.c),
"SES.c"=mean(Dweller.data$SES.c))
predict(Dweller.cenCov.fit,newdata=Rural.data.c)
```

```
##    1
## 49.2

# Non-Centered
# Town
Town.data <- data.frame("City"=0, "Rural"=0, "Neuroticism"=mean(Dweller.data$Neuroticism),
 "SES"=mean(Dweller.data$SES))
predict(Dweller.Cov.fit,newdata=Town.data)

##    1
## 59.4

# City
City.data <- data.frame("City"=1, "Rural"=0, "Neuroticism"=mean(Dweller.data$Neuroticism),
 "SES"=mean(Dweller.data$SES))
predict(Dweller.Cov.fit,newdata=City.data)

##    1
## 34.4

# Rural
Rural.data <- data.frame("City"=0, "Rural"=1, "Neuroticism"=mean(Dweller.data$Neuroticism),
 "SES"=mean(Dweller.data$SES))
predict(Dweller.Cov.fit,newdata=Rural.data)

##    1
## 49.2
```

8.5 Further Reading

For information on other types of coding, see http://www.ats.ucla.edu/stat/r/library/contrast_coding.htm

Chapter 9

Interactions with Categorical Variables

9.1 Nominal scale by nominal scale interactions

9.1.1 The 2 by 2 design

Import the data in CCAW table 9.1.1.[1]

```
# Table 9.1.1 data
table911.data <- read.table("C0901DT.txt", header=TRUE)
head(table911.data)

##     YA   YB   YC  LESION   DRUG
## 1 7.35 15.3 9.35 SURGERY ACTIVE
## 2 6.91 14.9 8.91 SURGERY ACTIVE
## 3 7.09 15.1 9.09 SURGERY ACTIVE
## 4 5.43 13.4 7.43 SURGERY ACTIVE
## 5 6.10 14.1 8.10 SURGERY ACTIVE
## 6 7.36 15.4 9.36 SURGERY ACTIVE
```

The categories are different in the data set than shown in the book. The *Surgery* category for the `LESION` variable in the dataset is the same as the *Frontal* category in CCAW Table 9.1.1. We will re-code it for consistency with the book's output.

```
library(car)
# Save original coding as new variable
table911.data$LESION.orig <- table911.data$LESION
# Recode LESION variable
table911.data$LESION <- recode(table911.data$LESION, '"SURGERY"="FRONTAL"')
```

Combining the `xtabs()` and `aggregate()` functions in **R** can produce the cell means in CCAW's table 9.1.1. The `apply()` function can produce the marginal means.

```
# No Interaction
# Cell means
ya <- round(xtabs(YA~DRUG+LESION, aggregate(YA~DRUG+LESION,table911.data,mean)))
# Column means
apply(ya, 1, mean)

##  ACTIVE PLACEBO
##       5       9
```

[1]The `LESION` and `DRUG` variable names have a $ attached to them in the original raw data, e.g., `LESION$`. I removed them before entering the data. In general, it is not good practice to name variables, at least for analysis in **R**, using characters other than letters and numbers, although there are some exceptions (e.g, a period, .)

```
# Row means
apply(ya, 2, mean)

## FRONTAL    SHAM
##       8       6

# Crossed Interaction
# Cell means
yb <- round(xtabs(YB~DRUG+LESION, aggregate(YB~DRUG+LESION,table911.data,mean)))
# Column means
apply(yb, 1, mean)

##  ACTIVE PLACEBO
##       9      11

# Row means
apply(yb, 2, mean)

## FRONTAL    SHAM
##      12       8

# Ordinal Interaction
# Cell means
yc <- round(xtabs(YC~DRUG+LESION, aggregate(YC~DRUG+LESION,table911.data,mean)))
# Column means
apply(yc, 1, mean)

##  ACTIVE PLACEBO
##       5      11

# Row means
apply(yc, 2, mean)

## FRONTAL    SHAM
##      10       6
```

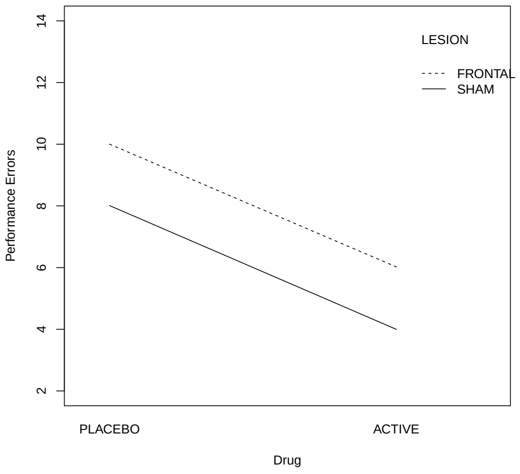
```
# Combination table for the No Interaction table
rbind(cbind(ya,apply(ya, 1, mean)),c(apply(ya, 2, mean), round(mean(table911.data$YA))))

##         FRONTAL SHAM
## ACTIVE        6    4 5
## PLACEBO      10    8 9
##               8    6 7
```

To create plots of the interactions with categorical variables, use the `interaction.plot()` function. First, we need to reorder the `DRUG` variable's levels to match CCAW's Figure 9.1.1.

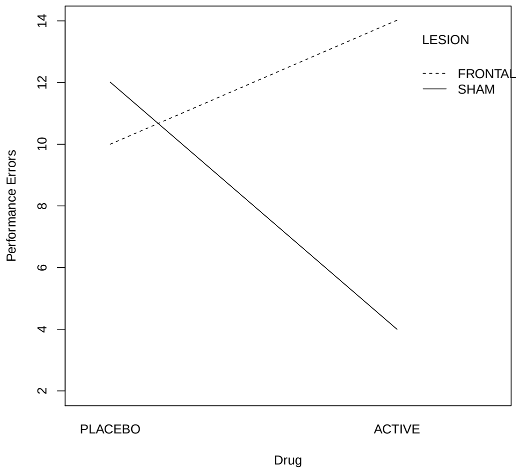
```
# Reorder the DRUG variable's levels to match CCAW's Figure 9.1.1
table911.data$DRUG <- relevel(table911.data$DRUG, ref="PLACEBO")
```

The plots are given in Figure 9.1

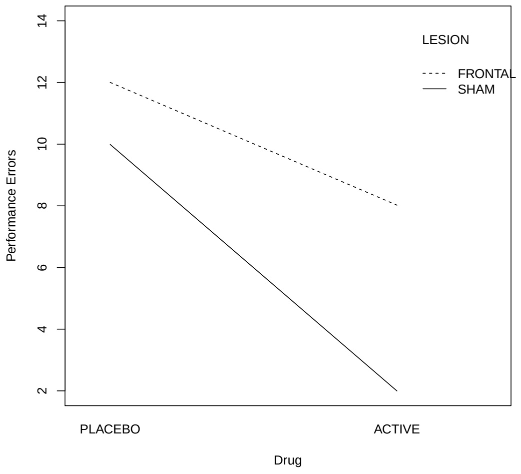
```
with(table911.data,
interaction.plot(DRUG, LESION, YA, fun=mean, xlab="Drug", ylab="Performance Errors", ylim=c(2,14))
)
with(table911.data,
interaction.plot(DRUG, LESION, YB, fun=mean, xlab="Drug", ylab="Performance Errors", ylim=c(2,14))
)
with(table911.data,
interaction.plot(DRUG, LESION, YC, fun=mean, xlab="Drug", ylab="Performance Errors", ylim=c(2,14))
)
```



(a) No interaction



(b) Disordinal interaction



(c) Ordinal interaction

Figure 9.1: Types of interactions with nominal variables

By default, since the `DRUG` and `LESION` variables have characters as input, **R** treats them like factors in a dataframe environment. Thus, they will automatically be dummy coded.

```
# Order the levels of DRUG and LEVEL factors to match CCAW dummy coding scheme (cf. Table 9.1.3)
table911.data$DRUG <- relevel(table911.data$DRUG, ref="PLACEBO")
table911.data$LESION <- relevel(table911.data$LESION, ref="SHAM")

yaDum.fit <- lm(YA~DRUG*LESION , data=table911.data)
summary(yaDum.fit)

##
## Call:
## lm(formula = YA ~ DRUG * LESION, data = table911.data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7389 -0.6627  0.0319  0.5989  2.3912
##
## Coefficients:
##                           Estimate Std. Error t value             Pr(>|t|)
## (Intercept)                 8.0106     0.1380   58.05 < 0.0000000000000002 ***
## DRUGACTIVE                 -4.0141     0.2056  -19.52 < 0.0000000000000002 ***
## LESIONFRONTAL               1.9915     0.2303    8.65   0.0000000000000084 ***
## DRUGACTIVE:LESIONFRONTAL    0.0336     0.3256    0.10                 0.92
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.976 on 147 degrees of freedom
## Multiple R-squared:  0.834,Adjusted R-squared:  0.83
## F-statistic:  245 on 3 and 147 DF,  p-value: <0.0000000000000002

confint(yaDum.fit)

##                           2.5 % 97.5 %
## (Intercept)                7.74  8.283
## DRUGACTIVE                -4.42 -3.608
## LESIONFRONTAL              1.54  2.447
## DRUGACTIVE:LESIONFRONTAL -0.61  0.677

ybDum.fit <- lm(YB~DRUG*LESION , data=table911.data)
summary(ybDum.fit)

##
## Call:
## lm(formula = YB ~ DRUG * LESION, data = table911.data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7389 -0.6627  0.0319  0.5989  2.3912
##
## Coefficients:
##                           Estimate Std. Error t value             Pr(>|t|)
## (Intercept)                 12.011      0.138   87.03 < 0.0000000000000002 ***
## DRUGACTIVE                  -8.014      0.206  -38.98 < 0.0000000000000002 ***
## LESIONFRONTAL               -2.008      0.230   -8.72   0.0000000000000055 ***
## DRUGACTIVE:LESIONFRONTAL    12.034      0.326   36.96 < 0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.976 on 147 degrees of freedom
## Multiple R-squared:  0.94,Adjusted R-squared:  0.939
## F-statistic:  769 on 3 and 147 DF,  p-value: <0.0000000000000002
```

```
confint(ybDum.fit)

##                            2.5 % 97.5 %
## (Intercept)                11.74  12.28
## DRUGACTIVE                 -8.42  -7.61
## LESIONFRONTAL              -2.46  -1.55
## DRUGACTIVE:LESIONFRONTAL 11.39  12.68

ycDum.fit <- lm(YC~DRUG*LESION , data=table911.data)
summary(ycDum.fit)

##
## Call:
## lm(formula = YC ~ DRUG * LESION, data = table911.data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7389 -0.6132  0.0319  0.5536  2.3912
##
## Coefficients:
##                          Estimate Std. Error t value             Pr(>|t|)
## (Intercept)                 9.991      0.136   73.53 < 0.0000000000000002 ***
## DRUGACTIVE                 -7.994      0.202  -39.49 < 0.0000000000000002 ***
## LESIONFRONTAL               2.011      0.227    8.87   0.0000000000000023 ***
## DRUGACTIVE:LESIONFRONTAL    4.014      0.321   12.52 < 0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.961 on 147 degrees of freedom
## Multiple R-squared:  0.94,Adjusted R-squared:  0.939
## F-statistic:  764 on 3 and 147 DF,  p-value: <0.0000000000000002

confint(ycDum.fit)

##                            2.5 % 97.5 %
## (Intercept)                 9.72  10.26
## DRUGACTIVE                 -8.39  -7.59
## LESIONFRONTAL               1.56   2.46
## DRUGACTIVE:LESIONFRONTAL  3.38   4.65
```

To get the cell means (i.e., results in CCAW Table 9.1.2), use the `lsmeans()` function in the `lsmeans` package.

```
library(lsmeans)

## Error in library(lsmeans):  there is no package called 'lsmeans'

# No interaction
lsmeans(yaDum.fit, pairwise~DRUG+LESION)

## Error in eval(expr, envir, enclos):  could not find function "lsmeans"

# Crossed interaction
lsmeans(ybDum.fit, pairwise~DRUG+LESION)

## Error in eval(expr, envir, enclos):  could not find function "lsmeans"

# Ordinal interaction
lsmeans(ycDum.fit, pairwise~DRUG+LESION)

## Error in eval(expr, envir, enclos):  could not find function "lsmeans"
```

Contrast coding was discussed earlier in section 8.3, so will not be reviewed here.

```
# Contrast coding
table911.data$LESION.con <- ifelse(table911.data$LESION == "FRONTAL", .5, -.5)
table911.data$DRUG.con <- ifelse(table911.data$DRUG=="ACTIVE", .5, -.5)
```

```
# Regression models with contrast codes

yaCont.fit <- lm(YA~DRUG.con*LESION.con , data=table911.data)
summary(yaCont.fit)

##
## Call:
## lm(formula = YA ~ DRUG.con * LESION.con, data = table911.data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7389 -0.6627  0.0319  0.5989  2.3912
##
## Coefficients:
##                      Estimate Std. Error t value            Pr(>|t|)
## (Intercept)            7.0077     0.0814    86.1 <0.0000000000000002 ***
## DRUG.con              -3.9972     0.1628   -24.6 <0.0000000000000002 ***
## LESION.con             2.0083     0.1628    12.3 <0.0000000000000002 ***
## DRUG.con:LESION.con    0.0336     0.3256     0.1                0.92
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.976 on 147 degrees of freedom
## Multiple R-squared:  0.834,Adjusted R-squared:  0.83
## F-statistic:  245 on 3 and 147 DF,  p-value: <0.0000000000000002

confint(yaCont.fit)

##                      2.5 % 97.5 %
## (Intercept)           6.85  7.169
## DRUG.con             -4.32 -3.675
## LESION.con            1.69  2.330
## DRUG.con:LESION.con  -0.61  0.677

ybCont.fit <- lm(YB~DRUG.con*LESION.con , data=table911.data)
summary(ybCont.fit)

##
## Call:
## lm(formula = YB ~ DRUG.con * LESION.con, data = table911.data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7389 -0.6627  0.0319  0.5989  2.3912
##
## Coefficients:
##                      Estimate Std. Error t value            Pr(>|t|)
## (Intercept)           10.0077     0.0814   122.9 <0.0000000000000002 ***
## DRUG.con              -1.9972     0.1628   -12.3 <0.0000000000000002 ***
## LESION.con             4.0083     0.1628    24.6 <0.0000000000000002 ***
## DRUG.con:LESION.con   12.0336     0.3256    37.0 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.976 on 147 degrees of freedom
## Multiple R-squared:  0.94,Adjusted R-squared:  0.939
## F-statistic:  769 on 3 and 147 DF,  p-value: <0.0000000000000002
```

```
confint(ybCont.fit)

##                        2.5 % 97.5 %
## (Intercept)             9.85  10.17
## DRUG.con               -2.32  -1.68
## LESION.con              3.69   4.33
## DRUG.con:LESION.con 11.39  12.68

ycCont.fit <- lm(YC~DRUG.con*LESION.con , data=table911.data)
summary(ycCont.fit)

##
## Call:
## lm(formula = YC ~ DRUG.con * LESION.con, data = table911.data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7389 -0.6132  0.0319  0.5536  2.3912
##
## Coefficients:
##                       Estimate Std. Error t value            Pr(>|t|)
## (Intercept)             8.0027     0.0801    99.8 <0.0000000000000002 ***
## DRUG.con               -5.9872     0.1603   -37.4 <0.0000000000000002 ***
## LESION.con              4.0183     0.1603    25.1 <0.0000000000000002 ***
## DRUG.con:LESION.con   4.0136     0.3206    12.5 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.961 on 147 degrees of freedom
## Multiple R-squared:  0.94,Adjusted R-squared:  0.939
## F-statistic:  764 on 3 and 147 DF,  p-value: <0.0000000000000002

confint(ycCont.fit)

##                        2.5 % 97.5 %
## (Intercept)             7.84   8.16
## DRUG.con               -6.30  -5.67
## LESION.con              3.70   4.34
## DRUG.con:LESION.con  3.38   4.65
```

9.1.2 Regression analyses of multiple sets of nominal variables with more than two categories

Import the data.

```
# Treatment data
treat.data <- read.table("C0902DT.txt", header=TRUE)
head(treat.data)

##   HOSPITAL TREATMEN    Y
## 1        1        1 66.8
## 2        1        1 42.4
## 3        1        1 20.7
## 4        1        1 46.4
## 5        1        1 54.3
## 6        1        1 38.0
```

It is difficult to estimate interactions without main effects in **R** using the native interaction functions (i.e, : and *). Consequently, we'll create the main effects and interaction terms by hand.

```
# Hospital Effects coding
treat.data$hosp1e <- ifelse(treat.data$HOSPITAL == "1", 1, ifelse(treat.data$HOSPITAL == "4", -1, 0))
treat.data$hosp2e <- ifelse(treat.data$HOSPITAL == "2", 1, ifelse(treat.data$HOSPITAL == "4", -1, 0))
treat.data$hosp3e <- ifelse(treat.data$HOSPITAL == "3", 1, ifelse(treat.data$HOSPITAL == "4", -1, 0))

# Treatment Effects coding
treat.data$treat4e <- ifelse(treat.data$TREATMEN == "2", 1, ifelse(treat.data$TREATMEN == "3", -1, 0))
treat.data$treat5e <- ifelse(treat.data$TREATMEN == "1", 1, ifelse(treat.data$TREATMEN == "3", -1, 0))

# Effects interactions
treat.data$HbyTE6 <- treat.data$hosp1e*treat.data$treat4e
treat.data$HbyTE7  <-  treat.data$hosp1e*treat.data$treat5e
treat.data$HbyTE8  <-  treat.data$hosp2e*treat.data$treat4e
treat.data$HbyTE9  <-  treat.data$hosp2e*treat.data$treat5e
treat.data$HbyTE10  <-  treat.data$hosp3e*treat.data$treat4e
treat.data$HbyTE11  <-  treat.data$hosp3e*treat.data$treat5e

# Treatment Dummy coding
treat.data$treat4d <- ifelse(treat.data$TREATMEN == "1", 1, 0)
treat.data$treat5d <- ifelse(treat.data$TREATMEN == "2", 1, 0)

# Dummy interactions
treat.data$HbyTD6 <- treat.data$hosp1e*treat.data$treat4d
treat.data$HbyTD7  <-  treat.data$hosp1e*treat.data$treat5d
treat.data$HbyTD8  <-  treat.data$hosp2e*treat.data$treat4d
treat.data$HbyTD9  <-  treat.data$hosp2e*treat.data$treat5d
treat.data$HbyTD10  <-  treat.data$hosp3e*treat.data$treat4d
treat.data$HbyTD11  <-  treat.data$hosp3e*treat.data$treat5d
```

To get the results in CCAW Table 9.1.4, use the `aov()` function.

```
# Table 9.1.4 results
# H,T, HxT
treat1.fit <- aov(Y~hosp1e + hosp2e + hosp3e + treat4e + treat5e + HbyTE6 + HbyTE7 + HbyTE8 + HbyTE9 + HbyTE10 +
summary(treat1.fit)

##              Df Sum Sq Mean Sq F value          Pr(>F)
## hosp1e        1   2149    2149    6.68         0.01127 *
## hosp2e        1    720     720    2.24         0.13805
## hosp3e        1     74      74    0.23         0.63249
## treat4e       1  11930   11930   37.07 0.0000000233761 ***
## treat5e       1  15948   15948   49.56 0.0000000002873 ***
## HbyTE6        1   3758    3758   11.68         0.00093 ***
## HbyTE7        1  21593   21593   67.10 0.0000000000011 ***
## HbyTE8        1  11135   11135   34.60 0.0000000588699 ***
## HbyTE9        1    622     622    1.93         0.16770
## HbyTE10       1   4094    4094   12.72         0.00057 ***
## HbyTE11       1   3186    3186    9.90         0.00220 **
## Residuals    96  30891     322
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

# H, HxT
treat2.fit <- aov(Y~hosp1e + hosp2e + hosp3e + HbyTE6 + HbyTE7 + HbyTE8 + HbyTE9 + HbyTE10 + HbyTE11, data=trea
summary(treat2.fit)

##              Df Sum Sq Mean Sq F value       Pr(>F)
## hosp1e        1   2149    2149    3.98       0.0488 *
## hosp2e        1    720     720    1.33       0.2511
## hosp3e        1     74      74    0.14       0.7119
## HbyTE6        1   4883    4883    9.05       0.0033 **
## HbyTE7        1  24590   24590   45.55 0.0000000011 ***
## HbyTE8       1  11982   11982   22.20 0.0000081320 ***
## HbyTE9       1    897     897    1.66       0.2004
## HbyTE10      1   3225    3225    5.97       0.0163 *
## HbyTE11      1   4673    4673    8.66       0.0041 **
## Residuals   98  52907     540
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

# T, HxT
treat3.fit <- aov(Y~treat4e + treat5e + HbyTE6 + HbyTE7 + HbyTE8 + HbyTE9 + HbyTE10 + HbyTE11, data=treat.data)
summary(treat3.fit)

##             Df Sum Sq Mean Sq F value          Pr(>F)
## treat4e      1  13196   13196   37.30 0.0000000199485 ***
## treat5e      1  15742   15742   44.50 0.0000000014724 ***
## HbyTE6       1   3100    3100    8.76         0.00385 **
## HbyTE7       1  21046   21046   59.49 0.0000000000098 ***
## HbyTE8       1   9431    9431   26.66 0.0000012556428 ***
## HbyTE9       1    517     517    1.46         0.22966
## HbyTE10      1   4794    4794   13.55         0.00038 ***
## HbyTE11      1   3251    3251    9.19         0.00311 **
## Residuals   99  35023     354
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

# H,T
treat4.fit <- aov(Y~hosp1e + hosp2e + hosp3e + treat4e + treat5e, data=treat.data)
summary(treat4.fit)

##              Df Sum Sq Mean Sq F value  Pr(>F)
## hosp1e        1   2149    2149    2.91 0.09098 .
## hosp2e        1    720     720    0.98 0.32573
## hosp3e        1     74      74    0.10 0.75206
## treat4e       1  11930   11930   16.16 0.00011 ***
## treat5e       1  15948   15948   21.61 0.00001 ***
## Residuals   102  75280     738
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

For the regression in CCAW Table 9.1.6, use the dummy coding of the Treatment variable (and the subsequent interaction terms)

```
treat.fit <- lm(Y~hosp1e + hosp2e + hosp3e + treat4d + treat5d + HbyTD6 + HbyTD7 + HbyTD8 + HbyTD9 + HbyTD10 +
summary(treat.fit)

##
## Call:
## lm(formula = Y ~ hosp1e + hosp2e + hosp3e + treat4d + treat5d +
##     HbyTD6 + HbyTD7 + HbyTD8 + HbyTD9 + HbyTD10 + HbyTD11, data = treat.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -44.56  -9.99   0.87   9.08  43.15
##
## Coefficients:
##             Estimate Std. Error t value             Pr(>|t|)
## (Intercept)    78.01       3.08   25.31 < 0.0000000000000002 ***
## hosp1e          8.60       6.03    1.43              0.15698
## hosp2e        -18.55       5.23   -3.55              0.00061 ***
## hosp3e         19.94       5.23    3.81              0.00024 ***
## treat4d       -11.29       4.20   -2.69              0.00852 **
## treat5d        23.95       4.45    5.38         0.0000005315 ***
## HbyTD6        -27.49       7.90   -3.48             0.00075 ***
## HbyTD7         13.38       8.03    1.67             0.09900 .
## HbyTD8         22.99       7.18    3.20             0.00186 **
## HbyTD9         51.06       7.60    6.72         0.0000000013 ***
## HbyTD10       -31.92       7.00   -4.56         0.0000148933 ***
## HbyTD11       -26.37       7.46   -3.54             0.00063 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 17.9 on 96 degrees of freedom
## Multiple R-squared:  0.709,Adjusted R-squared:  0.675
## F-statistic: 21.2 on 11 and 96 DF,  p-value: <0.0000000000000002
```

9.2 Interactions involving more than two nominal scales

9.2.1 An example of three nominal scales coded by alternative methods

Import the data.

```
# Treatment data
dweck.data <- read.table("C0903DT.txt", header=TRUE)
head(dweck.data)

##   ATTRIB DIFF FAIL     Y
## 1      1    1    1 12.06
## 2      1    1    1  9.25
## 3      1    1    1 11.12
## 4      1    1    1 14.20
## 5      1    1    1 12.11
## 6      1    1    1 17.50
```

```
# Make factor variables factors
dweck.data$ATTRIB <-  factor(dweck.data$ATTRIB)
dweck.data$DIFF <-  factor(dweck.data$DIFF)
dweck.data$FAIL <-  factor(dweck.data$FAIL)
```

```
# Main effects model
main.fit <- lm(Y~ATTRIB+DIFF+FAIL,data=dweck.data)
anova(main.fit)

## Analysis of Variance Table
##
## Response: Y
##             Df Sum Sq Mean Sq F value         Pr(>F)
## ATTRIB       2    595   297.3   30.05 0.000000000032 ***
## DIFF         1     15    14.7    1.48           0.23
## FAIL         1      3     2.9    0.29           0.59
## Residuals 115   1138     9.9
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

main2way.fit <- lm(Y~ATTRIB+DIFF+FAIL + ATTRIB:DIFF + ATTRIB:FAIL + DIFF:FAIL,data=dweck.data)
anova(main2way.fit)

## Analysis of Variance Table
##
## Response: Y
##               Df Sum Sq Mean Sq F value           Pr(>F)
## ATTRIB        2    595   297.3   38.06 0.00000000000027 ***
## DIFF          1     15    14.7    1.88             0.17
## FAIL          1      3     2.9    0.37             0.54
## ATTRIB:DIFF   2     31    15.3    1.95             0.15
## ATTRIB:FAIL   2    244   122.2   15.64 0.0000010504140 9 ***
## DIFF:FAIL     1      3     3.5    0.45             0.50
## Residuals   110    859     7.8
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

full.fit <- lm(Y~ATTRIB*DIFF*FAIL ,data=dweck.data)
anova(full.fit)

## Analysis of Variance Table
##
## Response: Y
##                   Df Sum Sq Mean Sq F value           Pr(>F)
## ATTRIB             2    595   297.3   38.20 0.00000000000028 ***
## DIFF               1     15    14.7    1.88             0.17
## FAIL               1      3     2.9    0.37             0.54
## ATTRIB:DIFF        2     31    15.3    1.96             0.15
## ATTRIB:FAIL        2    244   122.2   15.70 0.00000103171973 ***
## DIFF:FAIL          1      3     3.5    0.45             0.50
## ATTRIB:DIFF:FAIL   2     19     9.5    1.21             0.30
## Residuals        108    840     7.8
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

To get the cell means (CCAW Table 9.2.3), use the `xtabs()` function. As there are three factors, use the `ftable()` function to flatten the `xtabs()` result.

```
# Cell frequencies
ftable(xtabs(~FAIL+DIFF+ATTRIB, data=dweck.data),col.vars=1:2)

##        FAIL  1     2
##        DIFF  1  2  1  2
## ATTRIB
## 1            9  9  9 10
## 2           11 11  8 16
## 3           11 10  8  8

# Cell means
dweck.meanTable <- ftable(round(xtabs(Y~FAIL+DIFF+ATTRIB,
        aggregate(Y~ATTRIB+DIFF+FAIL,dweck.data,mean)),digits=2),col.vars=1:2)
dweck.meanTable

##        FAIL     1           2
##        DIFF     1     2     1     2
## ATTRIB
## 1           13.04 12.56 14.80 16.70
## 2           11.23 10.43 12.28 10.14
## 3           11.49  9.60  6.54  6.07

# Column means
apply(dweck.meanTable, 1, mean)

##     1     2     3
## 14.28 11.02  8.43

# Row means
apply(dweck.meanTable, 2, mean)

##  1_1  1_2  2_1  2_2
## 11.9 10.9 11.2 11.0
```

The regression models can be obtained from the various coding methods discussed in chapter 8.

```
# Regression models
# Dummy coded
contrasts(dweck.data$ATTRIB) <- contr.treatment(3, base=2)
contrasts(dweck.data$DIFF) <- contr.treatment(2, base=2)
contrasts(dweck.data$FAIL) <- contr.treatment(2, base=2)

dweckDum.fit <- lm(Y~ ATTRIB + DIFF + FAIL + ATTRIB:DIFF + ATTRIB:FAIL + DIFF:FAIL , data=dweck.data)
summary(dweckDum.fit)

##
## Call:
## lm(formula = Y ~ ATTRIB + DIFF + FAIL + ATTRIB:DIFF + ATTRIB:FAIL +
##     DIFF:FAIL, data = dweck.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -6.812 -1.485 -0.109  1.771  6.408
##
## Coefficients:
##               Estimate Std. Error t value             Pr(>|t|)
## (Intercept)     10.483      0.661   15.87 < 0.0000000000000002 ***
## ATTRIB1          5.827      1.005    5.80          0.000000066 ***
## ATTRIB3         -4.616      1.058   -4.36          0.000029007 ***
## DIFF1            1.104      0.999    1.10              0.27159
## FAIL1           -0.550      0.941   -0.58              0.56008
## ATTRIB1:DIFF1   -2.183      1.252   -1.74              0.08398 .
## ATTRIB3:DIFF1   -0.222      1.253   -0.18              0.85964
## ATTRIB1:FAIL1   -2.765      1.246   -2.22              0.02849 *
## ATTRIB3:FAIL1    4.447      1.253    3.55              0.00057 ***
## DIFF1:FAIL1      0.693      1.036    0.67              0.50469
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.79 on 110 degrees of freedom
## Multiple R-squared:  0.509,Adjusted R-squared:  0.469
## F-statistic: 12.7 on 9 and 110 DF,  p-value: 0.000000000000126

# Effects coded
contrasts(dweck.data$ATTRIB) <- contr.sum(3)
contrasts(dweck.data$DIFF) <- contr.sum(2)
contrasts(dweck.data$FAIL) <- contr.sum(2)

dweckEff.fit <- lm(Y~ ATTRIB + DIFF + FAIL + ATTRIB:DIFF + ATTRIB:FAIL + DIFF:FAIL , data=dweck.data)
summary(dweckEff.fit)

##
## Call:
## lm(formula = Y ~ ATTRIB + DIFF + FAIL + ATTRIB:DIFF + ATTRIB:FAIL +
##     DIFF:FAIL, data = dweck.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -6.812 -1.485 -0.109  1.771  6.408
##
## Coefficients:
##               Estimate Std. Error t value             Pr(>|t|)
## (Intercept)     11.216      0.259   43.26 < 0.0000000000000002 ***
## ATTRIB1          3.070      0.371    8.28     0.00000000000033 ***
## ATTRIB2         -0.283      0.355   -0.80                0.427
## DIFF1            0.324      0.259    1.25                0.213
## FAIL1             0.179      0.259    0.69               0.492
## ATTRIB1:DIFF1    -0.691      0.371   -1.86               0.065 .
## ATTRIB2:DIFF1     0.401      0.357    1.12               0.263
## ATTRIB1:FAIL1    -1.663      0.371   -4.49     0.00001792151052 ***
## ATTRIB2:FAIL1    -0.280      0.355   -0.79               0.432
## DIFF1:FAIL1       0.173      0.259    0.67               0.505
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.79 on 110 degrees of freedom
## Multiple R-squared:  0.509,Adjusted R-squared:  0.469
## F-statistic: 12.7 on 9 and 110 DF,  p-value: 0.000000000000126

# Weighted effects coded
a1 <- table(dweck.data$ATTRIB)[1]
a2 <- table(dweck.data$ATTRIB)[2]
a3 <- table(dweck.data$ATTRIB)[3]
d1 <- table(dweck.data$DIFF)[1]
d2 <- table(dweck.data$DIFF)[2]
f1 <- table(dweck.data$FAIL)[1]
f2 <- table(dweck.data$FAIL)[2]

attrib.weCod <- matrix(c(1, -a1/a2, 0, 0, -a3/a2, 1), ncol=2)
diff.weCod <- matrix(c(1, -d1/d2), ncol=1)
fail.weCod <- matrix(c(1, -f1/f2), ncol=1)

contrasts(dweck.data$ATTRIB) <- attrib.weCod
contrasts(dweck.data$DIFF) <- diff.weCod
contrasts(dweck.data$FAIL) <- fail.weCod

dweckWE.fit <- lm(Y~ ATTRIB + DIFF + FAIL + ATTRIB:DIFF + ATTRIB:FAIL + DIFF:FAIL , data=dweck.data)
summary(dweckWE.fit)

##
## Call:
## lm(formula = Y ~ ATTRIB + DIFF + FAIL + ATTRIB:DIFF + ATTRIB:FAIL +
##     DIFF:FAIL, data = dweck.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -6.812 -1.485 -0.109  1.771  6.408
##
## Coefficients:
##                Estimate Std. Error t value             Pr(>|t|)
## (Intercept)      11.174      0.258   43.38 < 0.0000000000000002 ***
## ATTRIB1           3.112      0.384    8.11      0.0000000000008 ***
## ATTRIB2          -2.750      0.386   -7.12      0.0000000001146 ***
## DIFF1             0.381      0.276    1.38               0.170
## FAIL1             0.144      0.253    0.57               0.572
## ATTRIB1:DIFF1    -0.769      0.410   -1.87               0.064 .
## ATTRIB2:DIFF1     0.277      0.411    0.67               0.501
## ATTRIB1:FAIL1    -1.615      0.378   -4.28      0.0000407330722 ***
## ATTRIB2:FAIL1     1.931      0.380    5.08      0.0000015303807 ***
## DIFF1:FAIL1       0.182      0.272    0.67               0.505
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.79 on 110 degrees of freedom
## Multiple R-squared:  0.509,Adjusted R-squared:  0.469
## F-statistic: 12.7 on 9 and 110 DF,  p-value: 0.000000000000126

# Contrast Coding
contCod <- matrix(c(.5, 0, -.5, -.5, 1, -.5), ncol=2)
```

```
contrasts(dweck.data$ATTRIB) <- contCod
contrasts(dweck.data$DIFF) <- c(.5,-.5)
contrasts(dweck.data$FAIL) <- c(.5,-.5)

dweckCont.fit <- lm(Y~ ATTRIB + DIFF + FAIL + ATTRIB:DIFF + ATTRIB:FAIL + DIFF:FAIL , data=dweck.data)
summary(dweckCont.fit)

##
## Call:
## lm(formula = Y ~ ATTRIB + DIFF + FAIL + ATTRIB:DIFF + ATTRIB:FAIL +
##     DIFF:FAIL, data = dweck.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -6.812 -1.485 -0.109  1.771  6.408
##
## Coefficients:
##               Estimate Std. Error t value             Pr(>|t|)
## (Intercept)     11.216      0.259   43.26 < 0.0000000000000002 ***
## ATTRIB1          5.856      0.653    8.97   0.0000000000000091 ***
## ATTRIB2         -0.283      0.355   -0.80                 0.43
## DIFF1            0.648      0.518    1.25                 0.21
## FAIL1            0.357      0.518    0.69                 0.49
## ATTRIB1:DIFF1   -1.961      1.303   -1.50                 0.14
## ATTRIB2:DIFF1    0.802      0.713    1.12                 0.26
## ATTRIB1:FAIL1   -7.212      1.307   -5.52   0.0000002281503023 ***
## ATTRIB2:FAIL1   -0.561      0.710   -0.79                 0.43
## DIFF1:FAIL1      0.693      1.036    0.67                 0.50
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.79 on 110 degrees of freedom
## Multiple R-squared:  0.509,Adjusted R-squared:  0.469
## F-statistic: 12.7 on 9 and 110 DF,  p-value: 0.000000000000126
```

To get the adjusted cell means from the regression with dummy codes, use the `lsmeans()` function.

```
lsmeans(dweckDum.fit,pairwise~ATTRIB + DIFF + FAIL + ATTRIB:DIFF + ATTRIB:FAIL + DIFF:FAIL)

## Error in eval(expr, envir, enclos):  could not find function "lsmeans"
```

9.3 Nominal Scale by Continuous Variable Interactions

Import the data.

```
# Salary data
salary.data <- read.table("C0904DT.txt", header=TRUE)
head(salary.data)

##   DEPART PUB TIME SALARY SEX
## 1      1  16    3  56465   1
## 2      1  25    7  92044   0
## 3      1  16    2  48980   1
## 4      1  24    1  53239   1
## 5      1  24    8  98948   0
## 6      1  26   14  64782   0
```

```
# Name the departments
salary.data$department <- factor(salary.data$DEPART, levels=1:3, labels=c("Psychology", "Sociology", "History")

# Center the PUBS variable
salary.data$pub.c <- salary.data$PUB - mean(salary.data$PUB)
```

Use the `describeBy()` function in the `psych` package to get the describe statistics by department.

```
library(psych)
describeBy(cbind(salary.data$SALARY,salary.data$PUB), salary.data$department)

## INDICES: Psychology
##    vars  n  mean       sd median trimmed     mad   min    max range skew kurtosis      se
## V1    1 60 61719 17589.31  58219 60842.5 17975.7 30833 103069 72236 0.41    -0.52 2270.77
## V2    2 60    19     8.08     19    18.6     8.9     4     39    35 0.26    -0.51    1.04
## ---------------------------------------------------------------------------
## INDICES: Sociology
##    vars  n    mean       sd median trimmed      mad   min    max range  skew kurtosis      se
## V1    1 44 66523.8 17530.14  67172 66386.9 20639.23 36446 102464 66019  0.12    -0.99 2642.77
## V2    2 44    15.2     5.61     15    15.3     5.93     4     26    22 -0.11    -0.84    0.85
## ---------------------------------------------------------------------------
## INDICES: History
##    vars  n    mean       sd median trimmed      mad   min    max range  skew kurtosis      se
## V1    1 46 64937.3 16001.40  64413 64777.0 17999.64 32995 108453 75458  0.15    -0.37 2359.28
## V2    2 46    11.2     5.96     11    11.3     7.41     1     23    22 -0.16    -1.10    0.88
```

9.3.1 Interactions of a continuous variable with dummy-variable coded groups

```
# Make sychology the reference group
contrasts(salary.data$department)  <- contr.treatment(3, base=1)


# Main Effects
salaryMainD.fit <- lm(SALARY~pub.c+department,data=salary.data)
summary(salaryMainD.fit)

##
## Call:
## lm(formula = SALARY ~ pub.c + department, data = salary.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -32072 -12315   -415  10880  47379
##
## Coefficients:
##             Estimate Std. Error t value             Pr(>|t|)
## (Intercept)    58482       2168   26.98 < 0.0000000000000002 ***
## pub.c            926        193    4.79            0.0000041 ***
## department2     8282       3249    2.55               0.0118 *
## department3    10447       3472    3.01               0.0031 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16000 on 146 degrees of freedom
## Multiple R-squared:  0.148,Adjusted R-squared:  0.131
## F-statistic: 8.46 on 3 and 146 DF,  p-value: 0.0000321

# Full Model
salaryFullD.fit <- lm(SALARY~pub.c*department,data=salary.data)
summary(salaryFullD.fit)
```

```
##
## Call:
## lm(formula = SALARY ~ pub.c * department, data = salary.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -32253 -10472   -417   9961  45235
##
## Coefficients:
##                     Estimate Std. Error t value             Pr(>|t|)
## (Intercept)            56918       2207   25.78 < 0.0000000000000002 ***
## pub.c                   1373        252    5.44           0.00000023 ***
## department2             9673       3235    2.99               0.0033 **
## department3             9796       3615    2.71               0.0076 **
## pub.c:department2      -1115        495   -2.25               0.0259 *
## pub.c:department3       -961        466   -2.06               0.0411 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15700 on 144 degrees of freedom
## Multiple R-squared:  0.189,Adjusted R-squared:  0.161
## F-statistic: 6.72 on 5 and 144 DF,  p-value: 0.0000117
```

To get the tolerance, use the `vif()` function in the `car` package.

```
library(car)
1/vif(salaryMainD.fit)

##            GVIF  Df GVIF^(1/(2*Df))
## pub.c      0.81 1.0           0.900
## department 0.81 0.5           0.949

1/vif(salaryFullD.fit)

##                   GVIF  Df GVIF^(1/(2*Df))
## pub.c            0.459 1.0           0.677
## department       0.659 0.5           0.901
## pub.c:department 0.412 0.5           0.801
```

The `plotSlopes()` function in the `rockchalk` package provides an easy way to plot a regression interaction.

```
library(rockchalk)
plotSlopes(salaryFullD.fit, plotx="pub.c", modx="department", xlab="Number of publications",
ylab="Salary", plotPoints=FALSE)
```

9.3.2 Interactions using weighted or unweighted effects codes

```
# Effects coding
effCod <- matrix(c(-1, 1,0, -1, 0, 1), ncol=2)
contrasts(salary.data$department)  <- effCod


# Main Effects
salaryMainE.fit <- lm(SALARY~pub.c+department,data=salary.data)
summary(salaryMainE.fit)

##
## Call:
```

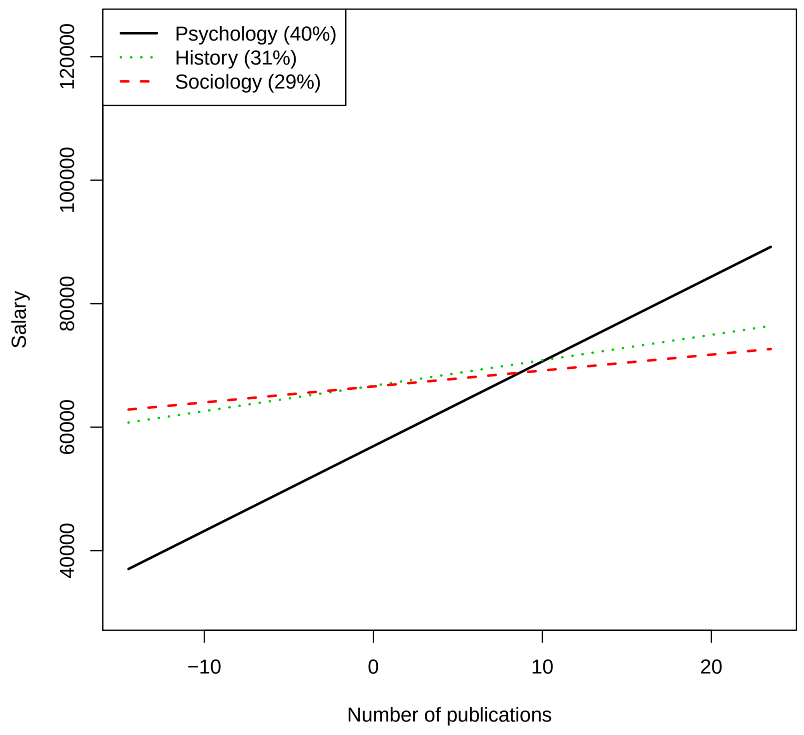


Figure 9.2: Slopes of salary on publications for three departments.

```
## lm(formula = SALARY ~ pub.c + department, data = salary.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -32072 -12315   -415  10880  47379
##
## Coefficients:
##             Estimate Std. Error t value            Pr(>|t|)
## (Intercept)    64725       1317   49.16 < 0.0000000000000002 ***
## pub.c            926        193    4.79           0.0000041 ***
## department1     2039       1912    1.07               0.288
## department2     4204       2039    2.06               0.041 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16000 on 146 degrees of freedom
## Multiple R-squared:  0.148,Adjusted R-squared:  0.131
## F-statistic: 8.46 on 3 and 146 DF,  p-value: 0.0000321

# Full Model
salaryFullE.fit <- lm(SALARY~pub.c*department,data=salary.data)
summary(salaryFullE.fit)

##
## Call:
## lm(formula = SALARY ~ pub.c * department, data = salary.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -32253 -10472   -417   9961  45235
##
## Coefficients:
##                   Estimate Std. Error t value            Pr(>|t|)
## (Intercept)          63408       1440    44.03 <0.0000000000000002 ***
## pub.c                  681        211     3.23              0.0015 **
## department1           3183       1984     1.60              0.1109
## department2           3306       2192     1.51              0.1337
## pub.c:department1     -423        324    -1.31              0.1936
## pub.c:department2     -269        309    -0.87              0.3857
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15700 on 144 degrees of freedom
## Multiple R-squared:  0.189,Adjusted R-squared:  0.161
## F-statistic: 6.72 on 5 and 144 DF,  p-value: 0.0000117
```

9.3.3 Interactions with a contrast-coded nominal scale

```
# Contrast coding
contCod <- matrix(c(2/3, -1/3, -1/3, 0, -.5, .5), ncol=2)
contrasts(salary.data$department)  <- contCod


# Main Effects
salaryMainC.fit <- lm(SALARY~pub.c+department,data=salary.data)
summary(salaryMainC.fit)

##
## Call:
## lm(formula = SALARY ~ pub.c + department, data = salary.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -32072 -12315   -415  10880  47379
##
## Coefficients:
##             Estimate Std. Error t value             Pr(>|t|)
## (Intercept)    64725       1317   49.16 < 0.0000000000000002 ***
## pub.c            926        193    4.79            0.0000041 ***
## department1    -9364       2885   -3.25               0.0015 **
## department2     2165       3454    0.63               0.5318
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16000 on 146 degrees of freedom
## Multiple R-squared:  0.148,Adjusted R-squared:  0.131
## F-statistic: 8.46 on 3 and 146 DF,  p-value: 0.0000321

# Full Model
salaryFullC.fit <- lm(SALARY~pub.c*department,data=salary.data)
summary(salaryFullC.fit)

##
## Call:
## lm(formula = SALARY ~ pub.c * department, data = salary.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -32253 -10472   -417   9961  45235
##
## Coefficients:
##                   Estimate Std. Error t value             Pr(>|t|)
## (Intercept)          63408       1440   44.03 < 0.0000000000000002 ***
## pub.c                  681        211    3.23              0.00152 **
## department1          -9734       2884   -3.37              0.00095 ***
## department2            123       3714    0.03              0.97356
## pub.c:department1     1038        384    2.70              0.00772 **
## pub.c:department2      154        579    0.27              0.79062
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15700 on 144 degrees of freedom
## Multiple R-squared:  0.189,Adjusted R-squared:  0.161
## F-statistic: 6.72 on 5 and 144 DF,  p-value: 0.0000117
```

9.3.4 Interactions coded to estimate simple slopes of groups

```
# Simple slopes coding
salary.data$d1 <- ifelse(salary.data$DEPART == "2", 1, 0)
salary.data$d2 <- ifelse(salary.data$DEPART == "3", 1, 0)
salary.data$sd0 <- ifelse(salary.data$DEPART == "1", salary.data$pub.c, 0)
salary.data$sd1 <- ifelse(salary.data$DEPART == "2", salary.data$pub.c, 0)
salary.data$sd2 <- ifelse(salary.data$DEPART == "3", salary.data$pub.c, 0)


# Main Effects
salaryMainSimp.fit <- lm(SALARY~pub.c+d1+d2,data=salary.data)
summary(salaryMainSimp.fit)

##
## Call:
## lm(formula = SALARY ~ pub.c + d1 + d2, data = salary.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -32072 -12315   -415  10880  47379
##
## Coefficients:
##             Estimate Std. Error t value             Pr(>|t|)
## (Intercept)    58482       2168   26.98 < 0.0000000000000002 ***
## pub.c            926        193    4.79            0.0000041 ***
## d1              8282       3249    2.55               0.0118 *
## d2             10447       3472    3.01               0.0031 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16000 on 146 degrees of freedom
## Multiple R-squared:  0.148,Adjusted R-squared:  0.131
## F-statistic: 8.46 on 3 and 146 DF,  p-value: 0.0000321

# Full Model
salaryFullSimp.fit <- lm(SALARY~d1+d2+sd0+sd1+sd2,data=salary.data)
summary(salaryFullSimp.fit)

##
## Call:
## lm(formula = SALARY ~ d1 + d2 + sd0 + sd1 + sd2, data = salary.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -32253 -10472   -417   9961  45235
##
## Coefficients:
```

```
##             Estimate Std. Error t value             Pr(>|t|)
## (Intercept)    56918       2207   25.78 < 0.0000000000000002 ***
## d1              9673       3235    2.99               0.0033 **
## d2              9796       3615    2.71               0.0076 **
## sd0             1373        252    5.44           0.00000023 ***
## sd1              258        426    0.61               0.5461
## sd2              412        392    1.05               0.2950
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15700 on 144 degrees of freedom
## Multiple R-squared:  0.189,Adjusted R-squared:  0.161
## F-statistic: 6.72 on 5 and 144 DF,  p-value: 0.0000117
```

To estimate salaries at different levels of publications, just define new variables with different cutoff values.

```
salary.data$pub.20 <- salary.data$PUB-20
salary.data$pub.10 <- salary.data$PUB-10
```

Then use those new variables in the model.

```
salaryMainSimpPub20.fit <- lm(SALARY~pub.20+d1+d2+pub.20:d1+pub.20:d2,data=salary.data)
summary(salaryMainSimpPub20.fit)

##
## Call:
## lm(formula = SALARY ~ pub.20 + d1 + d2 + pub.20:d1 + pub.20:d2,
##     data = salary.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -32253 -10472   -417   9961  45235
##
## Coefficients:
##             Estimate Std. Error t value             Pr(>|t|)
## (Intercept)    63114       2039   30.95 < 0.0000000000000002 ***
## pub.20          1373        252    5.44           0.00000023 ***
## d1              4640       3726    1.25                0.215
## d2              5459       4633    1.18                0.241
## pub.20:d1      -1115        495   -2.25                0.026 *
## pub.20:d2       -961        466   -2.06                0.041 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15700 on 144 degrees of freedom
## Multiple R-squared:  0.189,Adjusted R-squared:  0.161
## F-statistic: 6.72 on 5 and 144 DF,  p-value: 0.0000117

salaryMainSimpPub10.fit <- lm(SALARY~pub.10+d1+d2+pub.10:d1+pub.10:d2,data=salary.data)
summary(salaryMainSimpPub10.fit)

##
## Call:
## lm(formula = SALARY ~ pub.10 + d1 + d2 + pub.10:d1 + pub.10:d2,
##     data = salary.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -32253 -10472   -417   9961  45235
##
## Coefficients:
```

```
##             Estimate Std. Error t value             Pr(>|t|)
## (Intercept)    49385       3039   16.25 < 0.0000000000000002 ***
## pub.10          1373        252    5.44           0.00000023 ***
## d1             15790       4448    3.55              0.00052 ***
## d2             15069       3846    3.92              0.00014 ***
## pub.10:d1      -1115        495   -2.25              0.02592 *
## pub.10:d2       -961        466   -2.06              0.04109 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15700 on 144 degrees of freedom
## Multiple R-squared:  0.189,Adjusted R-squared:  0.161
## F-statistic: 6.72 on 5 and 144 DF,  p-value: 0.0000117
```

9.3.5 Categorical variable interactions with nonlinear effects of scaled independent variables

Import the data.

```
# Seniority data
senior.data <- read.table("C0907DT.txt", header=TRUE)
head(senior.data)

##   DEPART PUB TIME SALARYX SEX
## 1      1  16    3   56465   1
## 2      1  25    7   92044   0
## 3      1  16    2   48980   1
## 4      1  24    1   53239   1
## 5      1  24    8   98948   0
## 6      1  26   14   64782   0
```

```
# Center the time variable
senior.data$time.c <- senior.data$TIME - mean(senior.data$TIME)
# Create the quadratic effect
senior.data$time2 <- senior.data$time.c*senior.data$time.c
# Make the DEPART variable a factor
senior.data$department <- factor(senior.data$DEPART, levels=1:3, labels=c("Psychology", "Sociology", "History")
```

```
# Regression models

# Psychology only
senior.psychFit <- lm(SALARYX~time.c+time2, data=senior.data,subset=department=="Psychology")
summary(senior.psychFit)

##
## Call:
## lm(formula = SALARYX ~ time.c + time2, data = senior.data, subset = department ==
##     "Psychology")
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -25341 -10500  -1646   8412  33791
##
## Coefficients:
##             Estimate Std. Error t value             Pr(>|t|)
## (Intercept)  65326.4     2587.8   25.24 < 0.0000000000000002 ***
## time.c        1945.3      396.7    4.90            0.0000082 ***
## time2          -45.6       49.9   -0.91                 0.36
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14900 on 57 degrees of freedom
## Multiple R-squared:  0.311,Adjusted R-squared:  0.287
## F-statistic: 12.9 on 2 and 57 DF,  p-value: 0.0000244

# Sociology only
senior.socFit <- lm(SALARYX~time.c+time2, data=senior.data,subset=department=="Sociology")
summary(senior.socFit)

##
## Call:
## lm(formula = SALARYX ~ time.c + time2, data = senior.data, subset = department ==
##     "Sociology")
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -28269 -12842   1215   6616  34365
##
## Coefficients:
##             Estimate Std. Error t value            Pr(>|t|)
## (Intercept)    63246       3594   17.60 <0.0000000000000002 ***
## time.c           841        528    1.59                0.12
## time2            122        103    1.19                0.24
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16900 on 41 degrees of freedom
## Multiple R-squared:  0.111,Adjusted R-squared:  0.068
## F-statistic: 2.57 on 2 and 41 DF,  p-value: 0.0889

# History only
senior.histFit <- lm(SALARYX~time.c+time2, data=senior.data,subset=department=="History")
summary(senior.histFit)

##
## Call:
## lm(formula = SALARYX ~ time.c + time2, data = senior.data, subset = department ==
##     "History")
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -25672 -11650   -167  11829  36543
##
## Coefficients:
##             Estimate Std. Error t value            Pr(>|t|)
## (Intercept)  64529.4     2819.7   22.89 <0.0000000000000002 ***
## time.c        1606.8      492.4    3.26              0.0022 **
## time2          -60.7       78.2   -0.78              0.4421
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14500 on 43 degrees of freedom
## Multiple R-squared:  0.21,Adjusted R-squared:  0.173
## F-statistic: 5.72 on 2 and 43 DF,  p-value: 0.00628
```

```
# Make psychology the reference group
contrasts(senior.data$department)  <- contr.treatment(3, base=1)

# Hierarchical Regression
# Main effects
senior1.fit <- lm(SALARYX~department+time.c, data=senior.data)
```

```
summary(senior1.fit)

##
## Call:
## lm(formula = SALARYX ~ department + time.c, data = senior.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -31884 -11408  -1063  11214  42056
##
## Coefficients:
##             Estimate Std. Error t value             Pr(>|t|)
## (Intercept)    63406       2006   31.60 < 0.0000000000000002 ***
## department2     2712       3074    0.88                 0.38
## department3     -283       3072   -0.09                 0.93
## time.c          1463        245    5.97          0.000000018 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15400 on 146 degrees of freedom
## Multiple R-squared:  0.208,Adjusted R-squared:  0.191
## F-statistic: 12.7 on 3 and 146 DF,  p-value: 0.00000019

# Main effects + time^2
senior2.fit <- lm(SALARYX~department+time.c+time2, data=senior.data)
summary(senior2.fit)

##
## Call:
## lm(formula = SALARYX ~ department + time.c + time2, data = senior.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -32166 -11487  -1213  11275  42553
##
## Coefficients:
##             Estimate Std. Error t value             Pr(>|t|)
## (Intercept)  63770.3     2395.9   26.62 < 0.0000000000000002 ***
## department2   2614.3     3103.5    0.84                 0.40
## department3   -395.1     3107.8   -0.13                 0.90
## time.c        1492.9      267.7    5.58           0.00000012 ***
## time2          -11.0       39.4   -0.28                 0.78
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15400 on 145 degrees of freedom
## Multiple R-squared:  0.208,Adjusted R-squared:  0.186
## F-statistic: 9.52 on 4 and 145 DF,  p-value: 0.00000073

# Main effects + time^2 + time x dept interactions
senior3.fit <- lm(SALARYX~department+time.c+time2+department:time.c, data=senior.data)
summary(senior3.fit)

##
## Call:
## lm(formula = SALARYX ~ department + time.c + time2 + department:time.c,
##     data = senior.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -31546 -11200   -204  11133  39213
##
## Coefficients:
```

```
##                     Estimate Std. Error t value              Pr(>|t|)
## (Intercept)          64423.0     2443.8   26.36 < 0.0000000000000002 ***
## department2           2290.6     3118.9    0.73                  0.46
## department3           -828.0     3141.6   -0.26                  0.79
## time.c                1854.7      396.7    4.68             0.0000067 ***
## time2                  -18.9       39.9   -0.47                  0.64
## department2:time.c    -842.6      604.4   -1.39                  0.17
## department3:time.c    -373.3      591.8   -0.63                  0.53
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15400 on 143 degrees of freedom
## Multiple R-squared:  0.219,Adjusted R-squared:  0.186
## F-statistic: 6.67 on 6 and 143 DF,  p-value: 0.00000306

# Main effects + time^2 + time x dept interactions + time^2 x dept interactions
senior4.fit <- lm(SALARYX~department+time.c+time2+department:time.c+department:time2,
        data=senior.data)
summary(senior4.fit)

##
## Call:
## lm(formula = SALARYX ~ department + time.c + time2 + department:time.c +
##     department:time2, data = senior.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -28269 -11822   -183   8848  36543
##
## Coefficients:
##                     Estimate Std. Error t value              Pr(>|t|)
## (Intercept)          65326.4     2682.2   24.36 < 0.0000000000000002 ***
## department2          -2080.5     4229.1   -0.49                 0.624
## department3           -797.1     4012.2   -0.20                 0.843
## time.c                1945.3      411.2    4.73             0.0000054 ***
## time2                  -45.6       51.7   -0.88                 0.379
## department2:time.c   -1104.3      632.0   -1.75                 0.083 .
## department3:time.c    -338.5      663.8   -0.51                 0.611
## department2:time2      168.2      107.1    1.57                 0.119
## department3:time2      -15.0       97.6   -0.15                 0.878
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15400 on 141 degrees of freedom
## Multiple R-squared:  0.234,Adjusted R-squared:  0.19
## F-statistic: 5.38 on 8 and 141 DF,  p-value: 0.00000651
```

A plot of the quadratic interaction is given in Figure 9.3.

```
# Plot the quadratic interactions
par(mar=par()$mar+c(0,0,0,7))
plot(senior.data$time.c, senior.data$SALARYX, ylab="Salary",
xlab="Seniority (Mean Centered)", col="gray")
curve (cbind (1,0, 0,x, x*x, 0*x,0*x, 0*x*x,0*x*x) %*% coef(senior4.fit), add=TRUE, col="red4", lwd=3)
curve (cbind (1,1, 0,x, x*x, 1*x,0*x, 1*x*x,0*x*x) %*% coef(senior4.fit), add=TRUE, col="blue4", lwd=3)
curve (cbind (1,0, 1,x, x*x, 0*x,1*x, 0*x*x,1*x*x) %*% coef(senior4.fit), add=TRUE, col="green4", lwd=3)

legend(max(senior.data$time.c)+1, mean(senior.data$SALARYX),c("Psychology", "Sociology", "History"),
title="Department", lwd = 3, col=c("red4", "blue4", "green4"), xpd=TRUE)
```

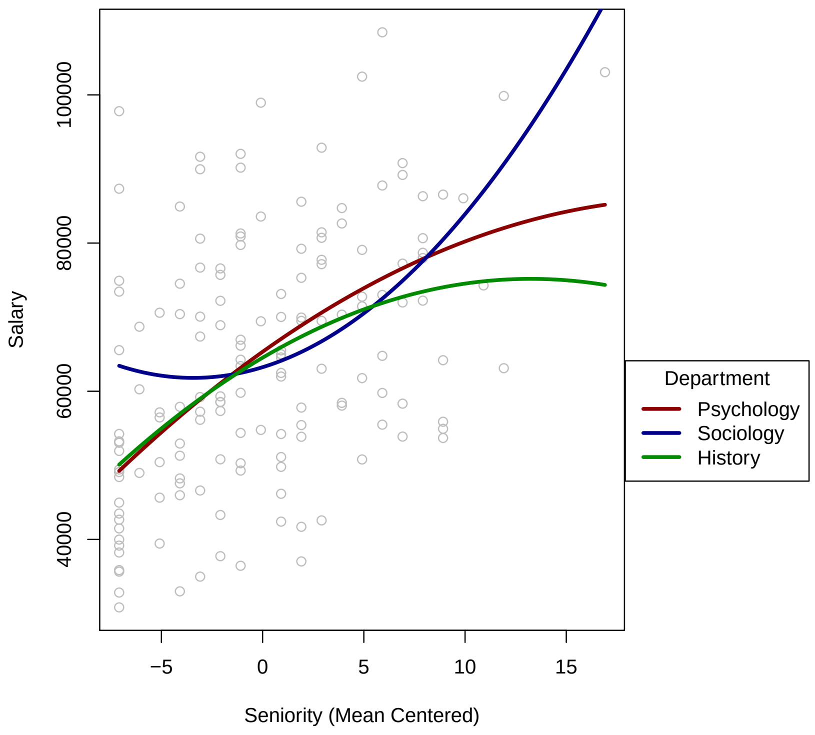


Figure 9.3: Quadratic slopes of salary on seniority.

Chapter 10

Outliers and multicollinearity: Diagnosing and solving regression problems II

10.1 Outliers

Import both PhD-Publications datasets given in CCAW Table 10.2.1

```r
#PhD/Publication data (Table 10.2.1)
phd1.data<-read.table("C10e01dt1.txt")
phd2.data<-read.table("C10e01dt2.txt")
names(phd1.data) <- c("CASE", "Year", "Pub")
names(phd2.data) <- c("CASE", "Year", "Pub")
```

Show the two dataset concurrently using the `cbind()` function. Table 10.1 compares the two datasets.

```r
cbind(phd1.data, phd2.data)
```

Table 10.1: Years Since Ph.D. and Number of Publications: Data

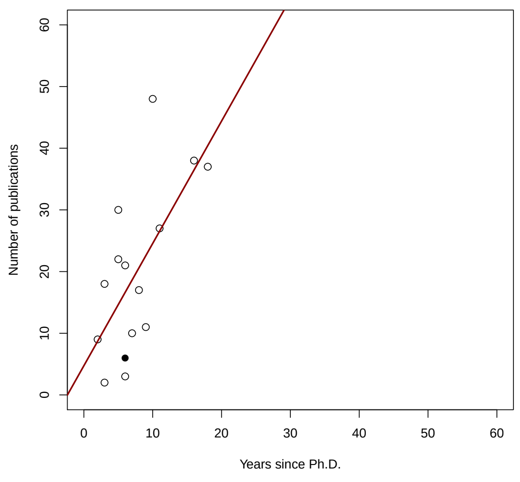
	CASE	Year	Pub	CASE	Year	Pub
1	1	3	18	1	3	18
2	2	6	3	2	6	3
3	3	3	2	3	3	2
4	4	8	17	4	8	17
5	5	9	11	5	9	11
6	6	6	6	6	60	6
7	7	16	38	7	16	38
8	8	10	48	8	10	48
9	9	2	9	9	2	9
10	10	5	22	10	5	22
11	11	5	30	11	5	30
12	12	6	21	12	6	21
13	13	7	10	13	7	10
14	14	11	27	14	11	27
15	15	18	37	15	18	37

We use the same regression model to fit both datasets.

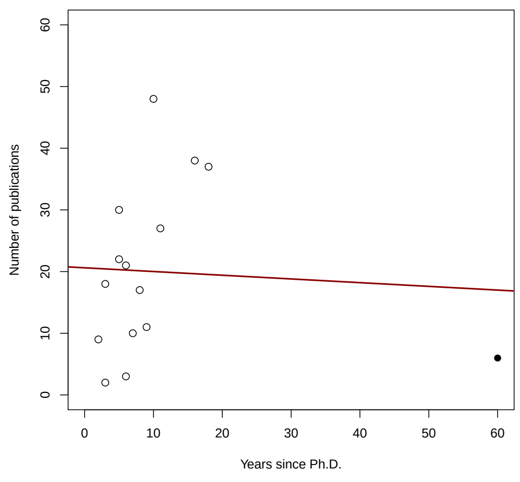
```
# No outlier
reg.fit1<-lm(Pub~Year, data=phd1.data)
# Outlier
reg.fit2<-lm(Pub~Year, data=phd2.data)
# Remove outlier
reg.fit3<-lm(Pub~Year, data=phd2.data[-6,])
summary(reg.fit1); summary(reg.fit2); summary(reg.fit3)

##
## Call:
## lm(formula = Pub ~ Year, data = phd1.data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -13.628  -8.645   0.303   5.846  23.440
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    4.731      5.591    0.85   0.4128
## Year           1.983      0.632    3.14   0.0078 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.8 on 13 degrees of freedom
## Multiple R-squared:  0.431,Adjusted R-squared:  0.387
## F-statistic: 9.85 on 1 and 13 DF,  p-value: 0.00783
##
## Call:
## lm(formula = Pub ~ Year, data = phd2.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -18.43 -10.59  -2.43   8.37  27.99
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  20.6119     4.7794    4.31  0.00084 ***
## Year         -0.0602     0.2689   -0.22  0.82626
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14.3 on 13 degrees of freedom
## Multiple R-squared:  0.00384,Adjusted R-squared:  -0.0728
## F-statistic: 0.0502 on 1 and 13 DF,  p-value: 0.826
##
## Call:
## lm(formula = Pub ~ Year, data = phd2.data[-6, ])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -14.504  -8.151  -0.463   5.560  22.825
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    5.999      5.716    1.05    0.315
## Year           1.918      0.634    3.03    0.011 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.8 on 12 degrees of freedom
## Multiple R-squared:  0.433,Adjusted R-squared:  0.386
## F-statistic: 9.16 on 1 and 12 DF,  p-value: 0.0105
```

Use the `plot()` function for quick scatterplots; the `pch` argument allows you to change the scatterplot symbols. The `abline()` allows you to add a regression line to the plot. The results are shown in Figure 10.1.

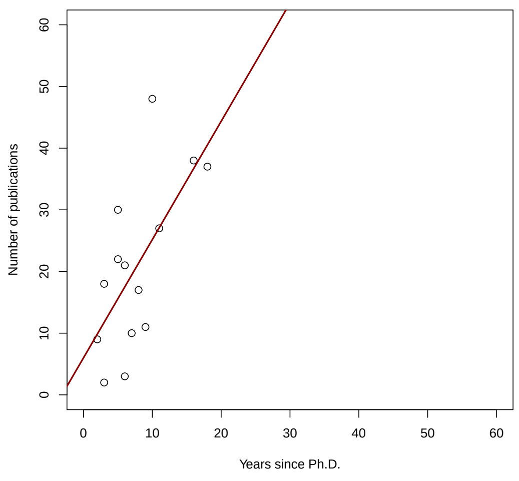
```
# Define the scatterplot symbols
pchs <- c(1,16)

# Scatterplot without outlier
plot(phd1.data$Year, phd1.data$Pub, xlim=c(0,60),ylim=c(0,60), xlab="Years since Ph.D.",
ylab="Number of publications", pch=pchs[as.numeric(phd1.data$CASE==6)+1],cex=1.2)
abline(lm(Pub~Year, data=phd1.data), col="red4", lwd=2)
# Scatterplot with outlier
plot(phd2.data$Year, phd2.data$Pub, xlim=c(0,60), ylim=c(0,60), xlab="Years since Ph.D.",
ylab="Number of publications",
pch=pchs[as.numeric(phd1.data$CASE==6)+1],cex=1.2)
abline(lm(Pub~Year, data=phd2.data), col="red4", lwd=2)
```



(a) Original data



(b) Data containing outlier



(c) Data deleting outlier

Figure 10.1: Plot of years since Ph.D. vs. number of publications (case 6 is black circle)

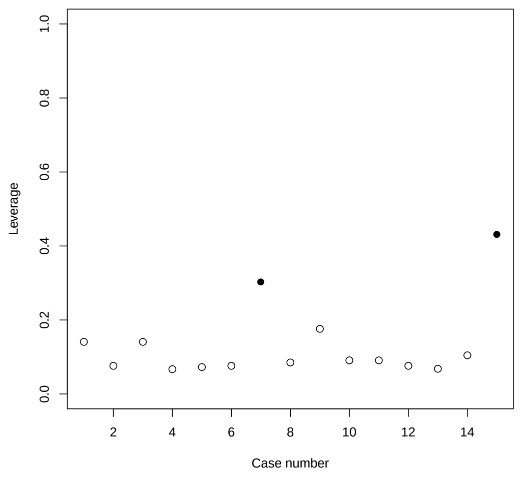
10.1.1 Detecting Outliers: Regression Diagnostics

10.1.1.1 Leverage

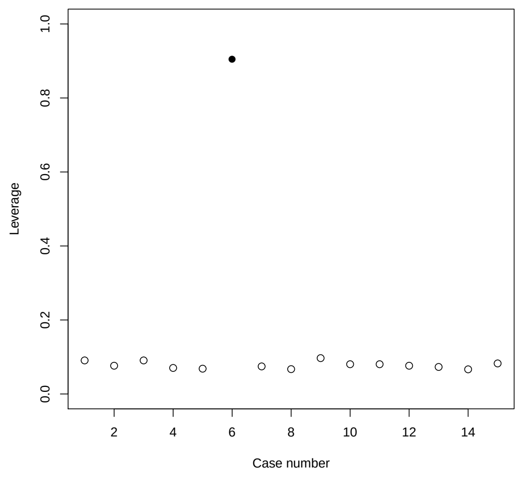
Leverage values can be calculated by using the `hatvalues()` function.

```
hatvalues(reg.fit1)

##      1      2      3      4      5      6      7      8      9     10     11     12     13     14
## 0.1409 0.0761 0.1409 0.0670 0.0727 0.0761 0.3034 0.0852 0.1761 0.0909 0.0909 0.0761 0.0682 0.1045
##     15
## 0.4307

hatvalues(reg.fit2)

##      1      2      3      4      5      6      7      8      9     10     11     12     13     14
## 0.0908 0.0765 0.0908 0.0704 0.0685 0.9044 0.0746 0.0672 0.0970 0.0805 0.0805 0.0765 0.0731 0.0667
##     15
## 0.0827
```

Use the `plot()` function for index plots. The results are given in Figure 10.2



(a) Original data



(b) Data containing outlier

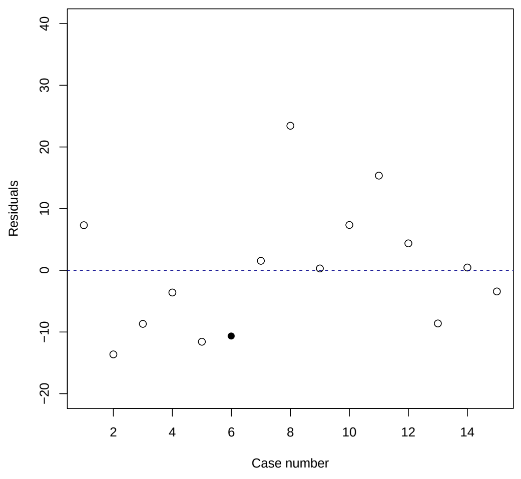
Figure 10.2: Index plot of leverage vs. case number

10.1.1.2 Discrepancy

Scatterplots of residuals for each case can be obtained by using the `plot()` and `resid()` functions. The result is given in Figure 10.3.

```
# Residuals plot without outlier
plot(resid(reg.fit1), xlab="Case number", ylab="Residuals", ylim=c(-20,40), cex=1.2,
pch=pchs[as.numeric(phd1.data$CASE==6)+1])
abline(h=0, col="blue4", lty=2)
# Residuals plot witout outlier
plot(resid(reg.fit2), xlab="Case number", ylab="Residuals", ylim=c(-20,40), cex=1.2,
pch=pchs[as.numeric(phd2.data$CASE==6)+1])
abline(h=0, col="blue4", lty=2)
```

Externally studentized residuals can be calculated by using the `rstudent()` function.


(a) Original data

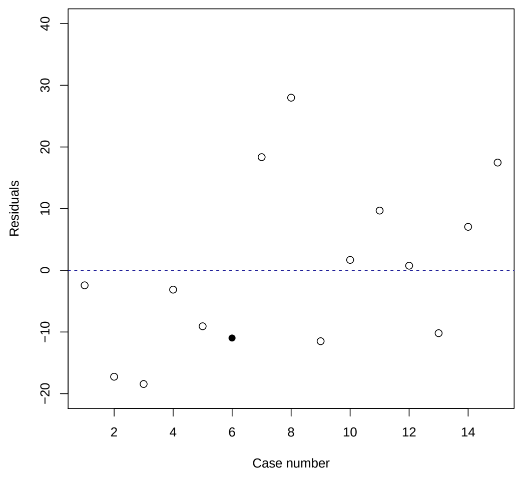

(b) Data containing outlier

Figure 10.3: Index Plot of residual vs. case number

```
rstudent(reg.fit1)

##       1       2       3       4       5       6       7       8       9      10      11      12
##  0.7162 -1.3517 -0.8567 -0.3320 -1.1224 -1.0240  0.1643  2.7976  0.0297  0.6988  1.5702  0.4067
##      13      14      15
## -0.8138  0.0429 -0.4057

rstudent(reg.fit2)

##       1       2       3       4       5       6       7       8       9      10      11      12
## -0.1713 -1.2848 -1.3989 -0.2183 -0.6413 -3.2943  1.3777  2.3503 -0.8347  0.1183  0.6915  0.0523
##      13      14      15
## -0.7257  0.4947  1.3086
```

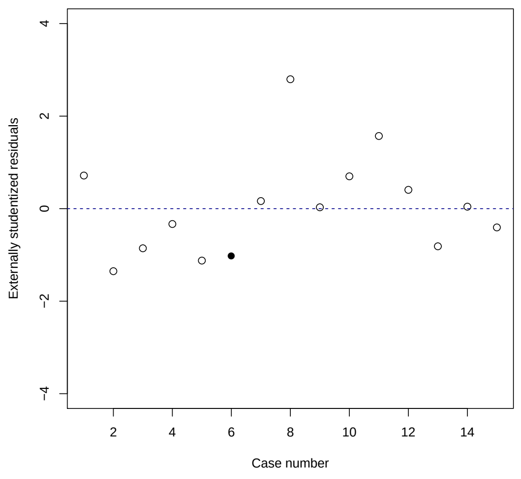
Scatterplots of the studentized residuals are given in Figure 10.4.

```
# Studentized residuals plot without outlier
plot(rstudent(reg.fit1), xlab="Case number", ylab="Externally studentized residuals", ylim=c(-4,4),
cex=1.2,
pch=pchs[as.numeric(phd1.data$CASE==6)+1])
abline(h=0, col="blue4", lty=2)
# Studentized residuals plot with outlier
plot(resid(reg.fit2), xlab="Case number", ylab="Externally studentized residuals", ylim=c(-4,4),
cex=1.2,
pch=pchs[as.numeric(phd2.data$CASE==6)+1])
abline(h=0, col="blue4", lty=2)
```

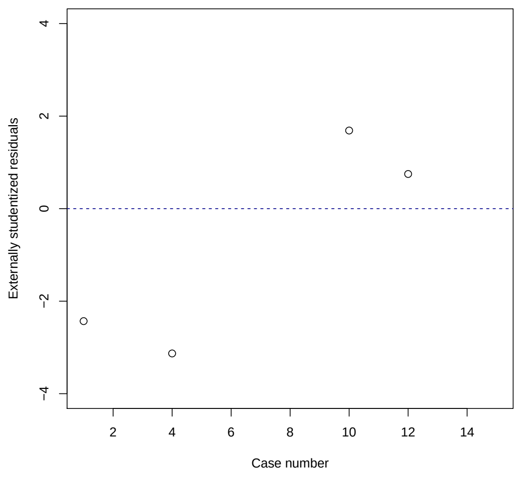
10.1.1.3 Influence

Differences in fit (DFFITS) values can be calculated by using the `dffits()` function.

```
dffits(reg.fit1)

##       1       2       3       4       5       6       7       8       9      10      11      12
##  0.2901 -0.3880 -0.3470 -0.0890 -0.3143 -0.2940  0.1084  0.8539  0.0137  0.2210  0.4966  0.1167
##      13      14      15
## -0.2201  0.0147 -0.3529

dffits(reg.fit2)
```



(a) Original data



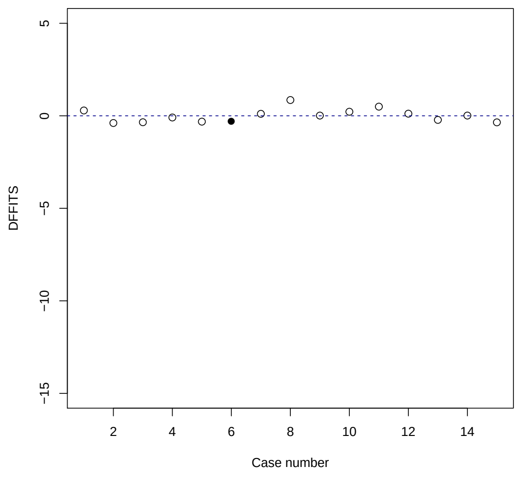
(b) Data containing outlier

Figure 10.4: Index Plot of externally studentized residual vs. case number

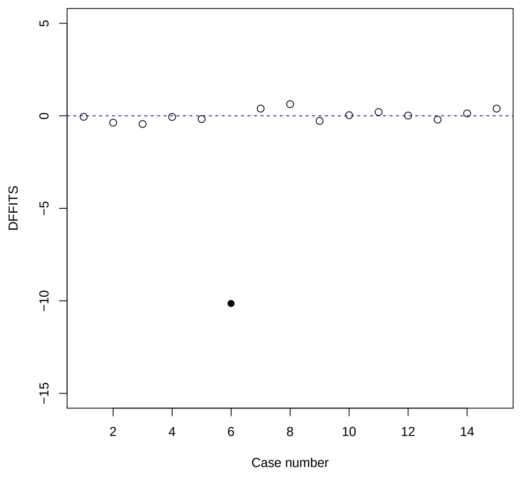
```
##          1         2         3         4         5         6         7         8         9        10        11
##    -0.0541   -0.3696   -0.4420   -0.0601   -0.1739  -10.1328    0.3911    0.6310   -0.2735    0.0350    0.2046
##         12        13        14        15
##     0.0151   -0.2038    0.1322    0.3928
```

Scatterplots of the DFFITS are given in Figure 10.5.

```r
# Studentized residuals plot without outlier
plot(dffits(reg.fit1), xlab="Case number", ylab="DFFITS", ylim=c(-15,5),
cex=1.2,
pch=pchs[as.numeric(phd1.data$CASE==6)+1])
abline(h=0, col="blue4", lty=2)
# Studentized residuals plot with outlier
plot(dffits(reg.fit2), xlab="Case number", ylab="DFFITS", ylim=c(-15,5),
cex=1.2,
pch=pchs[as.numeric(phd2.data$CASE==6)+1])
abline(h=0, col="blue4", lty=2)
```



(a) Original data



(b) Data containing outlier

Figure 10.5: Index Plot of DFFITS vs. case number

For Cook's (1977) D values, use the `cooks.distance()` function.

```
cooks.distance(reg.fit1)

##        1        2        3        4        5        6        7        8        9       10       11
## 0.043709 0.070779 0.061446 0.004251 0.048432 0.043048 0.006352 0.239054 0.000102 0.025418 0.110791
##       12       13       14       15
## 0.007283 0.024876 0.000116 0.066545

cooks.distance(reg.fit2)

##         1         2         3         4         5         6         7         8         9        10
##  0.001583  0.065064  0.090989  0.001948  0.015836 29.203927  0.071532  0.147681  0.038299  0.000663
##        11        12        13        14        15
##  0.021811  0.000123  0.021548  0.009284  0.073147
```

Assess the fit for each regression coefficient using DFBETA and standardized DFBETA (DFBETAS). In **R**, use the `dfbeta()` and `dfbetas()` functions, respectively. The results will contain a column for each term in the model, including the intercept.

```
# Unstandardized DFBETA
dfbeta(reg.fit1)

##    (Intercept)     Year
## 1       1.6074 -0.13556
## 2      -1.6260  0.08382
## 3      -1.9058  0.16073
## 4      -0.2233 -0.00438
## 5      -0.3973 -0.05675
## 6      -1.2681  0.06537
## 7      -0.3346  0.06289
## 8       0.1456  0.20382
## 9       0.0791 -0.00711
## 10      1.1032 -0.07355
## 11      2.3032 -0.15355
## 12      0.5216 -0.02689
## 13     -0.7771  0.02100
## 14     -0.0104  0.00580
## 15      1.2233 -0.21186

dfbeta(reg.fit2)

##    (Intercept)      Year
## 1      -0.2661  0.007797
## 2      -1.6362  0.034700
## 3      -2.0174  0.059111
## 4      -0.2682  0.003880
## 5      -0.7368  0.007785
## 6      14.6131 -1.977814
## 7       0.9490  0.033110
## 8       2.1516 -0.013408
## 9      -1.3170  0.041595
## 10      0.1682 -0.004061
## 11      0.9650 -0.023294
## 12      0.0711 -0.001508
## 13     -0.9193  0.016546
## 14      0.5116 -0.000711
## 15      0.7601  0.045238

# Standardized DFBETA
dfbetas(reg.fit1)
```

```
##    (Intercept)     Year
## 1      0.28208 -0.21056
## 2     -0.29996  0.13685
## 3     -0.33739  0.25184
## 4     -0.03855 -0.00669
## 5     -0.07176 -0.09074
## 6     -0.22725  0.10367
## 7     -0.05756  0.09576
## 8      0.03216  0.39849
## 9      0.01359 -0.01082
## 10     0.19341 -0.11412
## 11     0.43458 -0.25642
## 12     0.09025 -0.04117
## 13    -0.13719  0.03282
## 14    -0.00179  0.00882
## 15     0.21167 -0.32444
```

```
dfbetas(reg.fit2)
```

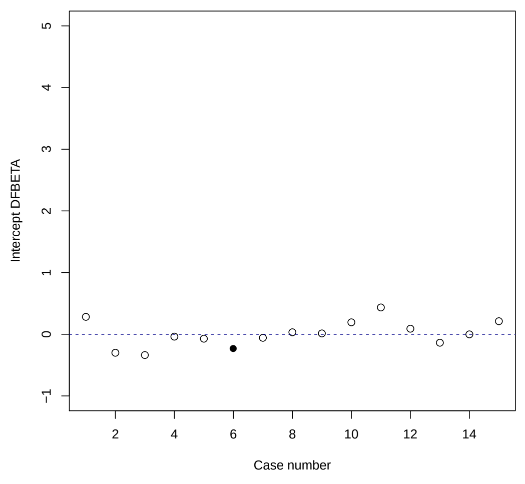
```
##    (Intercept)     Year
## 1      -0.0536  0.02789
## 2      -0.3508  0.13224
## 3      -0.4374  0.22778
## 4      -0.0540  0.01389
## 5      -0.1506  0.02829
## 6       4.0538 -9.75216
## 7       0.2053  0.12732
## 8       0.5227 -0.05789
## 9      -0.2723  0.15288
## 10      0.0338 -0.01452
## 11      0.1978 -0.08487
## 12      0.0143 -0.00539
## 13     -0.1888  0.06040
## 14      0.1039 -0.00256
## 15      0.1633  0.17279
```

Scatterplots of the DFBETAS are given in Figure 10.6.

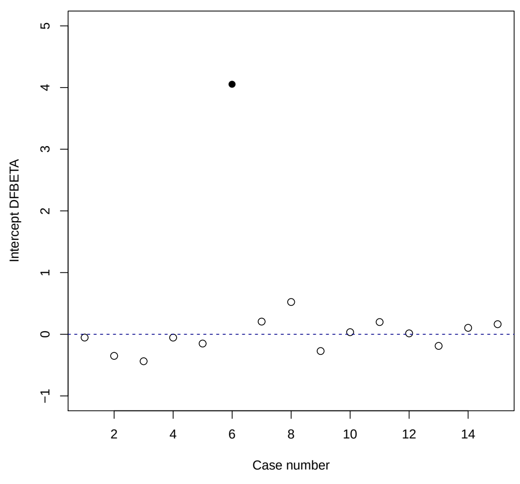
```
# DFBETAS for intercept without outliers
plot(dfbetas(reg.fit1)[,1], xlab="Case number", ylab="Intercept DFBETA", ylim=c(-1,5),
cex=1.2, pch=pchs[as.numeric(phd1.data$CASE==6)+1])
abline(h=0, col="blue4", lty=2)
# DFBETAS for intercept with outlier
plot(dfbetas(reg.fit2)[,1], xlab="Case number", ylab="Intercept DFBETA",  ylim=c(-1,5),
cex=1.2, pch=pchs[as.numeric(phd2.data$CASE==6)+1])
abline(h=0, col="blue4", lty=2)
# DFBETAS for slope without outliers
plot(dfbetas(reg.fit1)[,2], xlab="Case number", ylab="Slope DFBETA", ylim=c(-10,2),
cex=1.2, pch=pchs[as.numeric(phd1.data$CASE==6)+1])
abline(h=0, col="blue4", lty=2)
# DFBETAS for slope with outlier
plot(dfbetas(reg.fit2)[,2], xlab="Case number", ylab="Slope DFBETA",  ylim=c(-10,2),
cex=1.2, pch=pchs[as.numeric(phd2.data$CASE==6)+1])
abline(h=0, col="blue4", lty=2)
```

10.1.1.4 Location of outlying points and diagnostic statistics

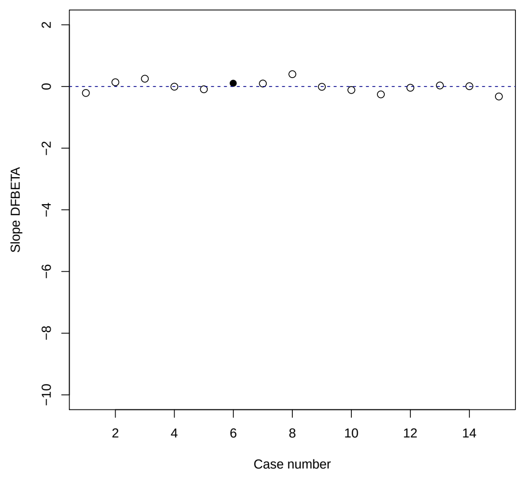
CCAW use four different datasets (each a slight alteration of case 6 in the original Phd-Publications data). So we have import each dataset.



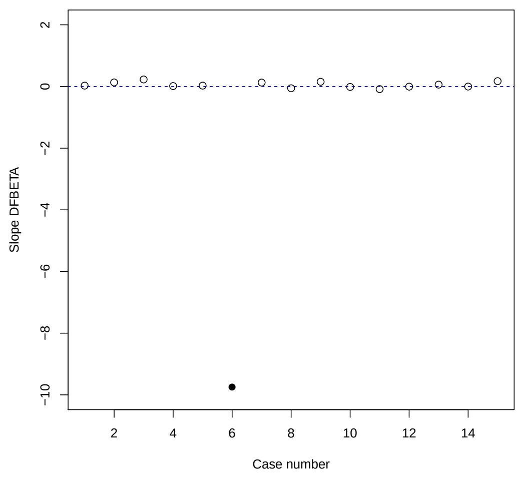
(a) Original data



(b) Data containing outlier



(c) Original data



(d) Data containing outlier

Figure 10.6: Index plot of DFBETAs vs. case number

```
#PhD/Publication data with various outliers
out1.data<-read.table("C10e02dt1.txt")
out2.data<-read.table("C10e02dt2.txt")
out3.data<-read.table("C10e02dt3.txt")
out4.data<-read.table("C10e02dt4.txt")
names(out1.data) <- names(out2.data) <- names(out3.data) <- names(out4.data) <- c("CASE", "Year", "Pub")
```

Scatterplots of the datasets are given in Figure 10.7.

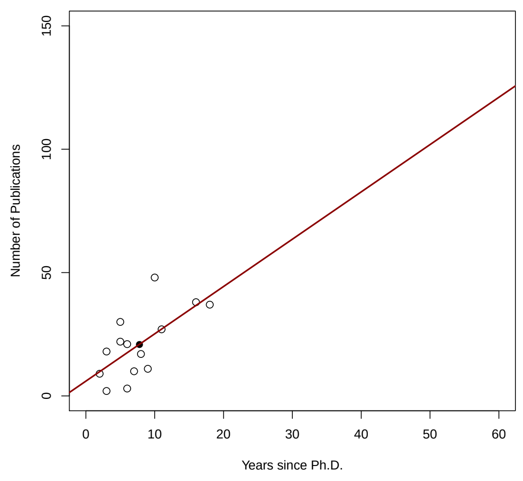
```
# Dataset 1
plot(out1.data$Year, out1.data$Pub, xlab="Years since Ph.D.", ylab="Number of Publications",
ylim=c(0,150), xlim=c(0,60),cex=1.2, pch=pchs[as.numeric(out1.data$CASE==6)+1])
abline(lm(Pub~Year,data=out1.data), col="red4", lwd=2)
# Dataset 2
plot(out2.data$Year, out2.data$Pub, xlab="Years since Ph.D.", ylab="Number of Publications",
ylim=c(0,150), xlim=c(0,60),cex=1.2, pch=pchs[as.numeric(out2.data$CASE==6)+1])
abline(lm(Pub~Year,data=out2.data), col="red4", lwd=2)
# Dataset 2
plot(out3.data$Year, out3.data$Pub, xlab="Years since Ph.D.", ylab="Number of Publications",
ylim=c(0,150), xlim=c(0,60),cex=1.2, pch=pchs[as.numeric(out3.data$CASE==6)+1])
abline(lm(Pub~Year,data=out3.data), col="red4", lwd=2)
# Dataset 4
plot(out4.data$Year, out4.data$Pub, xlab="Years since Ph.D.", ylab="Number of Publications",
ylim=c(0,150), xlim=c(0,60),cex=1.2, pch=pchs[as.numeric(out4.data$CASE==6)+1])
abline(lm(Pub~Year,data=out4.data), col="red4", lwd=2)
```

10.2 Sources of outliers and possible remedial actions

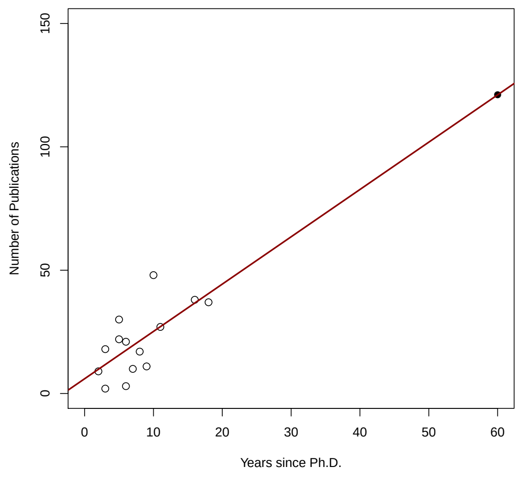
Import Hubner data.

```
# Hubner data
hubner.data <- read.table("C10e03dt.txt")
names(hubner.data)  <- c("CASE", "X", "Y", "out")
```

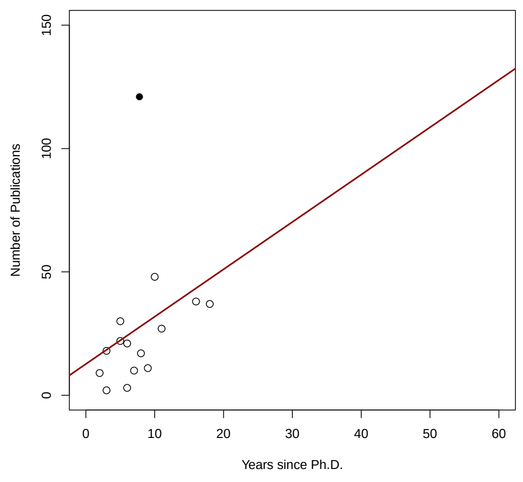
```
# Linear regression with outlier
lin.fit <- lm(Y~X,data=hubner.data)
summary(lin.fit)

##
## Call:
## lm(formula = Y ~ X, data = hubner.data)
##
## Residuals:
##      1      2      3      4      5      6
##  2.086  0.417 -0.271 -1.590 -1.388  0.746
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.0683     0.6328    0.11     0.92
## X            -0.0815     0.1359   -0.60     0.58
##
## Residual standard error: 1.55 on 4 degrees of freedom
## Multiple R-squared:  0.0824,Adjusted R-squared:  -0.147
## F-statistic: 0.359 on 1 and 4 DF,  p-value: 0.581

# Quadratic regression
quad.fit <- lm(Y~X+I(X^2),data=hubner.data)
summary(quad.fit)
```



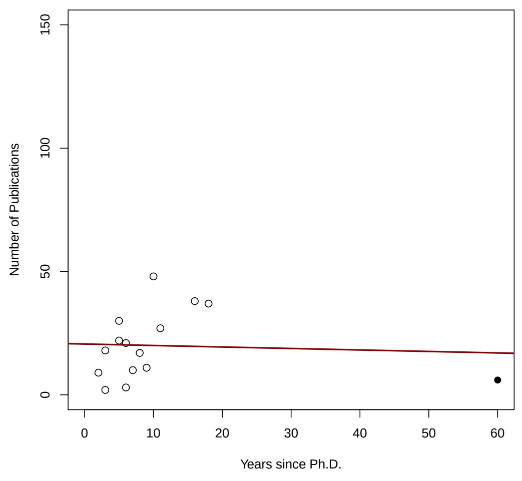
(a) Mean of X, mean of Y



(b) At extreme X, extreme Y (on original regression line)



(c) At the mean of X, extreme value of Y



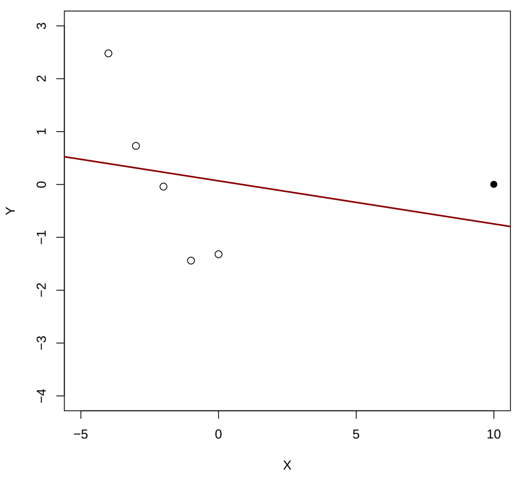
(d) Extreme value of X, extreme value of Y (not on original regression line)

Figure 10.7: Effect of adding a single data point at various locations

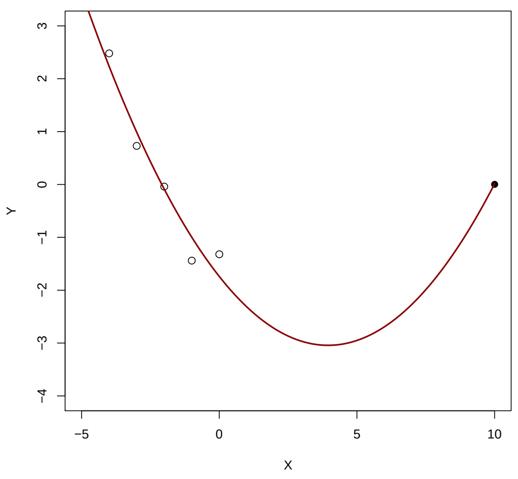
```
##
## Call:
## lm(formula = Y ~ X + I(X^2), data = hubner.data)
##
## Residuals:
##       1       2       3       4       5       6
##  0.2470 -0.2592  0.0477 -0.4424  0.4206 -0.0137
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.7406     0.2970   -5.86   0.0099 **
## X            -0.6595     0.0863   -7.64   0.0046 **
## I(X^2)        0.0835     0.0113    7.37   0.0052 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.41 on 3 degrees of freedom
## Multiple R-squared:  0.952,Adjusted R-squared:  0.92
## F-statistic: 29.7 on 2 and 3 DF,  p-value: 0.0105

# Linear regression without outlier
lin2.fit <- lm(Y~X,data=hubner.data[-6,])
summary(lin2.fit)

##
## Call:
## lm(formula = Y ~ X, data = hubner.data[-6, ])
##
## Residuals:
##      1      2      3      4      5
##  0.444 -0.329 -0.122 -0.545  0.552
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -1.872      0.429   -4.36    0.022 *
## X             -0.977      0.175   -5.57    0.011 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.554 on 3 degrees of freedom
## Multiple R-squared:  0.912,Adjusted R-squared:  0.883
## F-statistic: 31.1 on 1 and 3 DF,  p-value: 0.0114
```

Scatterplots of the Hubner data are given in Figure 10.8.

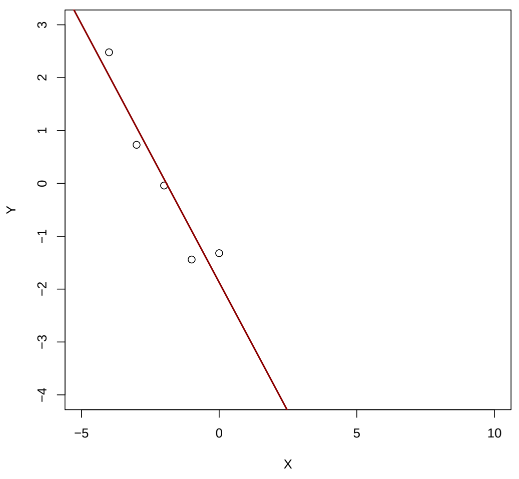
```
# linear
plot(hubner.data$X, hubner.data$Y, xlab="X", ylab="Y", ylim=c(-4,3), xlim=c(-5,10), cex=1.2,
pch=pchs[as.numeric(hubner.data$CASE==6)+1])
abline(lm(Y~X,data=hubner.data), col="red4", lwd=2)

# quadratic plot
plot(hubner.data$X, hubner.data$Y, xlab="X", ylab="Y", ylim=c(-4,3), xlim=c(-5,10), cex=1.2,
pch=pchs[as.numeric(hubner.data$CASE==6)+1])
curve( cbind (1,x,x^2) %*% coef(quad.fit), add=TRUE, lwd=2, col="red4")
# linear with outlier deleted
plot(hubner.data[-6,]$X, hubner.data[-6,]$Y, xlab="X", ylab="Y", ylim=c(-4,3), xlim=c(-5,10), cex=1.2)
abline(lm(Y~X,data=hubner.data[-6,]), col="red4", lwd=2)
```



(a) Fit of linear model



(b) Fit of quadratic model



(c) Fit of linear model with outlier deleted

Figure 10.8: Scatterplot of Hubner's data

10.3 Multicollinearity

The data are not given for CCAW Table 10.5.1, but we can simulate it using the `simulateData()` function in the `lavaan` package. The results will not be exactly like those in CCAW Table 10.5.1, but will be close.

To simulate the data, we use the covariances instead of the correlations. Remember

$$r_{12} = \frac{\sigma_{12}}{\sqrt{\sigma_1^2\sigma_2^2}} = \frac{\sigma_{12}}{\sigma_1\sigma_2}$$

so,

$$\sigma_{12} = r_{12}\sigma_1\sigma_2$$

```
library(lavaan)
# Model A
A.model <- '
# Specify covariances
Y ~~ (0.30*sqrt(5)*sqrt(3))*X1
Y ~~ (0.40*sqrt(5)*sqrt(4))*X2
X1 ~~ 0.00*X2

# Specify means
Y~20*1
X1~0*1
X2~0*1

# Specify variances
Y ~~ 5*Y
X1 ~~ 3*X1
X2 ~~ 4*X2
'

# Double check that the parameters were specified correctly
# Population statistics
fitted(sem(A.model))

## $cov
##     Y    X1   X2
## Y  5.00
## X1 1.16 3.00
## X2 1.79 0.00 4.00
##
## $mean
##  Y X1 X2
## 20  0  0

# Population correlations
cov2cor(fitted(sem(A.model))$cov)

##     Y   X1  X2
## Y  1.0
## X1 0.3 1.0
## X2 0.4 0.0 1.0

# Simulate data
# set the seed if you want your results to exactly match mine
set.seed(45545)
A.data <- simulateData(A.model, sample.nobs=100L)
```

```
# Model B
B.model <- '
# Specify covariances
Y ~~ (0.30*sqrt(5)*sqrt(3))*X1
Y ~~ (0.40*sqrt(5)*sqrt(4))*X2
X1 ~~ (0.10*sqrt(3)*sqrt(4))*X2

# Specify means
Y~20*1
X1~0*1
X2~0*1

# Specify variances
Y ~~ 5*Y
X1 ~~ 3*X1
X2 ~~ 4*X2
'

# Double check that the parameters were specified correctly
# Population statistics
fitted(sem(B.model))

## $cov
##    Y     X1    X2
## Y  5.000
## X1 1.162 3.000
## X2 1.789 0.346 4.000
##
## $mean
##  Y X1 X2
## 20  0  0

# Population correlations
cov2cor(fitted(sem(B.model))$cov)

##    Y   X1  X2
## Y  1.0
## X1 0.3 1.0
## X2 0.4 0.1 1.0

# Simulate data
# set the seed if you want your results to exactly match mine
set.seed(4555)
B.data <- simulateData(B.model, sample.nobs=100L)
```

```
# Model C
C.model <- '
# Specify covariances
Y ~~ (0.30*sqrt(5)*sqrt(3))*X1
Y ~~ (0.40*sqrt(5)*sqrt(4))*X2
X1 ~~ (0.50*sqrt(3)*sqrt(4))*X2

# Specify means
Y~20*1
X1~0*1
X2~0*1

# Specify variances
Y ~~ 5*Y
X1 ~~ 3*X1
X2 ~~ 4*X2
'
```

```
# Double check that the parameters were specified correctly
# Population statistics
fitted(sem(C.model))

## $cov
##    Y    X1   X2
## Y  5.00
## X1 1.16 3.00
## X2 1.79 1.73 4.00
##
## $mean
##  Y X1 X2
## 20  0  0

# Population correlations
cov2cor(fitted(sem(C.model))$cov)

##    Y   X1  X2
## Y  1.0
## X1 0.3 1.0
## X2 0.4 0.5 1.0

# Simulate data
# set the seed if you want your results to exactly match mine
set.seed(4555111)
C.data <- simulateData(C.model, sample.nobs=100L)
```

```
# Model D
D.model <- '
# Specify covariances
Y ~~ (0.30*sqrt(5)*sqrt(3))*X1
Y ~~ (0.40*sqrt(5)*sqrt(4))*X2
X1 ~~ (0.90*sqrt(3)*sqrt(4))*X2

# Specify means
Y~20*1
X1~0*1
X2~0*1

# Specify variances
Y ~~ 5*Y
X1 ~~ 3*X1
X2 ~~ 4*X2
'

# Double check that the parameters were specified correctly
# Population statistics
fitted(sem(D.model))

## $cov
##    Y    X1   X2
## Y  5.00
## X1 1.16 3.00
## X2 1.79 3.12 4.00
##
## $mean
##  Y X1 X2
## 20  0  0

# Population correlations
cov2cor(fitted(sem(D.model))$cov)
```

```
##    Y   X1  X2
## Y  1.0
## X1 0.3 1.0
## X2 0.4 0.9 1.0

# Simulate data
# set the seed if you want your results to exactly match mine
set.seed(4533355)
D.data <- simulateData(D.model, sample.nobs=100L)
```

```
# Model E
E.model <- '
# Specify covariances
Y ~~ (0.30*sqrt(5)*sqrt(3))*X1
Y ~~ (0.40*sqrt(5)*sqrt(4))*X2
X1 ~~ (0.949*sqrt(3)*sqrt(4))*X2

# Specify means
Y~20*1
X1~0*1
X2~0*1

# Specify variances
Y ~~ 5*Y
X1 ~~ 3*X1
X2 ~~ 4*X2
'

# Double check that the parameters were specified correctly
# Population statistics
fitted(sem(E.model))

## $cov
##    Y    X1   X2
## Y  5.00
## X1 1.16 3.00
## X2 1.79 3.29 4.00
##
## $mean
##  Y X1 X2
## 20  0  0

# Population correlations
cov2cor(fitted(sem(E.model))$cov)

##    Y     X1    X2
## Y  1.000
## X1 0.300 1.000
## X2 0.400 0.949 1.000

# Simulate data
# set the seed if you want your results to exactly match mine
set.seed(45512215)
E.data <- simulateData(E.model, sample.nobs=100L)
```

Similar syntax can simulate the data with four predictors.

```
# Model A4
A4.model <- '
# Specify covariances (they differ from the book---I took these from the CCAW SPSS file)
Y ~~ (0.30*sqrt(5)*sqrt(3))*X1
```

```
Y ~~ (0.40*sqrt(5)*sqrt(4))*X2
Y ~~ (0.30*sqrt(5)*sqrt(3))*X3
Y ~~ (0.40*sqrt(5)*sqrt(4))*X4

X1 ~~ 0.00*X2 + 0.00*X3 + 0.00*X4
X2 ~~ 0.00*X3 + 0.00*X4
X3 ~~ 0.00*X4

# Specify means
Y~20*1
X1~0*1
X2~0*1
X3~0*1
X4~0*1


# Specify variances
Y ~~ 5*Y
X1 ~~ 3*X1
X2 ~~ 4*X2
X3 ~~ 5*X3
X4 ~~ 4*X4

'

# Double check that the parameters were specified correctly
# Population statistics
fitted(sem(A4.model))
















# Population correlations
cov2cor(fitted(sem(A4.model))$cov)









# Simulate data
# set the seed if you want your results to exactly match mine
set.seed(45545553)
A4.data <- simulateData(A4.model, sample.nobs=100L)
```

```
# Model B4
B4.model <- '
# Specify covariances (they differ from the book---I took these from the CCAW SPSS file)
Y ~~ (0.30*sqrt(5)*sqrt(3))*X1
Y ~~ (0.35*sqrt(5)*sqrt(4))*X2
```

```
Y ~~ (0.40*sqrt(5)*sqrt(5))*X3
Y ~~ (0.35*sqrt(5)*sqrt(4))*X4

X1 ~~ (0.933*sqrt(3)*sqrt(4))*X2 + (0.933*sqrt(3)*sqrt(5))*X3 + 0.00*X4
X2 ~~ (0.933*sqrt(4)*sqrt(5))*X3 + 0.00*X4
X3 ~~ 0.00*X4

# Specify means
Y~20*1
X1~0*1
X2~0*1
X3~0*1
X4~0*1


# Specify variances
Y ~~ 5*Y
X1 ~~ 3*X1
X2 ~~ 4*X2
X3 ~~ 5*X3
X4 ~~ 4*X4

'

# Double check that the parameters were specified correctly
# Population statistics
fitted(sem(B4.model))
















# Population correlations
cov2cor(fitted(sem(B4.model))$cov)









# Simulate data
# set the seed if you want your results to exactly match mine
set.seed(43)
B4.data <- simulateData(B4.model, sample.nobs=100L)
```

Based on the simulated data, we can calculate the regression coefficients as well as the VIF using the `vif()` function in the `car` package. The partial correlation can be calculated from the `pcor()` function in the `ppcor` package.

```
# Two predictor models
# Model A
A.fit <- lm(Y~X1+X2, data=A.data)
```

```
summary(A.fit)

##
## Call:
## lm(formula = Y ~ X1 + X2, data = A.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -4.964 -1.210  0.187  1.238  3.895
##
## Coefficients:
##             Estimate Std. Error t value             Pr(>|t|)
## (Intercept)  20.3477     0.1829  111.23 < 0.0000000000000002 ***
## X1            0.3446     0.1082    3.19              0.00195 **
## X2            0.3561     0.0939    3.79              0.00026 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.82 on 97 degrees of freedom
## Multiple R-squared:  0.22,Adjusted R-squared:  0.204
## F-statistic: 13.7 on 2 and 97 DF,  p-value: 0.00000571

library(car)
vif(A.fit)

##   X1   X2
## 1.01 1.01

library(ppcor)
pcor(A.data)$estimate^2

##         Y       X1       X2
## Y  1.0000 0.094683 0.129225
## X1 0.0947 1.000000 0.000266
## X2 0.1292 0.000266 1.000000

# Model B
B.fit <- lm(Y~X1+X2, data=B.data)
summary(B.fit)

##
## Call:
## lm(formula = Y ~ X1 + X2, data = B.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -4.511 -1.520 -0.107  1.330  4.559
##
## Coefficients:
##             Estimate Std. Error t value             Pr(>|t|)
## (Intercept)   19.883      0.209   95.17 < 0.0000000000000002 ***
## X1             0.128      0.119    1.07                 0.29
## X2             0.508      0.103    4.94            0.0000033 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.08 on 97 degrees of freedom
## Multiple R-squared:  0.219,Adjusted R-squared:  0.203
## F-statistic: 13.6 on 2 and 97 DF,  p-value: 0.00000623

vif(B.fit)

##   X1   X2
## 1.01 1.01
```

```
pcor(B.data)$estimate^2

##         Y      X1      X2
## Y  1.0000 0.01177 0.20081
## X1 0.0118 1.00000 0.00338
## X2 0.2008 0.00338 1.00000

# Model C
C.fit <- lm(Y~X1+X2, data=C.data)
summary(C.fit)

##
## Call:
## lm(formula = Y ~ X1 + X2, data = C.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -4.871 -1.376 -0.202  1.109  4.791
##
## Coefficients:
##             Estimate Std. Error t value             Pr(>|t|)
## (Intercept)   19.686      0.185  106.67 < 0.0000000000000002 ***
## X1             0.166      0.122    1.36              0.17547
## X2             0.390      0.112    3.48              0.00075 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.84 on 97 degrees of freedom
## Multiple R-squared:  0.219,Adjusted R-squared:  0.203
## F-statistic: 13.6 on 2 and 97 DF,  p-value: 0.0000063

vif(C.fit)

##   X1   X2
## 1.42 1.42

pcor(C.data)$estimate^2

##         Y     X1    X2
## Y  1.0000 0.0188 0.111
## X1 0.0188 1.0000 0.213
## X2 0.1111 0.2130 1.000

# Model D
D.fit <- lm(Y~X1+X2, data=D.data)
summary(D.fit)

##
## Call:
## lm(formula = Y ~ X1 + X2, data = D.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -5.760 -1.219  0.004  1.231  6.813
##
## Coefficients:
##             Estimate Std. Error t value            Pr(>|t|)
## (Intercept)   19.940      0.210   95.14 <0.0000000000000002 ***
## X1             0.146      0.283    0.51               0.608
## X2             0.416      0.229    1.82               0.072 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.08 on 97 degrees of freedom
## Multiple R-squared:  0.227,Adjusted R-squared:  0.211
## F-statistic: 14.2 on 2 and 97 DF,  p-value: 0.00000375
```

```
vif(D.fit)

##   X1   X2
## 5.42 5.42

pcor(D.data)$estimate^2

##          Y      X1     X2
## Y  1.00000 0.00272 0.0329
## X1 0.00272 1.00000 0.7698
## X2 0.03294 0.76983 1.0000

# Model E
E.fit <- lm(Y~X1+X2, data=E.data)
summary(E.fit)

##
## Call:
## lm(formula = Y ~ X1 + X2, data = E.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -5.386 -1.272  0.035  1.365  4.361
##
## Coefficients:
##             Estimate Std. Error t value             Pr(>|t|)
## (Intercept)   19.984      0.184  108.73 < 0.0000000000000002 ***
## X1            -1.088      0.304   -3.58              0.00054 ***
## X2             1.150      0.257    4.47             0.000021 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.83 on 97 degrees of freedom
## Multiple R-squared:  0.203,Adjusted R-squared:  0.186
## F-statistic: 12.3 on 2 and 97 DF,  p-value: 0.0000168

vif(E.fit)

##   X1   X2
## 10.3 10.3

pcor(E.data)$estimate^2

##        Y    X1    X2
## Y  1.000 0.117 0.171
## X1 0.117 1.000 0.911
## X2 0.171 0.911 1.000
```

```
# Four predictor models

# Model A4
A4.fit <- lm(Y~X1+X2+X3+X4, data=A4.data)
summary(A4.fit)

##
## Call:
## lm(formula = Y ~ X1 + X2 + X3 + X4, data = A4.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -3.340 -0.995 -0.109  0.894  3.917
```

```
##
## Coefficients:
##             Estimate Std. Error t value            Pr(>|t|)
## (Intercept)  20.1869     0.1574  128.24 < 0.0000000000000002 ***
## X1            0.4452     0.0952    4.68          0.000009693 ***
## X2            0.4396     0.0761    5.77          0.000000097 ***
## X3            0.1548     0.0704    2.20                 0.03 *
## X4            0.3182     0.0712    4.47          0.000021487 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.54 on 95 degrees of freedom
## Multiple R-squared:  0.455,Adjusted R-squared:  0.432
## F-statistic: 19.8 on 4 and 95 DF,  p-value: 0.00000000000692

vif(A4.fit)

##   X1   X2   X3   X4
## 1.02 1.01 1.01 1.01

pcor(A4.data)$estimate^2

##         Y     X1     X2     X3     X4
## Y  1.0000 0.1870 0.2597 0.0484 0.1739
## X1 0.1870 1.0000 0.0808 0.0109 0.0103
## X2 0.2597 0.0808 1.0000 0.0303 0.0383
## X3 0.0484 0.0109 0.0303 1.0000 0.0093
## X4 0.1739 0.0103 0.0383 0.0093 1.0000

# Model B4
B4.fit <- lm(Y~X1+X2+X3+X4, data=B4.data)
summary(B4.fit)

##
## Call:
## lm(formula = Y ~ X1 + X2 + X3 + X4, data = B4.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -6.002 -1.069 -0.028  1.020  6.337
##
## Coefficients:
##             Estimate Std. Error t value            Pr(>|t|)
## (Intercept)   19.995      0.207   96.54 <0.0000000000000002 ***
## X1            -1.086      0.421   -2.58              0.0113 *
## X2             0.108      0.327    0.33              0.7418
## X3             0.991      0.326    3.04              0.0031 **
## X4             0.250      0.112    2.24              0.0276 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.04 on 95 degrees of freedom
## Multiple R-squared:  0.177,Adjusted R-squared:  0.143
## F-statistic: 5.12 on 4 and 95 DF,  p-value: 0.000881

vif(B4.fit)

##    X1    X2    X3    X4
## 12.73 10.38 13.37  1.03

pcor(B4.data)$estimate^2
```

```
##          Y      X1      X2     X3     X4
## Y  1.00000 0.0656 0.00115 0.0886 0.0501
## X1 0.06561 1.0000 0.16376 0.3869 0.0145
## X2 0.00115 0.1638 1.00000 0.1745 0.0001
## X3 0.08863 0.3869 0.17455 1.0000 0.0313
## X4 0.05005 0.0145 0.00010 0.0313 1.0000
```

The `colldiag()` function in the `perturb` package supplies other multicollinearity indices, such as the condition number.

```
library(perturb)
colldiag(A.fit)

## Condition
## Index Variance Decomposition Proportions
##          intercept X1    X2
## 1   1.000 0.113     0.486 0.284
## 2   1.042 0.620     0.004 0.338
## 3   1.143 0.267     0.510 0.378

colldiag(B.fit)

## Condition
## Index Variance Decomposition Proportions
##          intercept X1    X2
## 1   1.000 0.213     0.336 0.276
## 2   1.123 0.686     0.015 0.344
## 3   1.172 0.102     0.649 0.380

colldiag(C.fit)

## Condition
## Index Variance Decomposition Proportions
##          intercept X1    X2
## 1   1.000 0.002     0.228 0.226
## 2   1.241 0.980     0.000 0.008
## 3   1.844 0.018     0.772 0.766

colldiag(D.fit)

## Condition
## Index Variance Decomposition Proportions
##          intercept X1    X2
## 1   1.000 0.012     0.046 0.046
## 2   1.403 0.988     0.002 0.002
## 3   4.480 0.000     0.952 0.951

colldiag(E.fit)

## Condition
## Index Variance Decomposition Proportions
##          intercept X1    X2
## 1   1.000 0.001     0.025 0.025
## 2   1.399 0.996     0.000 0.000
## 3   6.263 0.003     0.975 0.975

colldiag(A4.fit)

## Condition
## Index Variance Decomposition Proportions
##          intercept X1    X2    X3    X4
## 1   1.000 0.412     0.076 0.166 0.147 0.021
## 2   1.030 0.026     0.354 0.125 0.109 0.268
## 3   1.059 0.000     0.039 0.270 0.353 0.268
## 4   1.170 0.046     0.529 0.140 0.000 0.424
## 5   1.223 0.516     0.003 0.300 0.391 0.020
```

```
colldiag(B4.fit)

## Condition
## Index Variance Decomposition Proportions
##          intercept X1    X2    X3    X4
## 1  1.000 0.002     0.009 0.011 0.008 0.005
## 2  1.681 0.542     0.000 0.001 0.001 0.375
## 3  1.778 0.435     0.000 0.000 0.000 0.607
## 4  6.718 0.000     0.264 0.978 0.135 0.001
## 5  7.836 0.021     0.727 0.011 0.856 0.013
```

Chapter 11

Missing Data

11.1 Comparing alternative methods

```
# PhD/Publication data with missing obervations (Table 11.3.2)
phdMisFull.data<-read.table("C1103DT.txt", header=TRUE, na.strings = 999)
```

```
# True Model
regTrue.fit <- lm(SALARY~TIMEA+SEX+PUBA+CITB,data=phdMisFull.data)
summary(regTrue.fit)

##
## Call:
## lm(formula = SALARY ~ TIMEA + SEX + PUBA + CITB, data = phdMisFull.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -14273  -3478   -223   3371  19895
##
## Coefficients:
##             Estimate Std. Error t value             Pr(>|t|)
## (Intercept)  38386.7     1896.8   20.24 < 0.0000000000000002 ***
## TIMEA          742.7      154.1    4.82            0.0000065 ***
## SEX           1143.4     1375.5    0.83                0.408
## PUBA           130.6       57.3    2.28                0.025 *
## CITB           214.2       38.4    5.58            0.0000003 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6160 on 82 degrees of freedom
## Multiple R-squared:  0.578,Adjusted R-squared:  0.557
## F-statistic: 28.1 on 4 and 82 DF,  p-value: 0.0000000000000109
```

```
# Subset data
vars <- c("TIME", "SEX", "PUB", "CITA", "SALARY")
phdMis.data <- phdMisFull.data[vars]
head(phdMis.data)

##   TIME SEX PUB CITA SALARY
## 1    3   0  18   50  51876
## 2    6   0   3   26  54511
## 3    3   0   2   50  53425
## 4    8   1  17   34  61863
## 5    9   0  11   41  52926
## 6    6   1   6   37  47034
```

11.1.1 Listwise Deletion

The default method in **R** is to delete missing values listwise.

```
# Listwise Deletion
regList.fit <- lm(SALARY~TIME+SEX+PUB+CITA,data=phdMis.data)
summary(regList.fit)

##
## Call:
## lm(formula = SALARY ~ TIME + SEX + PUB + CITA, data = phdMis.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -13377  -4482   -990   4316  20671
##
## Coefficients:
##             Estimate Std. Error t value             Pr(>|t|)
## (Intercept)  38669.6     2484.7   15.56 < 0.0000000000000002 ***
## TIME           857.0      287.9    2.98              0.00428 **
## SEX            917.8     1859.9    0.49              0.62360
## PUB             92.7       85.9    1.08              0.28498
## CITA           201.9       57.5    3.51              0.00088 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7080 on 57 degrees of freedom
##   (25 observations deleted due to missingness)
## Multiple R-squared:  0.503,Adjusted R-squared:  0.468
## F-statistic: 14.4 on 4 and 57 DF,  p-value: 0.0000000336
```

11.1.2 Full Information Maximum Likelihood (FIML)

Its equivalent to use FIML for the EM algorithm.

```
library(lavaan)

reg.model <- '
SALARY ~ 1+ TIME + SEX + PUB + CITA
'

phdFIML.fit <- sem(reg.model, data=phdMis.data, missing='fiml')
summary(phdFIML.fit,rsquare=TRUE)

## lavaan (0.5-20) converged normally after  81 iterations
##
##   Number of observations                            87
##
##   Number of missing patterns                         4
##
##   Estimator                                         ML
##   Minimum Function Test Statistic                0.000
##   Degrees of freedom                                 0
##
## Parameter Estimates:
##
##   Information                                 Observed
##   Standard Errors                             Standard
##
## Regressions:
##                      Estimate      Std.Err     Z-value  P(>|z|)
##   SALARY ~
##     TIME                856.407     235.173    3.642    0.000
##     SEX                1361.820    1465.239    0.929    0.353
##     PUB                 105.422      73.212    1.440    0.150
##     CITA                182.470      43.337    4.211    0.000
##
## Intercepts:
##                       Estimate      Std.Err    Z-value  P(>|z|)
##     SALARY            40226.796    1982.994   20.286    0.000
##
## Variances:
##                       Estimate      Std.Err    Z-value  P(>|z|)
##     SALARY          42004114.640 6509595.729    6.453    0.000
##
## R-Square:
##                       Estimate
##     SALARY               0.505
```

If you really want to use the EM algorithm, you have to do multiple steps.

1. Calculate the covariance matrix and mean vector using EM;
2. Input the EM-based covariance matrix and mean vector and estimate the regression coefficients from them.

```
# EM Algorithm

library(norm)

phdmis <- cbind(phdMis.data$TIME, phdMis.data$SEX, phdMis.data$PUB, phdMis.data$CITA,
        phdMis.data$SALARY)
colnames(phdmis) <- c("TIME", "SEX", "PUB", "CITA", "SALARY")
rownames(phdmis) <- seq(1,dim(phdmis)[1],1)

s <- prelim.norm(phdmis)
thetahat <- em.norm(s)

## Iterations of EM:
## 1...2...3...4...5...6...7...

phdEM.data <- getparam.norm(s,thetahat,corr=TRUE)
phdEM.cor <- getparam.norm(s,thetahat,corr=TRUE)$r
phdEM.sd <- getparam.norm(s,thetahat,corr=TRUE)$sdv
phdEM.mean <- getparam.norm(s,thetahat,corr=TRUE)$mu

names(phdEM.mean) <- colnames(phdEM.cor) <- rownames(phdEM.cor) <-
c("TIME", "SEX", "PUB", "CITA", "SALARY")

library(lavaan)
phdEM.cov <- cor2cov(phdEM.cor, phdEM.data$sd)


# With Intercept
phdEM.fit <- sem(reg.model, sample.cov=phdEM.cov, sample.nobs=87, sample.mean=phdEM.mean)
summary(phdEM.fit,rsquare=TRUE)

## lavaan (0.5-20) converged normally after  92 iterations
##
##   Number of observations                            87
##
##   Estimator                                         ML
##   Minimum Function Test Statistic               0.000
##   Degrees of freedom                                0
##
## Parameter Estimates:
##
##   Information                                Expected
##   Standard Errors                            Standard
##
## Regressions:
##                    Estimate      Std.Err     Z-value  P(>|z|)
##   SALARY ~
##     TIME              856.349      223.103    3.838    0.000
##     SEX              1361.875     1441.952    0.944    0.345
##     PUB               105.432       69.469    1.518    0.129
##     CITA              182.467       42.003    4.344    0.000
##
## Intercepts:
##                    Estimate      Std.Err     Z-value  P(>|z|)
##     SALARY          40227.165     1919.548   20.957    0.000
##
## Variances:
##                    Estimate      Std.Err     Z-value  P(>|z|)
##     SALARY         41522824.973 6295674.475    6.595    0.000
##
## R-Square:
##                    Estimate
##     SALARY              0.505
```

11.1.3 Pairwise Deletion

For pairwise deletion, you have to estimate the covariances first and use those as input into the regression.

```
phdPair.cov <- cov(phdMis.data,use="pairwise.complete.obs")
phdPair.mean <- colMeans(phdMis.data,na.rm=TRUE)

phdPair.fit <- sem(reg.model, sample.cov=phdPair.cov, sample.nobs=87, sample.mean=phdPair.mean)
summary(phdPair.fit,rsquare=TRUE)

## lavaan (0.5-20) converged normally after  91 iterations
##
##   Number of observations                           87
##
##   Estimator                                        ML
##   Minimum Function Test Statistic               0.000
##   Degrees of freedom                                0
##   Minimum Function Value               0.0000000000000
##
## Parameter Estimates:
##
##   Information                                Expected
##   Standard Errors                            Standard
##
## Regressions:
##                    Estimate      Std.Err     Z-value  P(>|z|)
##   SALARY ~
##     TIME              982.333      227.014    4.327    0.000
##     SEX              1127.386     1410.788    0.799    0.424
##     PUB                46.038       70.534    0.653    0.514
##     CITA              187.249       41.006    4.566    0.000
##
```

```
## Intercepts:
##                    Estimate      Std.Err     Z-value  P(>|z|)
##     SALARY            40468.497    1830.743   22.105    0.000
##
## Variances:
##                    Estimate      Std.Err     Z-value  P(>|z|)
##     SALARY         40204529.190 6095794.984    6.595    0.000
##
## R-Square:
##                    Estimate
##     SALARY                0.526
```

11.1.4 Mean Imputation

Mean imputation can be done using the `Hmisc` package.

```
library(Hmisc)
# Create mean-imputed data set
phdMeanI.data<-phdMis.data
phdMeanI.data$TIME <- impute(phdMeanI.data$TIME, fun=mean)
phdMeanI.data$PUB <- impute(phdMeanI.data$PUB, fun=mean)
phdMeanI.data$CITA <- impute(phdMeanI.data$CITA, fun=mean)

phdMeanI.fit <- sem(reg.model, data=phdMeanI.data)
summary(phdMeanI.fit, rsquare=TRUE)

## lavaan (0.5-20) converged normally after  92 iterations
##
##   Number of observations                            87
##
##   Estimator                                         ML
##   Minimum Function Test Statistic                0.000
##   Degrees of freedom                                 0
##   Minimum Function Value               0.0000000000000
##
## Parameter Estimates:
##
##   Information                                 Expected
##   Standard Errors                             Standard
##
## Regressions:
##                    Estimate      Std.Err     Z-value  P(>|z|)
##   SALARY ~
##     TIME                954.472     226.285    4.218    0.000
##     SEX                1249.409    1450.081    0.862    0.389
##     PUB                  78.357      71.426    1.097    0.273
##     CITA                192.431      43.385    4.435    0.000
##
## Intercepts:
##                    Estimate      Std.Err     Z-value  P(>|z|)
##     SALARY            39809.485    1970.758   20.200    0.000
##
## Variances:
##                    Estimate      Std.Err     Z-value  P(>|z|)
##     SALARY         42544889.396 6450639.483    6.595    0.000
##
## R-Square:
##                    Estimate
##     SALARY                0.498
```

11.1.5 Multiple Imputation

For multiple imputation, it is easiest to use the `Amelia` package to generate the imputations and the `semTools` package to analyze the results.

```
library(Amelia)
library(semTools)
bds <- matrix(c(1, 0, 40,3,0,100), byrow=TRUE, nrow = 2, ncol = 3)
phdMI.data <- amelia(phdMis.data,m=5, noms="SEX", p2s=0, bounds=bds)
phdMI.fit <- runMI(reg.model, data=phdMI.data$imputations, fun="sem")
summary(phdMI.fit,rsquare=TRUE)

## lavaan (0.5-20) converged normally after   5 iterations
##
##   Number of observations                            87
##
##   Estimator                                         ML
##   Minimum Function Test Statistic                0.000
##   Degrees of freedom                                 0
##   Minimum Function Value               0.5214981406669
##
## Parameter Estimates:
##
##   Information                                 Expected
##   Standard Errors                             Standard
##
## Regressions:
##                    Estimate      Std.Err     Z-value  P(>|z|)
##   SALARY ~
##     TIME              825.865      232.458    3.553    0.000
##     SEX              1165.115     1451.222    0.803    0.422
##     PUB               120.871       73.643    1.641    0.101
##     CITA              182.099       43.307    4.205    0.000
##
## Intercepts:
##                    Estimate      Std.Err     Z-value  P(>|z|)
##     SALARY          40137.377     1975.104   20.322    0.000
##
## Variances:
##                    Estimate      Std.Err     Z-value  P(>|z|)
##     SALARY         41418792.605 6446096.983    6.425    0.000
##
## R-Square:
##                    Estimate
##     SALARY              0.512
```

Chapter 12

Multiple Regression/Correlation and Causal Models

12.1 Models without Reciprocal Causation

CCAW give the path model shown in Figure 12.1. First, we need to import the data.

```
# Income Data for Chapter 12
Income.data<-read.table("C1201DT.txt", header=TRUE, sep="\t")
```

The `lavaan` package will allow us to estimate coefficients from path models. We will add labels to the model to help with interpretation. To make a label in `lavaan`, multiple the label by the predictor variable (i.e., `a*PREDICTOR`).

```
library(lavaan)
#Illustative five-variable model
fv.model<- '
SALARY ~ l*SEX + m*TIME + n*PUB + o*CIT
CIT ~ i*SEX + j*TIME + k*PUB
PUB ~ g*SEX + h*TIME
TIME ~ f*SEX
'
# fitting the model
fv.fit<- sem(fv.model, data=Income.data)

## Warning in lav_data_full(data = data, group = group, group.label = group.label, :  lavaan WARNING:
some observed variances are (at least) a factor 1000 times larger than others; use varTable(fit)
to investigate
```

```
# showing the results from the model
summary(fv.fit, standardized=TRUE, rsquare=TRUE)

## lavaan (0.5-20) converged normally after 135 iterations
##
##   Number of observations                            62
##
##   Estimator                                         ML
##   Minimum Function Test Statistic                0.000
##   Degrees of freedom                                 0
##   Minimum Function Value               0.0000000000000
##
## Parameter Estimates:
##
##   Information                                 Expected
```

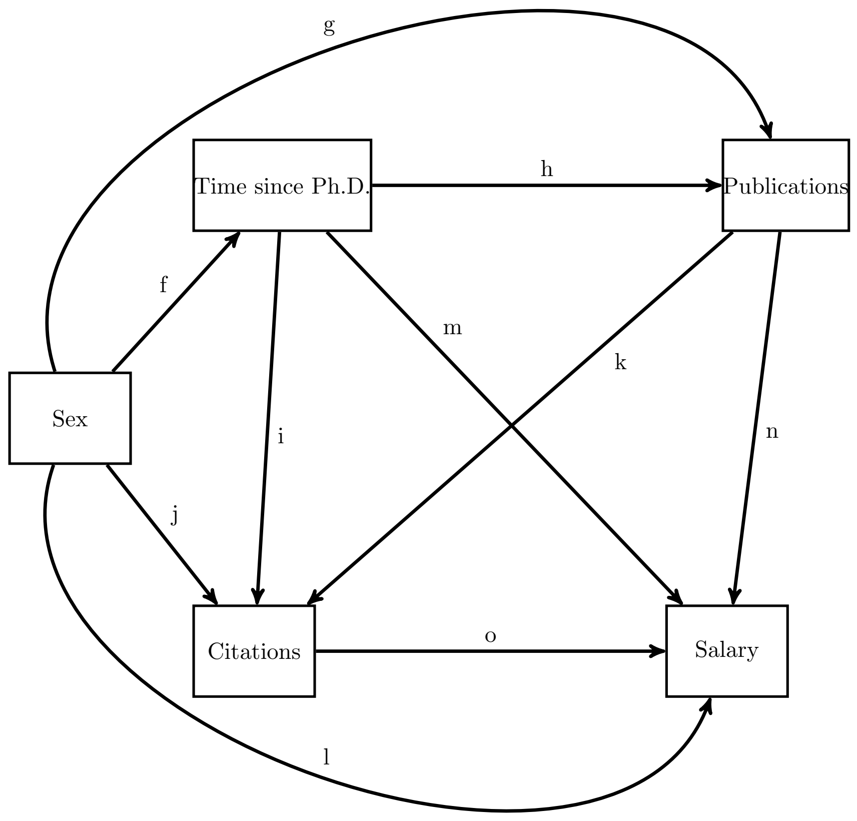


Figure 12.1: Five Variable Model from CCAW, p. 461

```
##   Standard Errors                             Standard
##
## Regressions:
##                    Estimate      Std.Err     Z-value  P(>|z|)   Std.lv       Std.all
##   SALARY ~
##     SEX        (l)      917.767     1783.362    0.515    0.607      917.767    0.047
##     TIME       (m)      857.006      276.091    3.104    0.002      857.006    0.378
##     PUB        (n)       92.746       82.391    1.126    0.260       92.746    0.134
##     CIT        (o)      201.931       55.141    3.662    0.000      201.931    0.357
##   CIT ~
##     SEX        (i)        2.426        4.096    0.592    0.554        2.426    0.071
##     TIME       (j)        1.034        0.622    1.661    0.097        1.034    0.257
##     PUB        (k)        0.190        0.188    1.008    0.314        0.190    0.155
##   PUB ~
##     SEX        (g)        0.657        2.762    0.238    0.812        0.657    0.023
##     TIME       (h)        2.114        0.323    6.548    0.000        2.114    0.646
##   TIME ~
##     SEX        (f)        1.794        1.063    1.688    0.091        1.794    0.210
##
## Variances:
##                    Estimate      Std.Err     Z-value  P(>|z|)   Std.lv       Std.all
##     SALARY         46042901.212 8269549.178    5.568    0.000 46042901.212    0.497
##     CIT                 244.239       43.867    5.568    0.000      244.239    0.842
##     PUB                 111.191       19.971    5.568    0.000      111.191    0.576
##     TIME                 17.214        3.092    5.568    0.000       17.214    0.956
##
## R-Square:
##                    Estimate
##     SALARY                0.503
##     CIT                   0.158
##     PUB                   0.424
##     TIME                  0.044
```

A path model with estimated values is given in Figure 12.2.

The zero-order B and β coefficients (see CCAW table 12.2.1) are simple regressions with unstandardized and standardized coefficients, respectively.

```
# Unstandardized
coef(lm(Income.data$SALARY~Income.data$SEX))[2]

## Income.data$SEX
##            3902

coef(lm(Income.data$SALARY~Income.data$TIME))[2]

## Income.data$TIME
##             1379

coef(lm(Income.data$SALARY~Income.data$PUB))[2]

## Income.data$PUB
##             351

coef(lm(Income.data$SALARY~Income.data$CIT))[2]

## Income.data$CIT
##             311

# Standardized
coef(lm(scale(Income.data$SALARY)~scale(Income.data$SEX)))[2]
```

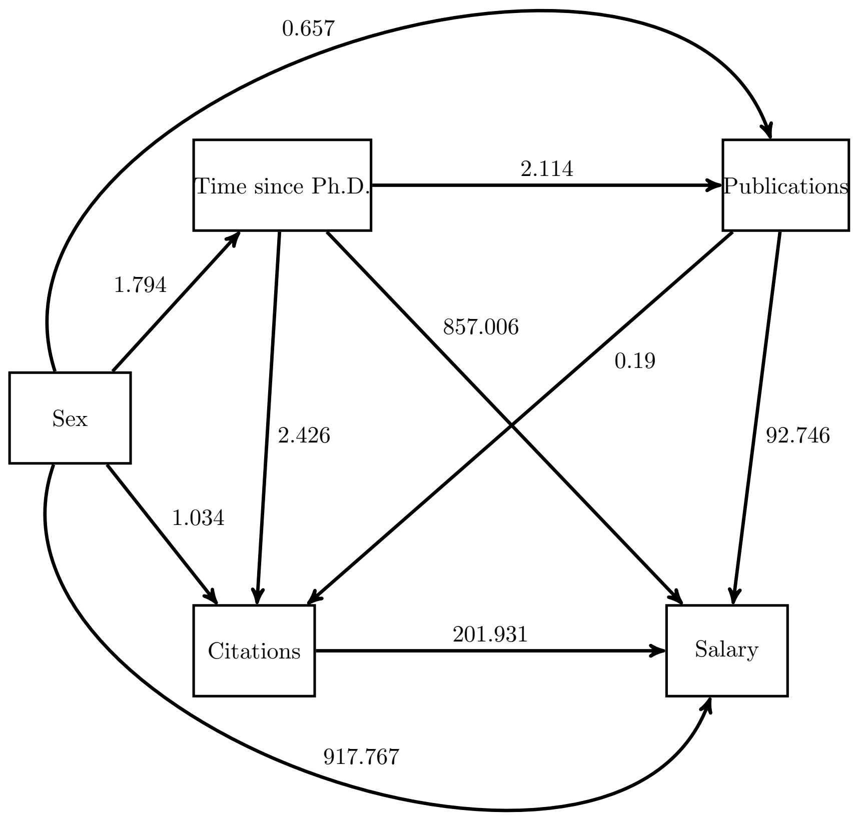


Figure 12.2: Five Variable Model from CCAW, p. 461, with path values

```
## scale(Income.data$SEX)
##                  0.201

coef(lm(scale(Income.data$SALARY)~scale(Income.data$TIME)))[2]

## scale(Income.data$TIME)
##                   0.608

coef(lm(scale(Income.data$SALARY)~scale(Income.data$PUB)))[2]

## scale(Income.data$PUB)
##                  0.506

coef(lm(scale(Income.data$SALARY)~scale(Income.data$CIT)))[2]

## scale(Income.data$CIT)
##                   0.55
```

To obtain the indirect and total effects from `lavaan`, we have to use the `:=` operator, which define a new parameter, based on parameters already defined in the model.

```
library(lavaan)
#Illustative five-variable model
fvIndirect.model<- '
SALARY ~ l*SEX + m*TIME + n*PUB + o*CIT
CIT ~ i*SEX + j*TIME + k*PUB
PUB ~ g*SEX + h*TIME
TIME ~ f*SEX

## Indirect Effects
# Sex
Salary.Citations.Sex:= i*o
Salary.Pubs.Sex:= g*n+g*k*o
Salary.Time.Sex:= f*m + f*h*n+f*j*o+f*h*k*o
Sex.Total:=Salary.Citations.Sex+Salary.Pubs.Sex+Salary.Time.Sex+l

# Time
Salary.Citations.Time := j*o
Salary.Pubs.Time := h*n +h*k*o
Time.Total:=Salary.Citations.Time + Salary.Pubs.Time + m

# Pubs
Salary.Citations.Pubs := k*o
Pubs.Total:=Salary.Citations.Pubs + n

# Citations
Citations.Total:=o
'
```

```
# estimate the model and obtain the coefficients
fvIndirect.fit<- sem(fvIndirect.model, data=Income.data)

## Warning in lav_data_full(data = data, group = group, group.label = group.label, :  lavaan WARNING:
some observed variances are (at least) a factor 1000 times larger than others; use varTable(fit)
to investigate

summary(fvIndirect.fit )

## lavaan (0.5-20) converged normally after 135 iterations
##
##   Number of observations                            62
```

```
##
##   Estimator                                         ML
##   Minimum Function Test Statistic                0.000
##   Degrees of freedom                                 0
##   Minimum Function Value               0.0000000000000
##
## Parameter Estimates:
##
##   Information                                 Expected
##   Standard Errors                             Standard
##
## Regressions:
##                    Estimate      Std.Err     Z-value  P(>|z|)
##   SALARY ~
##     SEX        (l)      917.767    1783.362    0.515    0.607
##     TIME       (m)      857.006     276.091    3.104    0.002
##     PUB        (n)       92.746      82.391    1.126    0.260
##     CIT        (o)      201.931      55.141    3.662    0.000
##   CIT ~
##     SEX        (i)        2.426       4.096    0.592    0.554
##     TIME       (j)        1.034       0.622    1.661    0.097
##     PUB        (k)        0.190       0.188    1.008    0.314
##   PUB ~
##     SEX        (g)        0.657       2.762    0.238    0.812
##     TIME       (h)        2.114       0.323    6.548    0.000
##   TIME ~
##     SEX        (f)        1.794       1.063    1.688    0.091
##
## Variances:
##                    Estimate      Std.Err     Z-value  P(>|z|)
##     SALARY         46042901.212 8269549.178    5.568    0.000
##     CIT                  244.239      43.867    5.568    0.000
##     PUB                  111.191      19.971    5.568    0.000
##     TIME                  17.214       3.092    5.568    0.000
##
## Defined Parameters:
##                    Estimate      Std.Err     Z-value  P(>|z|)
##     Salry.Cttns.Sx      489.976     837.824    0.585    0.559
##     Salary.Pubs.Sx       86.040     366.794    0.235    0.815
##     Salary.Time.Sx     2408.312    1486.802    1.620    0.105
##     Sex.Total          3902.094    2415.677    1.615    0.106
##     Salry.Cttns.Tm      208.701     137.961    1.513    0.130
##     Salary.Pubs.Tm      276.980     195.139    1.419    0.156
##     Time.Total         1342.687     232.944    5.764    0.000
##     Slry.Cttns.Pbs       38.300      39.421    0.972    0.331
##     Pubs.Total          131.046      90.130    1.454    0.146
##     Citations.Totl      201.931      55.141    3.662    0.000
```

Chapter 13

Alternative Regression Models: Logistic, Poisson Regression, and the Generalized Linear Model

13.1 Logistic Regression

Mammogram data from CCAW Table 13.2.2.

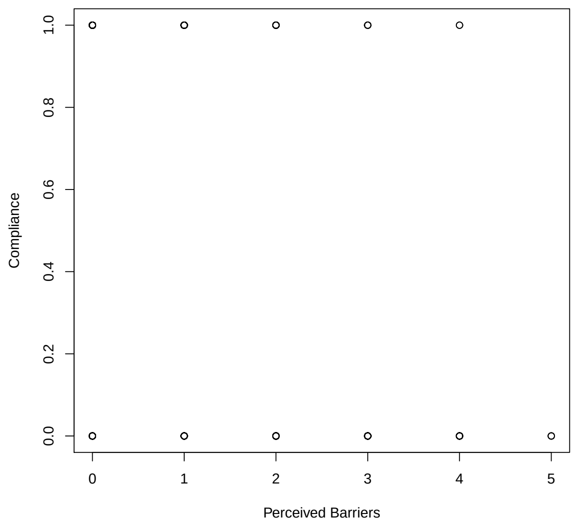
```
# Mammogram Data for Chapter 13
Mammogram.data<-read.table("C13E01DT.TXT", header=FALSE, sep="")
# name the variables
names(Mammogram.data) <- c("ID", "PHYSREC", "COMPLIANCE", "KNOWLEDG", "BENEFITS", "BARRIERS")
```

Even though CCAW do not plot the data, we will do so here.

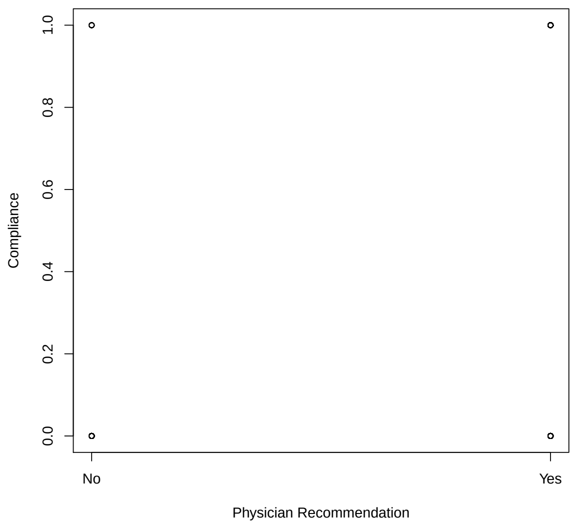
```
#scatterplot with continuous predictor
with(Mammogram.data, plot(KNOWLEDG, COMPLIANCE, xlab="Knowledge", ylab="Compliance"))

#scatterplot with categorical predictor
with(Mammogram.data, plot(PHYSREC, COMPLIANCE, xlab="Physician Recommendation", ylab="Compliance", xaxt='n', pc
axis(side=1, at=c(0,1), labels=c("No", "Yes"))
```

For a logistic regression, we need to used the `glm()` function with a logit link option for the `family` argument.

```
Mammogram.fit<-glm(COMPLIANCE~PHYSREC+KNOWLEDG+BENEFITS+BARRIERS, data=Mammogram.data,
        family=binomial(link = "logit"))
summary(Mammogram.fit)

##
## Call:
## glm(formula = COMPLIANCE ~ PHYSREC + KNOWLEDG + BENEFITS + BARRIERS,
##     family = binomial(link = "logit"), data = Mammogram.data)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.841  -0.839  -0.211   0.826   2.044
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -3.0512     1.3690   -2.23  0.02583 *
## PHYSREC       1.8423     0.4884    3.77  0.00016 ***
## KNOWLEDG     -0.0794     1.0736   -0.07  0.94105
## BENEFITS      0.5435     0.2426    2.24  0.02505 *
```



(a) Continuous Predictor



(b) Categorical Predictor

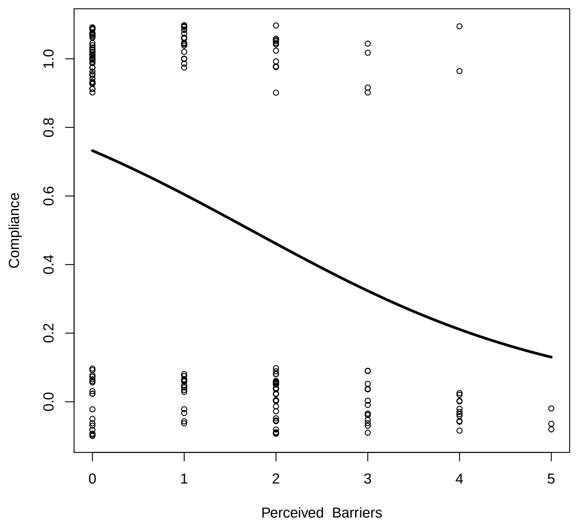
Figure 13.1: Plots of categorical outcomes

```
## BARRIERS     -0.5812     0.1660   -3.50  0.00046 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 226.47  on 163  degrees of freedom
## Residual deviance: 167.70  on 159  degrees of freedom
## AIC: 177.7
##
## Number of Fisher Scoring iterations: 5
```

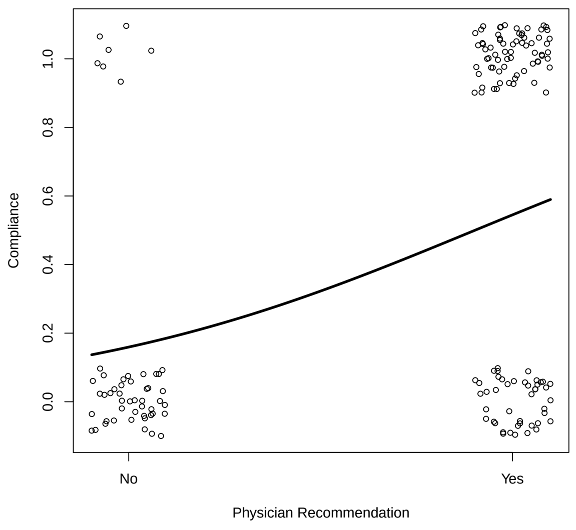
```
# Inverse logit function for graphing
invlogit<- function(x) {1/(1+exp(-x))}

with(Mammogram.data, plot(BARRIERS, COMPLIANCE, xlab="Perceived  Barriers",
ylab="Compliance",pch=1))
curve ( invlogit ( cbind (1,1, mean(Mammogram.data$KNOWLEDG), mean(Mammogram.data$BENEFITS), x) %*%
coef(Mammogram.fit)), add=TRUE, lwd=3)


with(Mammogram.data, plot(PHYSREC, COMPLIANCE, xlab="Physician Recommendation",
ylab="Compliance",pch=1, xaxt='n'))
axis(side=1, at=c(0,1), labels=c("No", "Yes"))
curve ( invlogit ( cbind (1,x, mean(Mammogram.data$KNOWLEDG), mean(Mammogram.data$BENEFITS),
mean(Mammogram.data$BARRIERS)) %*% coef(Mammogram.fit)), add=TRUE, lwd=3)
```

The `confint()` function will return the confidence intervals, but uses profile likelihoods. To obtain the CIs using the standard errors, use the `confint.default()` function.

```
confint.default(Mammogram.fit)

##                2.5 % 97.5 %
## (Intercept) -5.7343 -0.368
## PHYSREC      0.8851  2.799
## KNOWLEDG    -2.1835  2.025
## BENEFITS     0.0681  1.019
## BARRIERS    -0.9066 -0.256
```



(a) Continuous predictor.



(b) Categorical predictor.

Figure 13.2: Plots of categorical outcomes, with logistic regression lines. The full model was used for each plot holding all covariates constant at their mean values, except for physician recommendation which was held at 1 (i.e., physician recommended a mammography). Data points are jittered.

To obtain the odds ratios, exponentiate the regression coefficients and their CIs via the `exp()` function.

```
exp(cbind(OR = coef(Mammogram.fit), confint.default(Mammogram.fit)))

##                 OR   2.5 % 97.5 %
## (Intercept) 0.0473 0.00323  0.692
## PHYSREC     6.3110 2.42326 16.436
## KNOWLEDG    0.9237 0.11264  7.574
## BENEFITS    1.7220 1.07045  2.770
## BARRIERS    0.5592 0.40390  0.774
```

For the linear discriminant function, use the `lda()` function in the `MASS` package.

```
library(MASS)
Mammogram.lda<- lda(COMPLIANCE~PHYSREC+KNOWLEDG+BENEFITS+BARRIERS, data=Mammogram.data)
Mammogram.lda

## Call:
## lda(COMPLIANCE ~ PHYSREC + KNOWLEDG + BENEFITS + BARRIERS, data = Mammogram.data)
##
## Prior probabilities of groups:
##     0     1
## 0.537 0.463
##
## Group means:
##   PHYSREC KNOWLEDG BENEFITS BARRIERS
## 0   0.500    0.625     3.82    1.943
## 1   0.908    0.602     4.57    0.816
##
## Coefficients of linear discriminants:
##             LD1
## PHYSREC   1.419
## KNOWLEDG -0.312
## BENEFITS  0.295
## BARRIERS -0.416
```

The BaylorEdPsych package has a PseudoR2() to give pseudo R^2 and AIC values. (Note the the McFadden pseudo R^2 is similar to what CCAW call R^2_L)

```
library(BaylorEdPsych)
PseudoR2(Mammogram.fit)

##         McFadden     Adj.McFadden        Cox.Snell       Nagelkerke McKelvey.Zavoina
##            0.260            0.207            0.301            0.402            0.463
##           Effron            Count        Adj.Count              AIC    Corrected.AIC
##            0.300            0.689            0.329          177.696          178.076
```

To conduct the likelihood ratio test, we first need to fit a null model.

```
Mammogram.null<-glm(COMPLIANCE~1, data=Mammogram.data, family=binomial(link = "logit"))
```

Then, using the anova() function, we can test the null model against the model of interest giving *Chisq* for the test argument.

```
anova(Mammogram.null,Mammogram.fit, test="Chisq")

## Analysis of Deviance Table
##
## Model 1: COMPLIANCE ~ 1
## Model 2: COMPLIANCE ~ PHYSREC + KNOWLEDG + BENEFITS + BARRIERS
##   Resid. Df Resid. Dev Df Deviance        Pr(>Chi)
## 1       163        226
## 2       159        168  4     58.8 0.0000000000052 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

For full test of the model (i.e., comparing the null and model deviance), use the wald.test() from the aod package.

```
library(aod)

## Error in library(aod):  there is no package called 'aod'

wald.test(b = coef(Mammogram.fit), Sigma = vcov(Mammogram.fit), Terms = 2:5)

## Error in eval(expr, envir, enclos):  could not find function "wald.test"
```

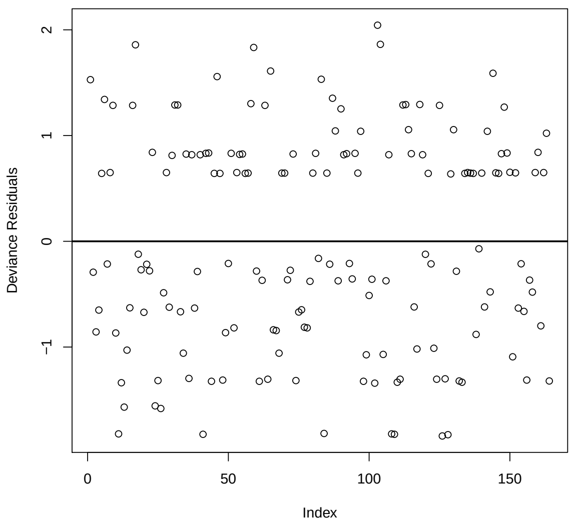
13.1.1 Diagnostics

There are multiple types of residuals to examine with logistics regression, CCAW mention two: Deviance residuals and Pearson residuals.

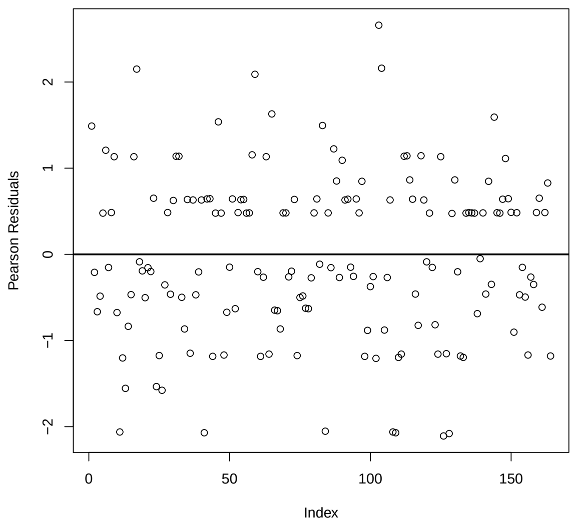
```
# Deviance residuals
Mammogram.dResid <- residuals(Mammogram.fit, type="dev")

# Pearson residuals
Mammogram.pResid <- residuals(Mammogram.fit, type="pear")
```

13.1.2 Classification

Based on the logistic regression model, we can obtain a probability of outcome for each respondent, π_i, and classify them if π_i is greater than some cutoff value. Such a table is given in Table 13.1.



(a) Deviance residuals



(b) Pearson residuals

Figure 13.3: Residual plots.

```r
cutoff <- 0.5
table(Mammogram.data$COMPLIANCE,fitted(Mammogram.fit)>=cutoff)
```

Table 13.1: Classification table

		Predicted Mammography	
		No	Yes
Actual	No	60	28
	Yes	23	53

13.2 Polytomous Logistic Regression

Steps to compliance data from CCAW Table 13.3.1.

```
# Steps Data for Chapter 13
steps.data<-read.table("c13e02dt.TXT", header=FALSE, sep="")
# name the variables
names(steps.data) <- c("ID", "STEPS", "INTERVEN")
```

```
# Examine data
with(steps.data, table(STEPS, INTERVEN))

##      INTERVEN
## STEPS  0  1
##     1 33 26
##     2 10 23
##     3  1  4
##     4  7 35
```

To do the three separate logistic regressions, we need to re-code to outcome to be a binary outcome.

```
# Recoding outcomes
# Mammogram vs. all other
steps.data$out1 <- ifelse(steps.data$STEPS==4,1,0)

# Mammogram apt. vs. contacting health professional or doing nothing
steps.data$out2 <- ifelse(steps.data$STEPS==3,1,ifelse(steps.data$STEPS==4,NA, 0))

# contacting health professional vs. doing nothing
steps.data$out3 <- ifelse(steps.data$STEPS==2,1,ifelse(steps.data$STEPS==4 | steps.data$STEPS==3,NA, 0))
```

Now, conduct the logistic regressions.

```
# Model 1
steps1.fit <- glm(out1~INTERVEN, data=steps.data, family=binomial(link = "logit"))
summary(steps1.fit)

##
## Call:
## glm(formula = out1 ~ INTERVEN, family = binomial(link = "logit"),
##     data = steps.data)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.007  -1.007  -0.543   1.358   1.993
##
## Coefficients:
##             Estimate Std. Error z value  Pr(>|z|)
## (Intercept)   -1.838      0.407   -4.52 0.0000063 ***
## INTERVEN       1.423      0.462    3.08     0.002 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 170.33  on 138  degrees of freedom
## Residual deviance: 159.08  on 137  degrees of freedom
## AIC: 163.1
##
## Number of Fisher Scoring iterations: 4
```

```
exp(coef(steps1.fit))

## (Intercept)    INTERVEN
##       0.159       4.151

PseudoR2(steps1.fit)

##         McFadden     Adj.McFadden        Cox.Snell       Nagelkerke McKelvey.Zavoina
##           0.0660           0.0308           0.0777           0.1100           0.1251
##           Effron            Count        Adj.Count              AIC    Corrected.AIC
##           0.0747               NA               NA         163.0808         163.1690

# Model 2
steps2.fit <- glm(out2~INTERVEN, data=steps.data, family=binomial(link = "logit"))
summary(steps2.fit)

##
## Call:
## glm(formula = out2 ~ INTERVEN, family = binomial(link = "logit"),
##     data = steps.data)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -0.396  -0.396  -0.396  -0.214   2.751
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -3.76       1.01   -3.72   0.0002 ***
## INTERVEN        1.26       1.14    1.10   0.2696
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 39.390  on 96  degrees of freedom
## Residual deviance: 37.908  on 95  degrees of freedom
##   (42 observations deleted due to missingness)
## AIC: 41.91
##
## Number of Fisher Scoring iterations: 6

exp(coef(steps2.fit))

## (Intercept)    INTERVEN
##      0.0233      3.5102

PseudoR2(steps2.fit)

##         McFadden     Adj.McFadden        Cox.Snell       Nagelkerke McKelvey.Zavoina
##           0.0376          -0.1147           0.0152           0.0455           0.1062
##           Effron            Count        Adj.Count              AIC    Corrected.AIC
##           0.0141               NA               NA          41.9077          42.0353

# Model 3
steps3.fit <- glm(out3~INTERVEN, data=steps.data, family=binomial(link = "logit"))
summary(steps3.fit)

##
## Call:
## glm(formula = out3 ~ INTERVEN, family = binomial(link = "logit"),
##     data = steps.data)
##
## Deviance Residuals:
```

```
##    Min      1Q  Median      3Q     Max
## -1.126  -1.126  -0.728   1.230   1.708
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -1.194      0.361   -3.31  0.00094 ***
## INTERVEN       1.071      0.461    2.33  0.02005 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 120.09  on 91  degrees of freedom
## Residual deviance: 114.39  on 90  degrees of freedom
##   (47 observations deleted due to missingness)
## AIC: 118.4
##
## Number of Fisher Scoring iterations: 4

exp(coef(steps3.fit))

## (Intercept)    INTERVEN
##       0.303       2.919

PseudoR2(steps3.fit)

##         McFadden     Adj.McFadden        Cox.Snell       Nagelkerke McKelvey.Zavoina
##          0.04749         -0.00247          0.06011          0.08247          0.07991
##           Effron            Count        Adj.Count              AIC    Corrected.AIC
##          0.06070               NA               NA        118.38664        118.52148
```

To do everything in one step using an ordered logistic regression, use the `polr()` function in the `MASS` package.

```
library(MASS)
steps.fit <- polr(STEPS ~ INTERVEN, data = steps.data, Hess = TRUE)
summary(steps.fit)

## Call:
## polr(formula = STEPS ~ INTERVEN, data = steps.data, Hess = TRUE)
##
## Coefficients:
##          Value Std. Error t value
## INTERVEN  1.46      0.354    4.13
##
## Intercepts:
##     Value Std. Error t value
## 1|2 0.601 0.288      2.086
## 2|3 1.693 0.319      5.308
## 3|4 1.872 0.325      5.761
##
## Residual Deviance: 311.41
## AIC: 319.41

# Odds Ratio
exp(cbind(OR = coef(steps.fit), confint.default(steps.fit)))

##            OR 2.5 % 97.5 %
## INTERVEN 4.32  2.16   8.66
```

Chapter 14

Random Coefficient Regression and Multilevel Models

Data from CCAW Table 14.2.1

```
# Disaggregated data
disag.data <- read.table("c14e01dt.txt")
names(disag.data) <- c("group", "caseingr", "treat", "treat.c", "motivat","motivatc", "pounds")
# Aggregated data
ag.data <- read.table("C14E02DT.txt")
names(ag.data) <- c("group", "treat", "size", "motmean", "pdsmean")
```

14.1 Analysis of clustered data with OLS regression

```
# Disaggregated, ignoring clustering
dis.fit <- lm(pounds~motivatc, data=disag.data)
summary(dis.fit)

##
## Call:
## lm(formula = pounds ~ motivatc, data = disag.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -9.061 -1.932 -0.061  2.210  9.939
##
## Coefficients:
##             Estimate Std. Error t value            Pr(>|t|)
## (Intercept)   15.003      0.156    96.4 <0.0000000000000002 ***
## motivatc       3.270      0.153    21.4 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.06 on 384 degrees of freedom
## Multiple R-squared:  0.545,Adjusted R-squared:  0.544
## F-statistic:  460 on 1 and 384 DF,  p-value: <0.0000000000000002
```

```
# Aggregated
ag.fit <- lm(pdsmean~motmean, data=ag.data)
summary(ag.fit)

##
## Call:
```

```
## lm(formula = pdsmean ~ motmean, data = ag.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -3.733 -1.257 -0.065  1.020  5.275
##
## Coefficients:
##             Estimate Std. Error t value   Pr(>|t|)
## (Intercept)    0.721      2.399    0.30       0.77
## motmean        4.162      0.686    6.07 0.00000046 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.92 on 38 degrees of freedom
## Multiple R-squared:  0.492,Adjusted R-squared:  0.479
## F-statistic: 36.8 on 1 and 38 DF,  p-value: 0.000000461
```

```
# Disaggregated, with dummy coded groups

# Make group 40 the reference group
disag.data$grp <- relevel(factor(disag.data$group), ref = 40)

disGrp.fit <- lm(pounds~motivatc+grp, data=disag.data)
summary(disGrp.fit)

##
## Call:
## lm(formula = pounds ~ motivatc + grp, data = disag.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -8.081 -1.593  0.031  1.751  7.342
##
## Coefficients:
##             Estimate Std. Error t value             Pr(>|t|)
## (Intercept) 15.26398    0.72211   21.14 < 0.0000000000000002 ***
## motivatc     3.11858    0.14329   21.76 < 0.0000000000000002 ***
## grp1        -1.19240    1.09777   -1.09               0.2781
## grp2         1.25354    1.12976    1.11               0.2680
## grp3        -1.40939    1.09469   -1.29               0.1988
## grp4        -3.56439    1.22314   -2.91               0.0038 **
## grp5         0.42394    1.28444    0.33               0.7416
## grp6        -0.59460    1.00289   -0.59               0.5536
## grp7        -1.48082    1.00243   -1.48               0.1405
## grp8        -3.33232    1.06750   -3.12               0.0020 **
## grp9        -3.34582    1.04176   -3.21               0.0014 **
## grp10       -1.76196    1.09706   -1.61               0.1092
## grp11        2.07084    1.28436    1.61               0.1078
## grp12       -0.85010    1.22314   -0.70               0.4875
## grp13        0.46561    1.16946    0.40               0.6908
## grp14       -0.27040    0.98622   -0.27               0.7841
## grp15       -1.38011    1.06608   -1.29               0.1963
## grp16       -3.26249    1.28436   -2.54               0.0115 *
## grp17        3.49664    1.06690    3.28               0.0012 **
## grp18       -1.00990    1.22183   -0.83               0.4091
## grp19       -0.77318    1.02449   -0.75               0.4509
## grp20        6.12335    1.22551    5.00           0.00000093 ***
## grp21       -0.23983    1.37665   -0.17               0.8618
## grp22       -0.90281    1.12868   -0.80               0.4243
## grp23       -0.60900    1.12865   -0.54               0.5898
## grp24        0.62640    1.12844    0.56               0.5792
## grp25        0.00246    1.09500    0.00               0.9982
## grp26        0.80339    1.13072    0.71               0.4779
## grp27        0.31656    1.04546    0.30               0.7622
## grp28        1.79680    1.28595    1.40               0.1632
## grp29        0.81656    1.28736    0.63               0.5263
## grp30       -0.72125    1.36940   -0.53               0.5987
## grp31        0.46775    1.04220    0.45               0.6539
## grp32       -0.86224    1.01496   -0.85               0.3962
## grp33        2.45688    1.12982    2.17               0.0303 *
## grp34        0.84483    1.06720    0.79               0.4291
## grp35        1.74314    1.06668    1.63               0.1031
## grp36       -1.59827    1.00341   -1.59               0.1121
## grp37       -0.19878    1.00712   -0.20               0.8437
## grp38        2.44370    1.28497    1.90               0.0580 .
## grp39       -2.93064    1.17793   -2.49               0.0133 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.6 on 345 degrees of freedom
## Multiple R-squared:  0.704,Adjusted R-squared:  0.669
## F-statistic: 20.5 on 40 and 345 DF,  p-value: <0.0000000000000002
```

14.2 Random coefficient regression

There are multiple packages in **R** that will handle multilevel/random coefficient models (Bliese, 2013). This presentation will use the `lme4` package (cf. Gelman & Hill, 2006).

```
library(lme4)

# Unconditional cell means model
m0.fit<-lmer(pounds~ 1 + (1 | group), data=disag.data)
summary(m0.fit)

## Linear mixed model fit by REML ['lmerMod']
## Formula: pounds ~ 1 + (1 | group)
##    Data: disag.data
##
## REML criterion at convergence: 2220
##
## Scaled residuals:
##    Min     1Q Median     3Q    Max
## -4.124 -0.590 -0.023  0.618  2.807
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  group    (Intercept)  4.91    2.22
##  Residual             16.07    4.01
## Number of obs: 386, groups:  group, 40
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)   15.115      0.409      37

# ICC
m0.resid <- attr(VarCorr(m0.fit), "sc")^2
m0.bet.resid<-VarCorr(m0.fit)$group[1]
(ICC<-m0.bet.resid/(m0.bet.resid+ m0.resid))

## [1] 0.234
```

```
# Random coefficient regression
m1.fit<-lmer(pounds~motivatc + (1 + motivatc | group), data=disag.data)
summary(m1.fit)

## Linear mixed model fit by REML ['lmerMod']
## Formula: pounds ~ motivatc + (1 + motivatc | group)
##    Data: disag.data
##
## REML criterion at convergence: 1874
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.8851 -0.6735 -0.0321  0.6409  2.4689
##
## Random effects:
##  Groups   Name        Variance Std.Dev. Corr
##  group    (Intercept) 2.397    1.548
##           motivatc    0.933    0.966    0.39
##  Residual             5.933    2.436
## Number of obs: 386, groups:  group, 40
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)   15.138      0.280    54.1
## motivatc       3.118      0.211    14.8
##
## Correlation of Fixed Effects:
##          (Intr)
## motivatc 0.250
```

```
# Multilevel interaction
m2.fit<-lmer(pounds~motivatc + treat.c + motivatc:treat.c +(1 + motivatc | group), data=disag.data)
summary(m2.fit)

## Linear mixed model fit by REML ['lmerMod']
## Formula: pounds ~ motivatc + treat.c + motivatc:treat.c + (1 + motivatc |      group)
##    Data: disag.data
##
## REML criterion at convergence: 1859
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.0932 -0.6431 -0.0446  0.6532  2.4224
##
## Random effects:
##  Groups   Name        Variance Std.Dev. Corr
##  group    (Intercept) 1.967    1.403
##           motivatc    0.556    0.746    0.14
##  Residual             5.933    2.436
## Number of obs: 386, groups:  group, 40
##
## Fixed effects:
##                  Estimate Std. Error t value
## (Intercept)        15.166      0.259    58.5
## motivatc            3.130      0.185    17.0
## treat.c             1.528      0.529     2.9
## motivatc:treat.c    1.245      0.377     3.3
##
## Correlation of Fixed Effects:
##             (Intr) motvtc tret.c
## motivatc     0.070
## treat.c     -0.003  0.043
## mtvtc:trt.c  0.043 -0.007  0.053

# slope-intercept covariance
VarCorr(m2.fit)

##  Groups   Name        Std.Dev. Corr
##  group    (Intercept) 1.403
##           motivatc    0.746    0.14
##  Residual             2.436
```

```
# OLS interaction
m2OLS.fit<-lm(pounds~motivatc + treat.c + motivatc:treat.c, data=disag.data)
summary(m2OLS.fit)

##
## Call:
## lm(formula = pounds ~ motivatc + treat.c + motivatc:treat.c,
##     data = disag.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -9.882 -1.968  0.017  1.955  8.203
##
## Coefficients:
##                  Estimate Std. Error t value             Pr(>|t|)
## (Intercept)        15.105      0.148  102.39 < 0.0000000000000002 ***
## motivatc            3.330      0.145   22.97 < 0.0000000000000002 ***
## treat.c             1.578      0.301    5.24           0.00000027 ***
## motivatc:treat.c    1.446      0.300    4.82           0.00000208 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.87 on 382 degrees of freedom
## Multiple R-squared:  0.601,Adjusted R-squared:  0.598
## F-statistic:  192 on 3 and 382 DF,  p-value: <0.0000000000000002
```

Chapter 15

Longitudinal Regression Methods

Data from CCAW Table 15.3.1

```
# Repeated measures data
rm2.data <- read.table("C1502DT.txt", header=TRUE)
```

15.1 Repeated measures analysis of variance

```
# Reshape data
rm2long.data <- reshape(rm2.data, direction = 'long',
varying = list(c('RPERF1', 'RPERF2', 'RPERF3','RPERF4')), v.name="RPERF")
rm2long.data$GROUP <- factor(rm2long.data$GROUP)
rm2long.data$time <- factor(rm2long.data$time)
rm2long.data$id <- factor(rm2long.data$id)
```

```
# Repeated Measures ANOVA
rm2.ANOVA.fit <-  aov(RPERF ~ GROUP*time + Error(id/(GROUP + time)), data=rm2long.data)
summary(rm2.ANOVA.fit)

##
## Error: id
##            Df Sum Sq Mean Sq F value     Pr(>F)
## GROUP       1  18284   18284    27.2 0.00000046 ***
## Residuals 198 132973     672
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Error: id:time
##             Df Sum Sq Mean Sq F value           Pr(>F)
## time         3   9713    3238    20.1 0.0000000000021 ***
## GROUP:time   3   5971    1990    12.3 0.0000000764849 ***
## Residuals  594  95767     161
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
model.tables(rm2.ANOVA.fit,"means")

## Tables of means
## Grand mean
##
## 53.3
##
```

```
##  GROUP
##          0      1
##      48.25  57.83
## rep 380.00 420.00
##
##  time
##          1      2      3      4
##      48.83  51.08  55.67  57.55
## rep 200.00 200.00 200.00 200.00
##
##  GROUP:time
##      time
## GROUP 1     2     3     4
##   0    40.3  44.7  51.3  56.8
##   rep  95.0  95.0  95.0  95.0
##   1    56.6  56.9  59.6  58.2
##   rep 105.0 105.0 105.0 105.0
```

```
# Means
with(rm2long.data, tapply(RPERF, GROUP, sum))

##     0     1
## 18337 24288
```

Data from CCAW Table 15.3.3

```
# Repeated measures data
rm3.data <- read.table("C1503DT.txt", header=TRUE)
```

```
# Reshape data
rm3long.data <- reshape(rm3.data, direction = 'long', varying = list(c('Y1', 'Y2', 'Y3')),v.name="Y")
rm3long.data$GROUP <- factor(rm3long.data$GROUP)
rm3long.data$time <- factor(rm3long.data$time)
rm3long.data$id <- factor(rm3long.data$id)
```

```
# Repeated Measures ANOVA
rm3.ANOVA.fit <-  aov(Y ~ GROUP*time + Error(id/(time)), data=rm3long.data)
summary(rm3.ANOVA.fit)

##
## Error: id
##           Df Sum Sq Mean Sq F value   Pr(>F)
## GROUP      3   62.7   20.90    11.8 0.000016 ***
## Residuals 36   63.7    1.77
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Error: id:time
##            Df Sum Sq Mean Sq F value     Pr(>F)
## time        2   66.2    33.1   17.78 0.00000053 ***
## GROUP:time  6   92.5    15.4    8.28 0.00000079 ***
## Residuals  72  134.0     1.9
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Means
with(rm3long.data, tapply(Y, GROUP:time, mean))
```

```
##   1:1   1:2   1:3   2:1   2:2   2:3   3:1   3:2   3:3   4:1   4:2   4:3
## 10.02  9.80  9.93 10.26 10.00 13.01  9.46 12.36 10.59  9.98 12.58 13.26

# SD
with(rm3long.data, tapply(Y, GROUP:time, sd))

##   1:1   1:2   1:3   2:1   2:2   2:3   3:1   3:2   3:3   4:1   4:2   4:3
## 0.686 0.762 1.281 1.047 1.091 0.446 0.938 1.183 1.084 1.039 0.748 3.420
```

Multilevel data Data for CCAW Table 15.4.1

```
# multilvel
rm4.data <- read.table("C1504DT.txt", header=TRUE)
```

15.2 Multilevel Regression of Individual Changes Over Time

```
library(lme4)

# YA Model 1
ya1.fit<-lmer(YA~ 1 + (1 | ID), data=rm4.data)
summary(ya1.fit)

## Linear mixed model fit by REML ['lmerMod']
## Formula: YA ~ 1 + (1 | ID)
##    Data: rm4.data
##
## REML criterion at convergence: 1282
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.0066 -0.7500 -0.0405  0.7073  2.1914
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  ID       (Intercept) 2.52     1.59
##  Residual             3.02     1.74
## Number of obs: 300, groups:  ID, 60
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)   11.949      0.228    52.4
```

```
# YA Model 2

ya2.fit<-lmer(YA~ 1 + (1 + TIME  | ID), data=rm4.data)
summary(ya2.fit)

## Linear mixed model fit by REML ['lmerMod']
## Formula: YA ~ 1 + (1 + TIME | ID)
##    Data: rm4.data
##
## REML criterion at convergence: 896
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.2529 -0.5474  0.0215  0.5287  2.3259
##
## Random effects:
```

```
##  Groups   Name        Variance Std.Dev. Corr
##  ID       (Intercept) 17.851   4.225
##           TIME         1.118   1.057    -0.91
##  Residual              0.228   0.478
## Number of obs: 300, groups:  ID, 60
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)   12.608      0.228    55.4

# slope-intercept covariance
VarCorr(ya2.fit)

##  Groups   Name        Std.Dev. Corr
##  ID       (Intercept) 4.225
##           TIME        1.057    -0.91
##  Residual             0.478
```

```
# YA Model 3

ya3.fit<-lmer(YA~ TIME + (1 + TIME  | ID), data=rm4.data)
summary(ya3.fit)

## Linear mixed model fit by REML ['lmerMod']
## Formula: YA ~ TIME + (1 + TIME | ID)
##    Data: rm4.data
##
## REML criterion at convergence: 687
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.1093 -0.5677 -0.0041  0.5648  2.3730
##
## Random effects:
##  Groups   Name        Variance Std.Dev. Corr
##  ID       (Intercept) 3.28127  1.8114
##           TIME        0.00947  0.0973   -0.28
##  Residual             0.22831  0.4778
## Number of obs: 300, groups:  ID, 60
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)   8.7900     0.2426    36.2
## TIME          1.0531     0.0232    45.4
##
## Correlation of Fixed Effects:
##      (Intr)
## TIME -0.347

# slope-intercept covariance
VarCorr(ya3.fit)

##  Groups   Name        Std.Dev. Corr
##  ID       (Intercept) 1.8114
##           TIME        0.0973   -0.28
##  Residual             0.4778
```

```
# YB Model 1

yb1.fit<-lmer(YB~ 1 + (1  | ID), data=rm4.data)
summary(yb1.fit)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: YB ~ 1 + (1 | ID)
##    Data: rm4.data
##
## REML criterion at convergence: 1440
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.2453 -0.4096 -0.0936  0.3319  2.6235
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  ID       (Intercept) 2.77     1.66
##  Residual             5.54     2.35
## Number of obs: 300, groups:  ID, 60
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)   12.958      0.254      51
```

```
# YB Model 2

yb2.fit<-lmer(YB~ 1 + (1 + TIME  | ID), data=rm4.data)
summary(yb2.fit)

## Linear mixed model fit by REML ['lmerMod']
## Formula: YB ~ 1 + (1 + TIME | ID)
##    Data: rm4.data
##
## REML criterion at convergence: 934
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.0842 -0.5685  0.0038  0.5388  2.3829
##
## Random effects:
##  Groups   Name        Variance Std.Dev. Corr
##  ID       (Intercept) 11.963   3.459
##           TIME         2.124   1.457    -0.87
##  Residual              0.231   0.481
## Number of obs: 300, groups:  ID, 60
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)   11.983      0.223    53.8

# slope-intercept covariance
VarCorr(yb2.fit)

##  Groups   Name        Std.Dev. Corr
##  ID       (Intercept) 3.459
##           TIME        1.457    -0.87
##  Residual             0.481
```

```
# YB Model 3

yb3.fit<-lmer(YB~ TIME + (1 + TIME  | ID), data=rm4.data)
summary(yb3.fit)

## Linear mixed model fit by REML ['lmerMod']
## Formula: YB ~ TIME + (1 + TIME | ID)
##    Data: rm4.data
##
## REML criterion at convergence: 893
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.0321 -0.5536 -0.0164  0.5194  2.3829
##
## Random effects:
##  Groups   Name        Variance Std.Dev. Corr
##  ID       (Intercept) 7.265    2.695
##           TIME        1.033    1.016    -0.77
##  Residual             0.231    0.481
## Number of obs: 300, groups:  ID, 60
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)    9.798      0.354   27.68
## TIME           1.053      0.133    7.94
##
## Correlation of Fixed Effects:
##      (Intr)
## TIME -0.777

# slope-intercept covariance
VarCorr(yb3.fit)

##  Groups   Name        Std.Dev. Corr
##  ID       (Intercept) 2.695
##           TIME        1.016    -0.77
##  Residual             0.481
```

```
# YB Model 4

yb4.fit<-lmer(YB~ TIME + GROUP + (1 + TIME  | ID), data=rm4.data)
summary(yb4.fit)

## Linear mixed model fit by REML ['lmerMod']
## Formula: YB ~ TIME + GROUP + (1 + TIME | ID)
##    Data: rm4.data
##
## REML criterion at convergence: 889
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.1182 -0.5662 -0.0002  0.5458  2.3963
##
## Random effects:
##  Groups   Name        Variance Std.Dev. Corr
##  ID       (Intercept) 23.322   4.829
##           TIME         1.033   1.016    -0.94
##  Residual              0.231   0.481
## Number of obs: 300, groups:  ID, 60
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)    7.359      0.663   11.09
## TIME           1.053      0.133    7.94
## GROUP          4.879      0.435   11.22
##
## Correlation of Fixed Effects:
```

```
##       (Intr) TIME
## TIME  -0.886
## GROUP -0.328  0.000

# slope-intercept covariance
VarCorr(yb4.fit)

##  Groups   Name        Std.Dev. Corr
##  ID       (Intercept) 4.829
##           TIME        1.016    -0.94
##  Residual             0.481
```

```
# YB Model 5

yb5.fit<-lmer(YB~ TIME*GROUP + (1 + TIME  | ID), data=rm4.data)
summary(yb5.fit)

## Linear mixed model fit by REML ['lmerMod']
## Formula: YB ~ TIME * GROUP + (1 + TIME | ID)
##    Data: rm4.data
##
## REML criterion at convergence: 687
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -1.9356 -0.5880 -0.0075  0.5447  2.4064
##
## Random effects:
##  Groups   Name        Variance Std.Dev. Corr
##  ID       (Intercept) 3.2280   1.7967
##           TIME        0.0097   0.0985   -0.43
##  Residual             0.2314   0.4811
## Number of obs: 300, groups:  ID, 60
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)  11.8053     0.3407    34.6
## TIME          0.0499     0.0331     1.5
## GROUP        -4.0137     0.4818    -8.3
## TIME:GROUP    2.0064     0.0468    42.9
##
## Correlation of Fixed Effects:
##            (Intr) TIME   GROUP
## TIME       -0.430
## GROUP      -0.707  0.304
## TIME:GROUP  0.304 -0.707 -0.430

# slope-intercept covariance
VarCorr(yb5.fit)

##  Groups   Name        Std.Dev. Corr
##  ID       (Intercept) 1.7967
##           TIME        0.0985   -0.43
##  Residual             0.4811
```

15.3 Latent growth models: Structural equation model representation of multilevel data

To analyze the data in a SEM framework, the data needs to be reshaped horizontally.

```
# Reshape data
rm4Wide.data <- reshape(rm4.data, direction="wide", v.names=c("YA", "YB"), idvar="ID", timevar="TIME")
head(rm4Wide.data)

##    ID GROUP YA.1  YB.1 YA.2  YB.2  YA.3  YB.3  YA.4  YB.4 YA.5  YB.5
## 1   1     0 7.51 10.01 8.04  9.54  9.51  9.85 10.87 10.37 11.5 10.02
## 6   2     0 7.62 10.12 7.40  8.90  9.62  9.80 10.64 10.14 12.2 10.69
## 11  3     0 7.13  9.63 8.17  9.67  9.13 10.12  9.70  9.20 11.7 10.15
## 16  4     0 8.10 10.60 9.01 10.51 10.10 10.46 10.95 10.45 12.2 10.67
## 21  5     0 7.76 10.26 8.18  9.68  9.76 10.24  9.74  9.24 11.4  9.94
## 26  6     0 7.50 10.00 8.40  9.90  9.50 10.18 10.71 10.21 11.3  9.83
```

We will use `lavaan` to fit the growth model, using the `growth()` function. First, the YA models.

```
library(lavaan)

latGrowthYA0.model <- '
i =~ 1*YA.1 + 1*YA.2 + 1*YA.3 + 1*YA.4 + 1*YA.5

YA.1~~r*YA.1
YA.2~~r*YA.2
YA.3~~r*YA.3
YA.4~~r*YA.4
YA.5~~r*YA.5
i~~0*i
'

latGrowthYA0.fit <- growth(latGrowthYA0.model, data=rm4Wide.data)
summary(latGrowthYA0.fit)

## lavaan (0.5-20) converged normally after  18 iterations
##
##   Number of observations                            60
##
##   Estimator                                         ML
##   Minimum Function Test Statistic              705.213
##   Degrees of freedom                                18
##   P-value (Chi-square)                           0.000
##
## Parameter Estimates:
##
##   Information                                 Expected
##   Standard Errors                             Standard
##
## Latent Variables:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##   i =~
##     YA.1              1.000
##     YA.2              1.000
##     YA.3              1.000
##     YA.4              1.000
##     YA.5              1.000
##
## Intercepts:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##     YA.1              0.000
##     YA.2              0.000
##     YA.3              0.000
##     YA.4              0.000
##     YA.5              0.000
##     i                11.949    0.135   88.356    0.000
```

```
##
## Variances:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##     YA.1       (r)    5.487    0.448   12.247    0.000
##     YA.2       (r)    5.487    0.448   12.247    0.000
##     YA.3       (r)    5.487    0.448   12.247    0.000
##     YA.4       (r)    5.487    0.448   12.247    0.000
##     YA.5       (r)    5.487    0.448   12.247    0.000
##     i                 0.000
```

```
latGrowthYA1.model <- '
i =~ 1*YA.1 + 1*YA.2 + 1*YA.3 + 1*YA.4 + 1*YA.5

YA.1~~r*YA.1
YA.2~~r*YA.2
YA.3~~r*YA.3
YA.4~~r*YA.4
YA.5~~r*YA.5
'

latGrowthYA1.fit <- growth(latGrowthYA1.model, data=rm4Wide.data)
summary(latGrowthYA1.fit)
```

```
## lavaan (0.5-20) converged normally after  25 iterations
##
##   Number of observations                            60
##
##   Estimator                                         ML
##   Minimum Function Test Statistic              623.852
##   Degrees of freedom                                17
##   P-value (Chi-square)                           0.000
##
## Parameter Estimates:
##
##   Information                                 Expected
##   Standard Errors                             Standard
##
## Latent Variables:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##   i =~
##     YA.1              1.000
##     YA.2              1.000
##     YA.3              1.000
##     YA.4              1.000
##     YA.5              1.000
##
## Intercepts:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##     YA.1              0.000
##     YA.2              0.000
##     YA.3              0.000
##     YA.4              0.000
##     YA.5              0.000
##     i                11.949    0.226   52.840    0.000
##
## Variances:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##     YA.1       (r)    3.023    0.276   10.954    0.000
##     YA.2       (r)    3.023    0.276   10.954    0.000
##     YA.3       (r)    3.023    0.276   10.954    0.000
##     YA.4       (r)    3.023    0.276   10.954    0.000
##     YA.5       (r)    3.023    0.276   10.954    0.000
##     i                 2.464    0.563    4.377    0.000
```

```
latGrowthYA2.model <- '
i =~ 1*YA.1 + 1*YA.2 + 1*YA.3 + 1*YA.4 + 1*YA.5
s =~ 0*YA.1 + 1*YA.2 + 2*YA.3 + 3*YA.4 + 4*YA.5

YA.1~~r*YA.1
YA.2~~r*YA.2
YA.3~~r*YA.3
YA.4~~r*YA.4
YA.5~~r*YA.5
s~0*1
'

latGrowthYA2.fit <- growth(latGrowthYA2.model, data=rm4Wide.data)
summary(latGrowthYA2.fit)

## lavaan (0.5-20) converged normally after  56 iterations
##
##   Number of observations                            60
##
##   Estimator                                         ML
##   Minimum Function Test Statistic              238.275
##   Degrees of freedom                                15
##   P-value (Chi-square)                           0.000
##
## Parameter Estimates:
##
##   Information                                 Expected
##   Standard Errors                             Standard
##
## Latent Variables:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##   i =~
##     YA.1              1.000
##     YA.2              1.000
##     YA.3              1.000
##     YA.4              1.000
##     YA.5              1.000
##   s =~
##     YA.1              0.000
##     YA.2              1.000
##     YA.3              2.000
##     YA.4              3.000
##     YA.5              4.000
##
## Covariances:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##   i ~~
##     s                -2.949    0.598   -4.935    0.000
##
## Intercepts:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##     s                 0.000
##     YA.1              0.000
##     YA.2              0.000
##     YA.3              0.000
##     YA.4              0.000
##     YA.5              0.000
##     i                12.608    0.226   55.865    0.000
##
## Variances:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##     YA.1       (r)    0.228    0.024    9.487    0.000
##     YA.2       (r)    0.228    0.024    9.487    0.000
##     YA.3       (r)    0.228    0.024    9.487    0.000
##     YA.4       (r)    0.228    0.024    9.487    0.000
##     YA.5       (r)    0.228    0.024    9.487    0.000
##     i                10.783    1.994    5.408    0.000
##     s                 1.118    0.208    5.367    0.000
```

```
latGrowthYA3.model <- '
i =~ 1*YA.1 + 1*YA.2 + 1*YA.3 + 1*YA.4 + 1*YA.5
s =~ 0*YA.1 + 1*YA.2 + 2*YA.3 + 3*YA.4 + 4*YA.5

YA.1~~r*YA.1
YA.2~~r*YA.2
YA.3~~r*YA.3
YA.4~~r*YA.4
YA.5~~r*YA.5
'

latGrowthYA3.fit <- growth(latGrowthYA3.model, data=rm4Wide.data)
summary(latGrowthYA3.fit)

## lavaan (0.5-20) converged normally after  41 iterations
##
##   Number of observations                            60
##
##   Estimator                                         ML
##   Minimum Function Test Statistic               23.401
##   Degrees of freedom                                14
##   P-value (Chi-square)                           0.054
##
## Parameter Estimates:
##
##   Information                                 Expected
##   Standard Errors                             Standard
##
## Latent Variables:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##   i =~
##     YA.1              1.000
##     YA.2              1.000
##     YA.3              1.000
##     YA.4              1.000
##     YA.5              1.000
##   s =~
##     YA.1              0.000
##     YA.2              1.000
##     YA.3              2.000
##     YA.4              3.000
##     YA.5              4.000
##
## Covariances:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##   i ~~
##     s                -0.038    0.043   -0.872    0.383
##
## Intercepts:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##     YA.1              0.000
##     YA.2              0.000
##     YA.3              0.000
##     YA.4              0.000
##     YA.5              0.000
##     i                 9.843    0.234   42.131    0.000
##     s                 1.053    0.023   45.772    0.000
##
## Variances:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##     YA.1       (r)    0.228    0.024    9.487    0.000
##     YA.2       (r)    0.228    0.024    9.487    0.000
##     YA.3       (r)    0.228    0.024    9.487    0.000
##     YA.4       (r)    0.228    0.024    9.487    0.000
##     YA.5       (r)    0.228    0.024    9.487    0.000
##     i                 3.138    0.598    5.247    0.000
##     s                 0.009    0.006    1.423    0.155
```

Next, the YB models, which includes a group factor.

```
latGrowthYB0.model <- '
i =~ 1*YB.1 + 1*YB.2 + 1*YB.3 + 1*YB.4 + 1*YB.5

YB.1~~c(r,r)*YB.1
YB.2~~c(r,r)*YB.2
YB.3~~c(r,r)*YB.3
YB.4~~c(r,r)*YB.4
YB.5~~c(r,r)*YB.5
i~~0*i
'

latGrowthYB0.fit <- growth(latGrowthYB0.model, data=rm4Wide.data, group="GROUP",
group.equal=c("means"))
summary(latGrowthYB0.fit)

## lavaan (0.5-20) converged normally after  24 iterations
##
##   Number of observations per group
##   0                                                 30
##   1                                                 30
##
##   Estimator                                         ML
##   Minimum Function Test Statistic              854.646
##   Degrees of freedom                                38
##   P-value (Chi-square)                           0.000
##
## Chi-square for each group:
##
##   0                                            370.387
##   1                                            484.259
##
## Parameter Estimates:
##
##   Information                                 Expected
##   Standard Errors                             Standard
##
##
## Group 1 [0]:
##
## Latent Variables:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##   i =~
##     YB.1              1.000
##     YB.2              1.000
##     YB.3              1.000
##     YB.4              1.000
##     YB.5              1.000
##
## Intercepts:
```

```
##                     Estimate  Std.Err  Z-value  P(>|z|)
##     YB.1               0.000
##     YB.2               0.000
##     YB.3               0.000
##     YB.4               0.000
##     YB.5               0.000
##     i       (.17.)    12.958    0.166   78.154    0.000
##
## Variances:
##                     Estimate  Std.Err  Z-value  P(>|z|)
##     YB.1       (r)     8.247    0.673   12.247    0.000
##     YB.2       (r)     8.247    0.673   12.247    0.000
##     YB.3       (r)     8.247    0.673   12.247    0.000
##     YB.4       (r)     8.247    0.673   12.247    0.000
##     YB.5       (r)     8.247    0.673   12.247    0.000
##     i                  0.000
##
##
## Group 2 [1]:
##
## Latent Variables:
##                     Estimate  Std.Err  Z-value  P(>|z|)
##   i =~
##     YB.1               1.000
##     YB.2               1.000
##     YB.3               1.000
##     YB.4               1.000
##     YB.5               1.000
##
## Intercepts:
##                     Estimate  Std.Err  Z-value  P(>|z|)
##     YB.1               0.000
##     YB.2               0.000
##     YB.3               0.000
##     YB.4               0.000
##     YB.5               0.000
##     i       (.17.)    12.958    0.166   78.154    0.000
##
## Variances:
##                     Estimate  Std.Err  Z-value  P(>|z|)
##     YB.1       (r)     8.247    0.673   12.247    0.000
##     YB.2       (r)     8.247    0.673   12.247    0.000
##     YB.3       (r)     8.247    0.673   12.247    0.000
##     YB.4       (r)     8.247    0.673   12.247    0.000
##     YB.5       (r)     8.247    0.673   12.247    0.000
##     i                  0.000
```

```
latGrowthYB1.model <- '
i =~ 1*YB.1 + 1*YB.2 + 1*YB.3 + 1*YB.4 + 1*YB.5

YB.1~~c(r,r)*YB.1
YB.2~~c(r,r)*YB.2
YB.3~~c(r,r)*YB.3
YB.4~~c(r,r)*YB.4
YB.5~~c(r,r)*YB.5
'

latGrowthYB1.fit <- growth(latGrowthYB1.model, data=rm4Wide.data, group="GROUP",
group.equal=c("means", "lv.variances"))
summary(latGrowthYB1.fit)

## lavaan (0.5-20) converged normally after  29 iterations
```

```
##
##   Number of observations per group
##   0                                                 30
##   1                                                 30
##
##   Estimator                                         ML
##   Minimum Function Test Statistic              809.528
##   Degrees of freedom                                37
##   P-value (Chi-square)                           0.000
##
## Chi-square for each group:
##
##   0                                            309.797
##   1                                            499.731
##
## Parameter Estimates:
##
##   Information                                 Expected
##   Standard Errors                             Standard
##
##
## Group 1 [0]:
##
## Latent Variables:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##   i =~
##     YB.1              1.000
##     YB.2              1.000
##     YB.3              1.000
##     YB.4              1.000
##     YB.5              1.000
##
## Intercepts:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##     YB.1              0.000
##     YB.2              0.000
##     YB.3              0.000
##     YB.4              0.000
##     YB.5              0.000
##     i       (.17.)   12.958    0.252   51.398    0.000
##
## Variances:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##     YB.1       (r)    5.542    0.506   10.954    0.000
##     YB.2       (r)    5.542    0.506   10.954    0.000
##     YB.3       (r)    5.542    0.506   10.954    0.000
##     YB.4       (r)    5.542    0.506   10.954    0.000
##     YB.5       (r)    5.542    0.506   10.954    0.000
##     i       (.11.)    2.705    0.704    3.845    0.000
##
##
## Group 2 [1]:
##
## Latent Variables:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##   i =~
##     YB.1              1.000
##     YB.2              1.000
##     YB.3              1.000
##     YB.4              1.000
##     YB.5              1.000
##
## Intercepts:
```

```
##                    Estimate  Std.Err  Z-value  P(>|z|)
##     YB.1              0.000
##     YB.2              0.000
##     YB.3              0.000
##     YB.4              0.000
##     YB.5              0.000
##     i       (.17.)   12.958    0.252   51.398    0.000
##
## Variances:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##     YB.1       (r)    5.542    0.506   10.954    0.000
##     YB.2       (r)    5.542    0.506   10.954    0.000
##     YB.3       (r)    5.542    0.506   10.954    0.000
##     YB.4       (r)    5.542    0.506   10.954    0.000
##     YB.5       (r)    5.542    0.506   10.954    0.000
##     i       (.11.)    2.705    0.704    3.845    0.000
```

```
latGrowthYB2.model <- '
i =~ 1*YB.1 + 1*YB.2 + 1*YB.3 + 1*YB.4 + 1*YB.5
s =~ 0*YB.1 + 1*YB.2 + 2*YB.3 + 3*YB.4 + 4*YB.5

YB.1~~c(r,r)*YB.1
YB.2~~c(r,r)*YB.2
YB.3~~c(r,r)*YB.3
YB.4~~c(r,r)*YB.4
YB.5~~c(r,r)*YB.5
s ~ 0*1



'

latGrowthYB2.fit <- growth(latGrowthYB2.model, data=rm4Wide.data, group="GROUP",
group.equal=c("means", "lv.variances", "lv.covariances"))
summary(latGrowthYB2.fit)

## lavaan (0.5-20) converged normally after  36 iterations
##
##   Number of observations per group
##   0                                                 30
##   1                                                 30
##
##   Estimator                                         ML
##   Minimum Function Test Statistic              303.246
##   Degrees of freedom                                35
##   P-value (Chi-square)                           0.000
##
## Chi-square for each group:
##
##   0                                            131.726
##   1                                            171.520
##
## Parameter Estimates:
##
##   Information                                 Expected
##   Standard Errors                             Standard
##
##
## Group 1 [0]:
##
## Latent Variables:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##   i =~
##     YB.1              1.000
##     YB.2              1.000
##     YB.3              1.000
##     YB.4              1.000
##     YB.5              1.000
##   s =~
##     YB.1              0.000
##     YB.2              1.000
##     YB.3              2.000
##     YB.4              3.000
##     YB.5              4.000
##
## Covariances:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##   i ~~
##     s       (.19.)   -2.262    0.531   -4.257    0.000
##
## Intercepts:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##     s                 0.000
##     YB.1              0.000
##     YB.2              0.000
##     YB.3              0.000
##     YB.4              0.000
##     YB.5              0.000
##     i       (.25.)   11.983    0.221   54.276    0.000
##
## Variances:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##     YB.1       (r)    0.231    0.024    9.487    0.000
##     YB.2       (r)    0.231    0.024    9.487    0.000
##     YB.3       (r)    0.231    0.024    9.487    0.000
##     YB.4       (r)    0.231    0.024    9.487    0.000
##     YB.5       (r)    0.231    0.024    9.487    0.000
##     i       (.17.)    5.267    0.987    5.336    0.000
##     s       (.18.)    2.124    0.392    5.418    0.000
##
##
## Group 2 [1]:
##
## Latent Variables:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##   i =~
##     YB.1              1.000
##     YB.2              1.000
##     YB.3              1.000
##     YB.4              1.000
##     YB.5              1.000
##   s =~
##     YB.1              0.000
##     YB.2              1.000
##     YB.3              2.000
##     YB.4              3.000
##     YB.5              4.000
##
## Covariances:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##   i ~~
##     s       (.19.)   -2.262    0.531   -4.257    0.000
##
## Intercepts:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##     s                 0.000
##     YB.1              0.000
##     YB.2              0.000
##     YB.3              0.000
##     YB.4              0.000
##     YB.5              0.000
##     i       (.25.)   11.983    0.221   54.276    0.000
##
## Variances:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##     YB.1       (r)    0.231    0.024    9.487    0.000
##     YB.2       (r)    0.231    0.024    9.487    0.000
##     YB.3       (r)    0.231    0.024    9.487    0.000
##     YB.4       (r)    0.231    0.024    9.487    0.000
##     YB.5       (r)    0.231    0.024    9.487    0.000
##     i       (.17.)    5.267    0.987    5.336    0.000
##     s       (.18.)    2.124    0.392    5.418    0.000
```

```
latGrowthYB3.model <- '
i =~ 1*YB.1 + 1*YB.2 + 1*YB.3 + 1*YB.4 + 1*YB.5
s =~ 0*YB.1 + 1*YB.2 + 2*YB.3 + 3*YB.4 + 4*YB.5

YB.1~~c(r,r)*YB.1
YB.2~~c(r,r)*YB.2
YB.3~~c(r,r)*YB.3
YB.4~~c(r,r)*YB.4
YB.5~~c(r,r)*YB.5
'

latGrowthYB3.fit <- growth(latGrowthYB3.model, data=rm4Wide.data, group="GROUP",
group.equal=c("means", "lv.variances", "lv.covariances"))
summary(latGrowthYB3.fit)

## lavaan (0.5-20) converged normally after  44 iterations
##
##   Number of observations per group
##   0                                                 30
##   1                                                 30
##
##   Estimator                                         ML
##   Minimum Function Test Statistic              259.643
##   Degrees of freedom                                34
##   P-value (Chi-square)                           0.000
##
## Chi-square for each group:
##
##   0                                            139.375
##   1                                            120.268
##
## Parameter Estimates:
##
##   Information                                 Expected
##   Standard Errors                             Standard
##
##
## Group 1 [0]:
##
## Latent Variables:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##   i =~
##     YB.1              1.000
##     YB.2              1.000
##     YB.3              1.000
##     YB.4              1.000
##     YB.5              1.000
##   s =~
##     YB.1              0.000
##     YB.2              1.000
##     YB.3              2.000
##     YB.4              3.000
##     YB.5              4.000
##
## Covariances:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##   i ~~
##     s       (.18.)   -1.070    0.304   -3.524    0.000
##
## Intercepts:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##     YB.1              0.000
##     YB.2              0.000
##     YB.3              0.000
##     YB.4              0.000
##     YB.5              0.000
##     i       (.24.)   10.852    0.262   41.391    0.000
##     s       (.25.)    1.053    0.132    8.006    0.000
##
## Variances:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##     YB.1       (r)    0.231    0.024    9.487    0.000
##     YB.2       (r)    0.231    0.024    9.487    0.000
##     YB.3       (r)    0.231    0.024    9.487    0.000
##     YB.4       (r)    0.231    0.024    9.487    0.000
##     YB.5       (r)    0.231    0.024    9.487    0.000
##     i       (.16.)    3.985    0.753    5.292    0.000
##     s       (.17.)    1.015    0.190    5.355    0.000
##
##
## Group 2 [1]:
##
## Latent Variables:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##   i =~
##     YB.1              1.000
##     YB.2              1.000
##     YB.3              1.000
##     YB.4              1.000
##     YB.5              1.000
##   s =~
##     YB.1              0.000
##     YB.2              1.000
##     YB.3              2.000
##     YB.4              3.000
##     YB.5              4.000
##
## Covariances:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##   i ~~
##     s       (.18.)   -1.070    0.304   -3.524    0.000
##
## Intercepts:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##     YB.1              0.000
##     YB.2              0.000
##     YB.3              0.000
##     YB.4              0.000
##     YB.5              0.000
##     i       (.24.)   10.852    0.262   41.391    0.000
##     s       (.25.)    1.053    0.132    8.006    0.000
##
## Variances:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##     YB.1       (r)    0.231    0.024    9.487    0.000
##     YB.2       (r)    0.231    0.024    9.487    0.000
##     YB.3       (r)    0.231    0.024    9.487    0.000
##     YB.4       (r)    0.231    0.024    9.487    0.000
##     YB.5       (r)    0.231    0.024    9.487    0.000
##     i       (.16.)    3.985    0.753    5.292    0.000
##     s       (.17.)    1.015    0.190    5.355    0.000
```

```
latGrowthYB4.model <- '
i =~ 1*YB.1 + 1*YB.2 + 1*YB.3 + 1*YB.4 + 1*YB.5
s =~ 0*YB.1 + 1*YB.2 + 2*YB.3 + 3*YB.4 + 4*YB.5

YB.1~~c(r,r)*YB.1
YB.2~~c(r,r)*YB.2
YB.3~~c(r,r)*YB.3
YB.4~~c(r,r)*YB.4
YB.5~~c(r,r)*YB.5
s~c(sl,sl)*1

'

latGrowthYB4.fit <- growth(latGrowthYB4.model, data=rm4Wide.data, group="GROUP",
group.equal=c("lv.variances", "lv.covariances"))
summary(latGrowthYB4.fit)

## lavaan (0.5-20) converged normally after  71 iterations
##
##   Number of observations per group
##   0                                                 30
##   1                                                 30
##
##   Estimator                                         ML
##   Minimum Function Test Statistic              255.788
##   Degrees of freedom                                33
##   P-value (Chi-square)                           0.000
##
## Chi-square for each group:
##
##   0                                            136.249
##   1                                            119.539
##
## Parameter Estimates:
##
##   Information                                 Expected
##   Standard Errors                             Standard
##
##
## Group 1 [0]:
##
## Latent Variables:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##   i =~
##     YB.1              1.000
##     YB.2              1.000
##     YB.3              1.000
##     YB.4              1.000
##     YB.5              1.000
##   s =~
##     YB.1              0.000
##     YB.2              1.000
##     YB.3              2.000
##     YB.4              3.000
##     YB.5              4.000
##
## Covariances:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##   i ~~
##     s       (.19.)   -3.517    0.686   -5.126    0.000
##
## Intercepts:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##     s         (sl)    1.053    0.132    8.006    0.000
##     YB.1              0.000
##     YB.2              0.000
##     YB.3              0.000
##     YB.4              0.000
##     YB.5              0.000
##     i                 8.412    0.543   15.482    0.000
##
## Variances:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##     YB.1       (r)    0.231    0.024    9.487    0.000
##     YB.2       (r)    0.231    0.024    9.487    0.000
##     YB.3       (r)    0.231    0.024    9.487    0.000
##     YB.4       (r)    0.231    0.024    9.487    0.000
##     YB.5       (r)    0.231    0.024    9.487    0.000
##     i       (.17.)   14.833    2.734    5.426    0.000
##     s       (.18.)    1.015    0.190    5.355    0.000
##
##
## Group 2 [1]:
##
## Latent Variables:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##   i =~
##     YB.1              1.000
##     YB.2              1.000
##     YB.3              1.000
##     YB.4              1.000
##     YB.5              1.000
##   s =~
##     YB.1              0.000
##     YB.2              1.000
##     YB.3              2.000
##     YB.4              3.000
##     YB.5              4.000
##
## Covariances:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##   i ~~
##     s       (.19.)   -3.517    0.686   -5.126    0.000
##
## Intercepts:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##     s         (sl)    1.053    0.132    8.006    0.000
##     YB.1              0.000
##     YB.2              0.000
##     YB.3              0.000
##     YB.4              0.000
##     YB.5              0.000
##     i                13.291    0.543   24.461    0.000
##
## Variances:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##     YB.1       (r)    0.231    0.024    9.487    0.000
##     YB.2       (r)    0.231    0.024    9.487    0.000
##     YB.3       (r)    0.231    0.024    9.487    0.000
##     YB.4       (r)    0.231    0.024    9.487    0.000
##     YB.5       (r)    0.231    0.024    9.487    0.000
##     i       (.17.)   14.833    2.734    5.426    0.000
##     s       (.18.)    1.015    0.190    5.355    0.000
```

```
latGrowthYB5.model <- '
i =~ 1*YB.1 + 1*YB.2 + 1*YB.3 + 1*YB.4 + 1*YB.5
s =~ 0*YB.1 + 1*YB.2 + 2*YB.3 + 3*YB.4 + 4*YB.5

YB.1~~c(r,r)*YB.1
YB.2~~c(r,r)*YB.2
YB.3~~c(r,r)*YB.3
YB.4~~c(r,r)*YB.4
YB.5~~c(r,r)*YB.5
'

latGrowthYB5.fit <- growth(latGrowthYB5.model, data=rm4Wide.data, group="GROUP",
group.equal=c("lv.variances", "lv.covariances"))
summary(latGrowthYB5.fit)

## lavaan (0.5-20) converged normally after  56 iterations
##
##   Number of observations per group
##   0                                                 30
##   1                                                 30
##
##   Estimator                                         ML
##   Minimum Function Test Statistic               46.552
##   Degrees of freedom                                32
##   P-value (Chi-square)                           0.046
##
## Chi-square for each group:
##
##   0                                             27.286
##   1                                             19.266
##
## Parameter Estimates:
##
##   Information                                 Expected
##   Standard Errors                             Standard
##
##
## Group 1 [0]:
##
## Latent Variables:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##   i =~
##     YB.1              1.000
##     YB.2              1.000
##     YB.3              1.000
##     YB.4              1.000
##     YB.5              1.000
##   s =~
##     YB.1              0.000
##     YB.2              1.000
##     YB.3              2.000
##     YB.4              3.000
##     YB.5              4.000
##
## Covariances:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##   i ~~
##     s       (.18.)   -0.063    0.043   -1.449    0.147
##
## Intercepts:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##     YB.1              0.000
##     YB.2              0.000
##     YB.3              0.000
##     YB.4              0.000
##     YB.5              0.000
##     i                11.855    0.322   36.780    0.000
##     s                 0.050    0.033    1.534    0.125
##
## Variances:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##     YB.1       (r)    0.231    0.024    9.487    0.000
##     YB.2       (r)    0.231    0.024    9.487    0.000
##     YB.3       (r)    0.231    0.024    9.487    0.000
##     YB.4       (r)    0.231    0.024    9.487    0.000
##     YB.5       (r)    0.231    0.024    9.487    0.000
##     i       (.16.)    2.978    0.569    5.231    0.000
##     s       (.17.)    0.009    0.006    1.369    0.171
##
##
## Group 2 [1]:
##
## Latent Variables:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##   i =~
##     YB.1              1.000
##     YB.2              1.000
##     YB.3              1.000
##     YB.4              1.000
##     YB.5              1.000
##   s =~
##     YB.1              0.000
##     YB.2              1.000
##     YB.3              2.000
##     YB.4              3.000
##     YB.5              4.000
##
## Covariances:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##   i ~~
##     s       (.18.)   -0.063    0.043   -1.449    0.147
##
## Intercepts:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##     YB.1              0.000
##     YB.2              0.000
##     YB.3              0.000
##     YB.4              0.000
##     YB.5              0.000
##     i                 9.848    0.322   30.553    0.000
##     s                 2.056    0.033   63.207    0.000
```

```
##
## Variances:
##                    Estimate  Std.Err  Z-value  P(>|z|)
##     YB.1       (r)    0.231    0.024    9.487    0.000
##     YB.2       (r)    0.231    0.024    9.487    0.000
##     YB.3       (r)    0.231    0.024    9.487    0.000
##     YB.4       (r)    0.231    0.024    9.487    0.000
##     YB.5       (r)    0.231    0.024    9.487    0.000
##     i       (.16.)    2.978    0.569    5.231    0.000
##     s       (.17.)    0.009    0.006    1.369    0.171
```

References

Bliese, P. D. (2013). *Multilevel modeling in R (2.5): A brief introduction to R, the multilevel package and the nlme package* (Tech. Rep.).

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed.). Mahwah, NJ: Erlbaum.

Cohen, J., Cohen, P., West, S. G., & Aiken, L. S. (2003). *Applied multiple regression/correlation analysis for the behavioral sciences* (3rd ed.). Mahwah, NJ: Lawrence Erlbaum.

Cook, R. D. (1977). Detection of influential observation in linear regression. *Technometrics*, *19*, 15-18.

Efron, B., & Tibshirani, R. J. (1994). *An introduction to the bootstrap.* London: Chapman Hall.

Gelman, A., & Hill, J. (2006). *Data analysis using regression and multilevel/hierarchical models.* New York, NY: Cambridge.

Harris, B. (1988). Tetrachoric correlation coefficient. In L. Kotz & N. L. Johnson (Eds.), *Encyclopedia of statistical sciences* (Vol. 9, p. 223-225). New York: Wiley.

Olkin, I., & Finn, J. (1995). Correlations redux. *Psychological Bulletin*, *118*, 155-164. doi: 10.1037/0033-2909.118.1.155

R Development Core Team. (2015). ***R**: A language and environment for statistical computing* [Computer program]. Vienna, Austria: **R** Foundation for Statistical Computing.

Spearman, C. E. (1904). The proof and measurement of association between two things. *The American Journal of Psychology*, *15*, 72-101. doi: 10.2307/1412159

Trexler, J. C., & Travis, J. (1993). Nontraditional regression analyses. *Ecology*, *74*, 1629-1637. doi: 10.2307/1939921

Wickham, H. (2009). *ggplot2: Elegant graphics for data analysis* . New York, NY: Springer.

Index