Linear Models with R

Emphasis on Basics of Multiple Regression
To accompany Introductory Statistics Classes at the University at Albany.

Bruce Dudek

2026-02-07

Table of contents

Preface

Linear Modeling, in it's most rudimentary form, is also termed multiple regression. This document provides a template for execution of most of the basic analyses associated with this methodology. It is intended for students of the APSY510/511 introductory statistics sequence at the University at Albany, but can be a standalone document for others learning to use R for data analysis. The level of the document is targeted to an audience of researchers in training who are simultaneously learning linear modeling/regression theory and R programming. Some introduction to regression modeling with the `lm` function previously had been covered for simple regression and can be found in an accompanying document. The current document extends that work to a model with two IVs as an extensive illustration and then briefly covers models with more than two IVs.

Initially, the document works through extensive details of modeling with two Independent Variables to keep the conceptual development simple. The data set used is one already covered extensively with manual computations and SPSS implementation. Some extension to models with more than two IVs is also included in the "extensions" chapter.

The document emphasizes models where all variables are numeric/quantitative. Categorical IVs are covered in later documents, although one brief illustration is included in this document.

This book/monograph was created with Quarto, and was built largely using **rmarkdown** (Allaire et al., 2020) and **knitr** (Xie, 2015). RStudio (RStudio Team, 2015) was used for all writing and programming.

1 Introduction and Goals

This document lays out basic strategies for linear modeling in R. It is structured as a reflection of what, in Social, Behavioral, and Life Sciences is called Multiple Regression. It approaches linear modeling where one outcome variable (dependent variable) is predicted from multiple independent variables. Each variable is a quantitative measurement. The document presumes a background in the basics of multiple regression from both equational and conceptual perspectives, including the roles of partial and semi-partial correlation. The document is intended for students in the APSY510/511 course sequence at the University at Albany, but can be more generally applicable.

Initially, the approach here will use a multiple regression problem with only two independent variables. This permits comparability for a companion presentation on implementation of multiple regression with SPSS that used the same data set. The data set is the Cohen, Cohen, West and Aiken textbook chapter 3 example on faculty salaries, publications and citations (Cohen, Cohen, West, & Aiken, 2003). Later chapters extend the approach to problems with more than two independent variables (predictors, or "features").

The flow of the document moves from univariate description, to bivariate description, to all aspects of linear modeling. It extends to topics of evaluation of assumptions for inferential tests, influence analysis, and model criticism. Categorical predictors are only briefly discussed/analyzed. That topic is found in later documents.

The emphasis is narrowly placed on implementation of the standard OLS methods in R. Conceptual rationales for the approach, alternative methods (e.g., bayesian), variable selection strategies, model building, and the role of regression in causality assessment are treated briefly or elsewhere although a brief introduction to Bayes Factors is included here. The document does not address nonparametric or semi-parametric regression such as quantile regression nor does it address curvilinear regression.

1.1 A note on the R Programming environment

All of the analyses and graphical displays found in this document were produced in R. Usually, the document shows the relevant R code for each topic. The purpose of the document is has a primary focus on the how-to in R, but also emphasizes the conceptual progression related to understanding linear modeling in the simplest of multiple regression applications, the two-IV model. The document can be an extensive template for R usage in these types of analyses.

Some extension to models with larger numbers of IVs is also included. To that end, all the code is available both in this document and one other source. In the spirit of reproducible and open source research, this document was created in **rmarkdown** and **Quarto**. The qmd files contain ALL of the R code required to reproduce the analyses and figures contained in the document.

Graphs are drawn with **ggplot2**, base system graphics, and a convenient 3D surface plotting capability from the **plot3D** package. Analyses are extensively reliant on the base system `lm` function. Additional analyses use other packages and BCD-created functions introduced as the document progresses.

1.2 Required Packages

Several packages are required for the work in this document.

```r
library(BayesFactor)
library(bcdstats)
library(boot)
library(broom)
library(car)
library(GGally)
library(ggExtra)
library(ggplot2)
library(ggthemes)
library(grid)
library(gvlma)
library(gt)
library(HH)
library(knitr)
library(lattice)
library(lmtest)
library(MASS)
library(moments)
library(nortest)
library(olsrr)
library(psych)
library(plot3D)
library(plot3Drgl)
library(plyr)
library(rcompanion)
library(rmarkdown)
library(sandwich)
library(tseries)
library(UsingR)
library(yhat)
library(ggfortify)
library(Metrics)
library(MLmetrics)
library(paletteer)
```

Package citations for packages loaded here (in the above order): **BayesFactor** (Morey & Rouder, 2018), **bcdstats** (Dudek, 2026), **boot** (Canty & Ripley, 2019), **broom** (Robinson & Hayes, 2020), **car** (Fox, Weisberg, & Price, 2020), **GGally** (Schloerke et al., 2020), **ggExtra** (Attali & Baker, 2019), **ggplot2** (Wickham et al., 2020), **ggthemes** (Arnold, 2019), **grid** (Auguie, 2017), **gvlma** (Edsel A. Pena & Slate, 2019), **gt** (Iannone, Cheng, & Schloerke, 2019), **HH** (Heiberger, 2020), **knitr** (Xie, 2020), **lattice** (Sarkar, 2020), **lmtest** (Hothorn, Zeileis, Farebrother, & Cummins, 2019), **MASS** (Ripley, 2019), **moments** (Komsta & Novomestky, 2015), **nortest** (Gross & Ligges, 2015), **olsrr** (Hebbali, 2020), **psych** (Revelle, 2020), **plot3D** (Soetaert, 2019), **plot3Drgl**, (Soetaert, 2016), **plyr** (Wickham, 2020), **rcompanion** (Mangiafico, 2020), **rmarkdown** (Allaire et al., 2020), **sandwich** (Zeileis & Lumley, 2019), **tseries** (Trapletti & Hornik, 2019), **UsingR** (Trapletti & Hornik, 2019), **yhat** (Nimon, Oswald, & Roberts., 2013), **ggfortify** (Horikoshi & Tang, 2018), **Metrics** (Hamner & Frasco, 2018), **MLmetrics** (Yan, 2024), **paletteer** (file., 2024)

The **bcdstats** package can be installed with instructions found at its github repository:

https://github.com/bcdudek/bcdstats

2 Import the Data Set and Perform Numerical and Graphical EDA

Cohen, Cohen, West and Aiken (2003) have presented a data set where, ostensibly, faculty salaries in a university department are predicted from several IVs (old data, low salaries!). For the detailed example in this document two IVS are used. Both are quantitative: number of publications (pubs) and number of citations (cits). In this illustration all three variables are quantitative and can be thought of as measured on ratio scales (although pubs and cits are integer). The data are in a .csv file. It contains the three variables of interest plus three more. A subject/case number variable, time since PhD degree awarded, and sex. The file is "cohen.csv". The elaboration of this two-IV example is intended to parallel and extend the implementation previously done with SPSS for the same variables. The data file is available cohen.csv.

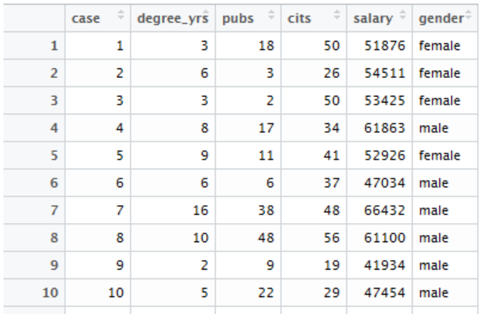
2.1 Read the data

The data file is read and a data frame is produced in the typical manner for .csv files that contain a header row with variable names. We also need to establish that our quantitative variables are read, by R, as numeric variables. We can examine the data frame with the 'View' function but I just show the first chunk of the data frame with a screen capture here. Later in this document the additional IVs, degree_yrs, and gender will be considered, but most analyses just use pubs and cits.

Finally, the data frame is "attached" so that easier naming of variables can be accomplished in the many of the initial R functions to be employed. Although this is often not a recommended approach in R, we will not encounter the downside in this document. The data frame will be "detached" prior to use in `lm` functions.

```r
cohen1 <- read.csv("data/cohen.csv", stringsAsFactors=T)
cohen1$degree_yrs <- as.numeric(cohen1$degree_yrs)
cohen1$pubs <- as.numeric(cohen1$pubs)
cohen1$cits <- as.numeric(cohen1$cits)
cohen1$salary <- as.numeric(cohen1$salary)
#View(cohen1)
attach(cohen1)
```

Here is a screen capture of what the first few lines of the data frame look like:



	case	degree_yrs	pubs	cits	salary	gender
1	1	3	18	50	51876	female
2	2	6	3	26	54511	female
3	3	3	2	50	53425	female
4	4	8	17	34	61863	male
5	5	9	11	41	52926	female
6	6	6	6	37	47034	male
7	7	16	38	48	66432	male
8	8	10	48	56	61100	male
9	9	2	9	19	41934	male
10	10	5	22	29	47454	male

2.2 Graphical and Numerical Description of the Variables

The best practices strategy we have put in place prioritizes exploratory data analysis at the outset. "Get to know your Data." This involves graphical and numerical summaries. The exact approaches taken here are somewhat arbitrary. There are many ways to draw useful graphs in R and an equally large number of ways to obtain numerical descriptive information. The functions used here are similar or identical to those used for previous topics in the course, beginning with the location test problems and progressing through simple regression and bivariate correlation.

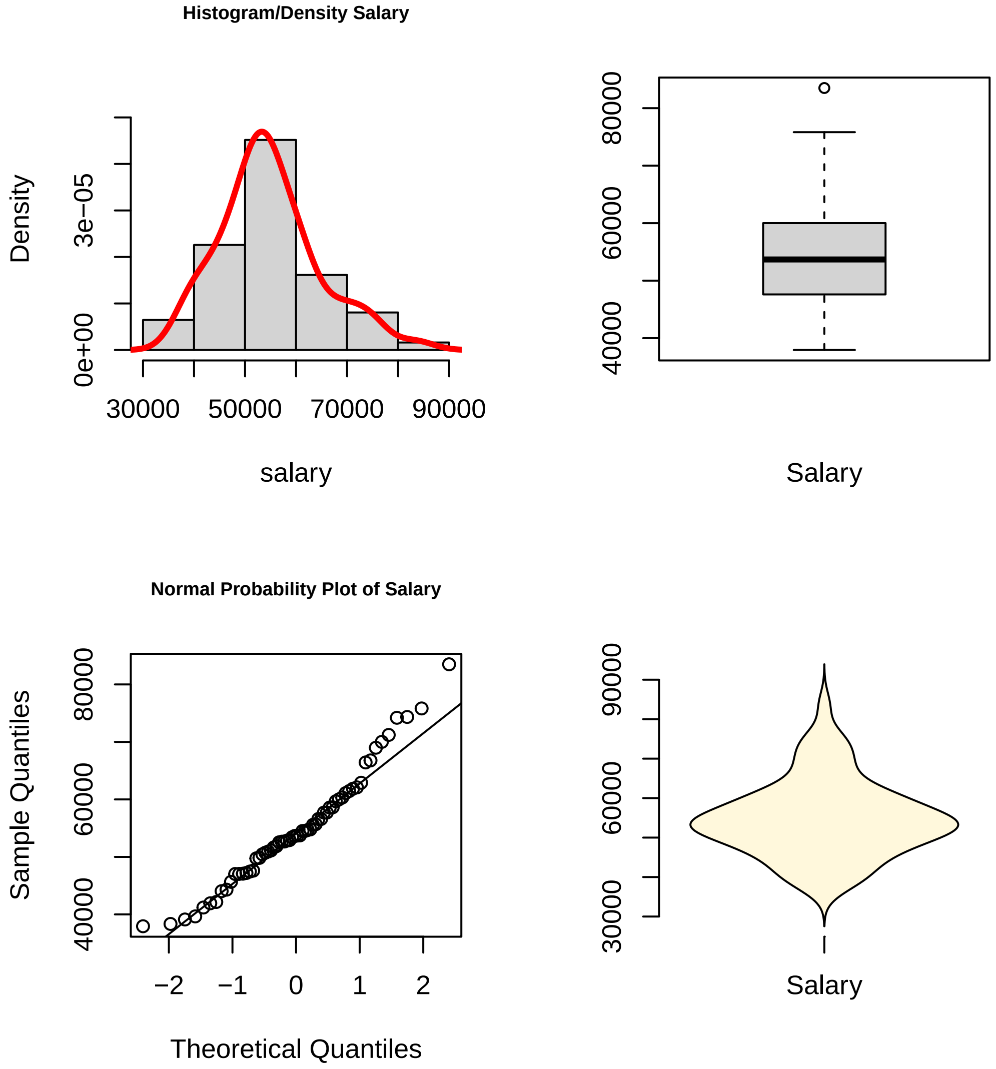
2.2.1 Univariate Distributional Displays for each of the three Variables, salary, publications, and citations

Each of our three variables is examined here in several ways, using base R graphics. First, I have used base system graphics to plot a few standard exploratory graphs. The basic set of four figures includes the frequency histogram with a kernel density function overlaid, a qq normal probability plot, box plot, and a violinplot.

Salary, the DV is first.

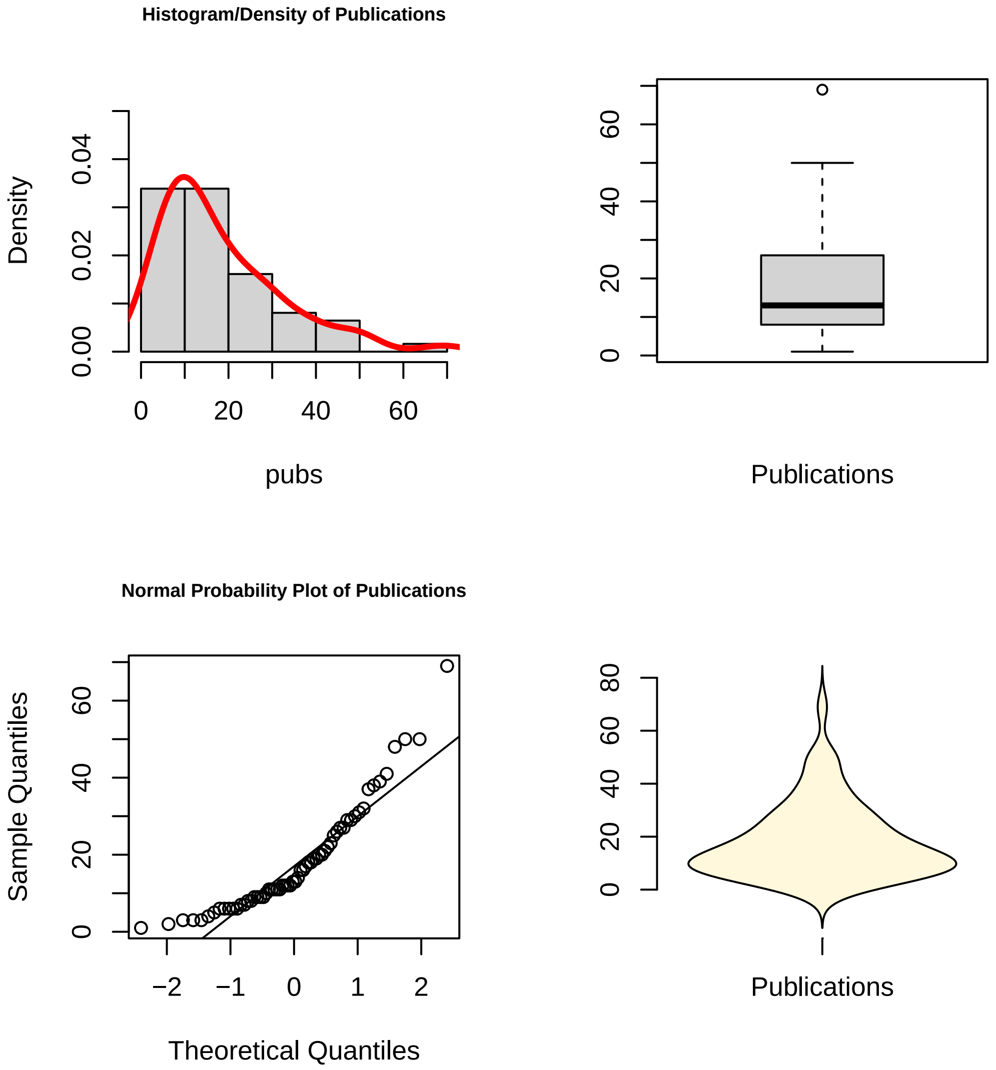
```
#win.graph(8,6) # use quartz() on MAC OS
layout(matrix(c(1,2,3,4),2,2)) #optional 4 graphs/page
hist(salary,probability=TRUE,breaks=6,
     ylim= c(0, 1.2*max(density(salary)$y)),
     main="Histogram/Density Salary",
     cex.main=.7)
lines(density(salary),col= "red",lwd = 3)
qqnorm(salary,main="Normal Probability Plot of Salary",
       cex.main=.7);qqline(salary)
```

```
boxplot(salary,xlab="Salary",ylab="")
UsingR::violinplot(salary,col="cornsilk",names="Salary")
```

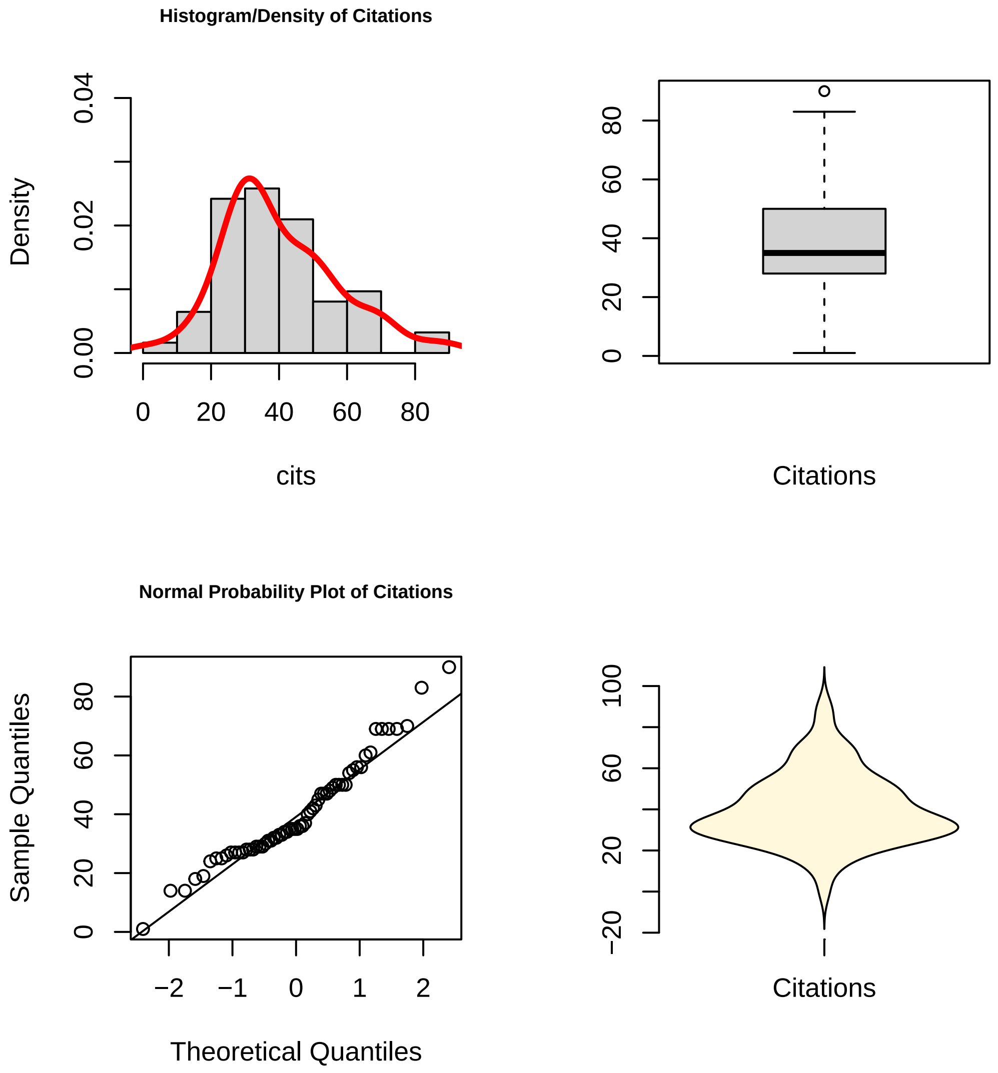


Then, publications and citations, the IV's.

```
#win.graph(8,6) # use quartz() on MAC OS
layout(matrix(c(1,2,3,4),2,2)) #optional 4 graphs/page
hist(pubs,probability=TRUE,
     ylim= c(0, 1.5*max(density(pubs)$y)),
     main="Histogram/Density of Publications",
     cex.main=.7)
lines(density(pubs),col = "red",lwd = 3)
qqnorm(pubs,main="Normal Probability Plot of Publications",
       cex.main=.7);qqline(pubs)
boxplot(pubs,xlab="Publications",ylab="")
UsingR::violinplot(pubs,col="cornsilk",names="Publications")
```

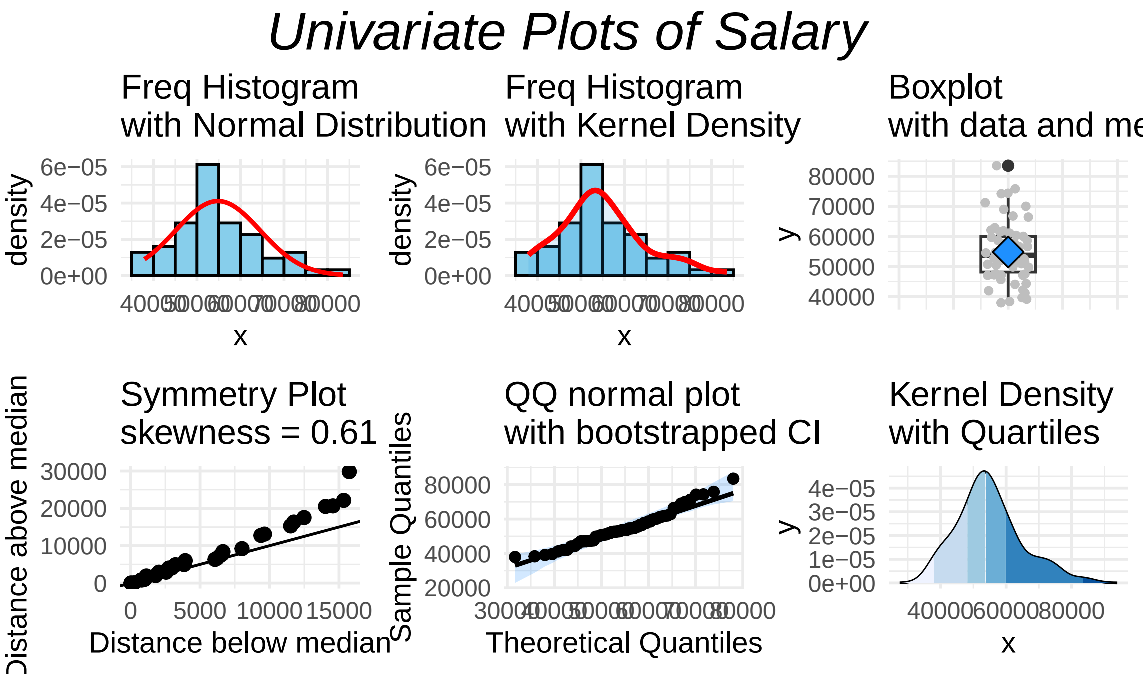

Histogram/Density of Publications
Density
0.00
0.02
0.04
0
20
40
60
pubs
0
20
40
60
Publications
Normal Probability Plot of Publications
Sample Quantiles
0
20
40
60
−2
−1
0
1
2
Theoretical Quantiles
0
20
40
60
80
Publications

```r
#win.graph(8,6) # use quartz() on MAC OS
layout(matrix(c(1,2,3,4),2,2)) #optional 4 graphs/page
hist(cits,probability=TRUE,
     ylim= c(0, 1.5*max(density(cits)$y)),
     main="Histogram/Density of Citations",
     cex.main=.7)
lines(density(cits),col = "red",lwd = 3)
qqnorm(cits,main="Normal Probability Plot of Citations",
       cex.main=.7);qqline(cits)
boxplot(cits,xlab="Citations",ylab="")
UsingR::violinplot(cits,col="cornsilk",names="Citations")
```


Histogram/Density of Citations
Density
0.00
0.02
0.04
0
20
40
60
80
cits
0
20
40
60
80
Citations
Normal Probability Plot of Citations
Sample Quantiles
0
20
40
60
80
−2
−1
0
1
2
Theoretical Quantiles
−20
20
60
100
Citations

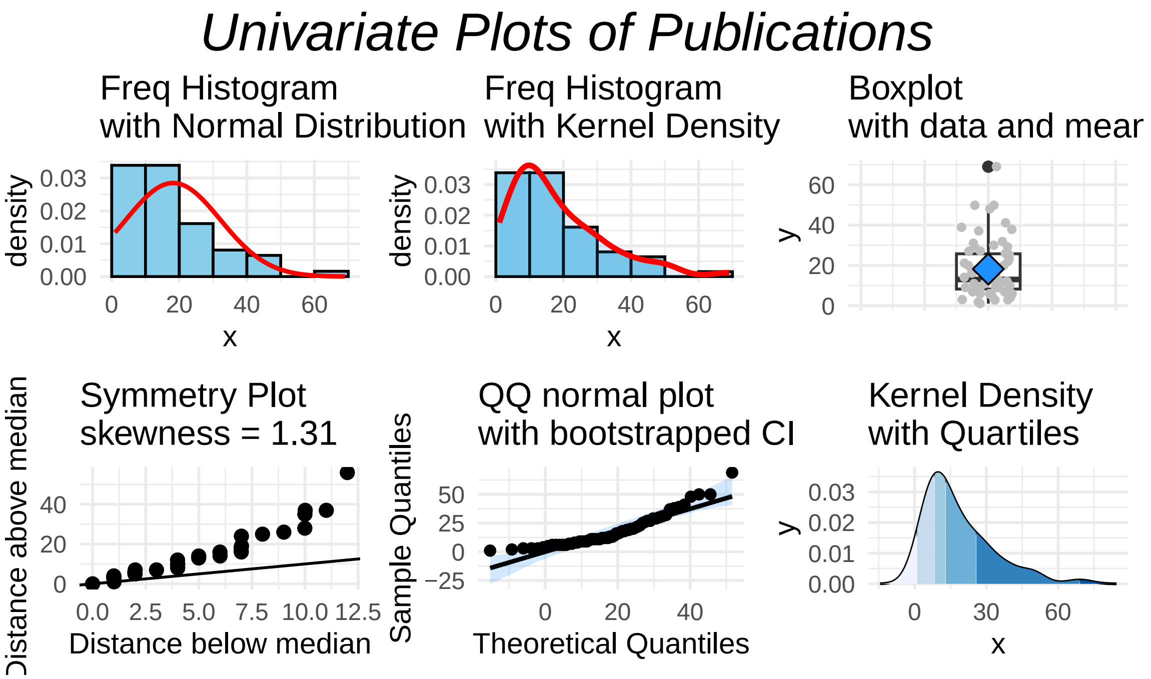
An alternative that is a bit more efficient is to use the 'explore' function in the **bcdstats** package. This produces not only graphs, but also numeric summaries of the variables.

```r
bcdstats::explore(salary, varname="Salary")
```



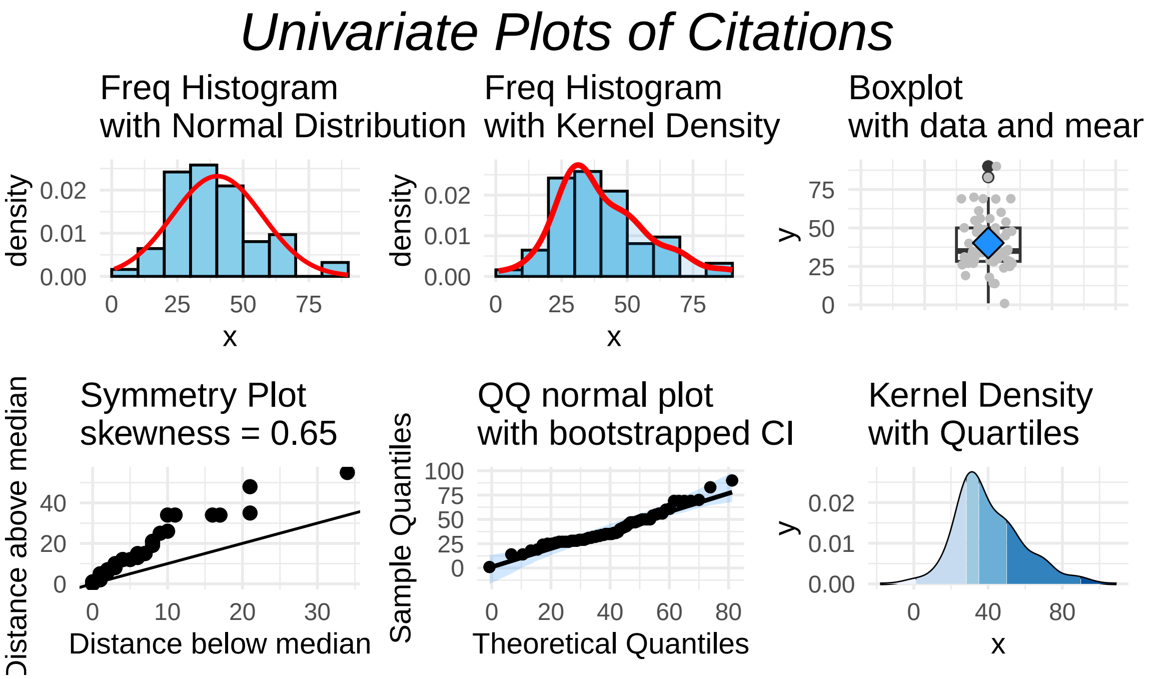
```
   vars  n     mean      sd median trimmed     mad   min   max range skew
X1    1 62 54815.76 9706.02  53681 54226.1 9119.47 37939 83503 45564 0.64
   kurtosis      se
X1      0.5 1232.67
```

```r
bcdstats::explore(pubs, varname="Publications")
```



```
   vars  n  mean sd median trimmed   mad min max range skew kurtosis   se
X1    1 62 18.18 14     13   16.28 10.38   1  69    68 1.38     2.01 1.78
```

```
bcdstats::explore(cits, varname="Citations")
```



```
   vars  n  mean    sd median trimmed   mad min max range skew kurtosis   se
X1    1 62 40.23 17.17     35   39.08 14.08   1  90    89 0.68     0.57 2.18
```

If you don't want to use the 'explore' function from 'bcdstats', then you can still easily obtain numerical summaries with the 'describe' function from the 'psych' package as we have done in prior work, and perhaps just go with the earlier graphs instead of the six panel figure from explore.

```
# our three variables are the third through fifth in the data frame
psych::describe(cohen1[,c("salary","pubs","cits")], skew=T, type=2)
```

```
       vars  n     mean      sd median  trimmed     mad   min   max range skew
salary    1 62 54815.76 9706.02  53681 54226.10 9119.47 37939 83503 45564 0.64
pubs      2 62    18.18   14.00     13    16.28   10.38     1    69    68 1.38
cits      3 62    40.23   17.17     35    39.08   14.08     1    90    89 0.68
       kurtosis      se
salary     0.50 1232.67
pubs       2.01    1.78
cits       0.57    2.18
```

2.2.2 Summary of Univariate EDA

All three variables have some degree of positive skewness, pubs more than cits and salary. This may mean that the residual normality assumption for the linear models is not satisfied. We will look carefully at that assumption following those analyses.

2.3 Bivariate Relationships among the three variables

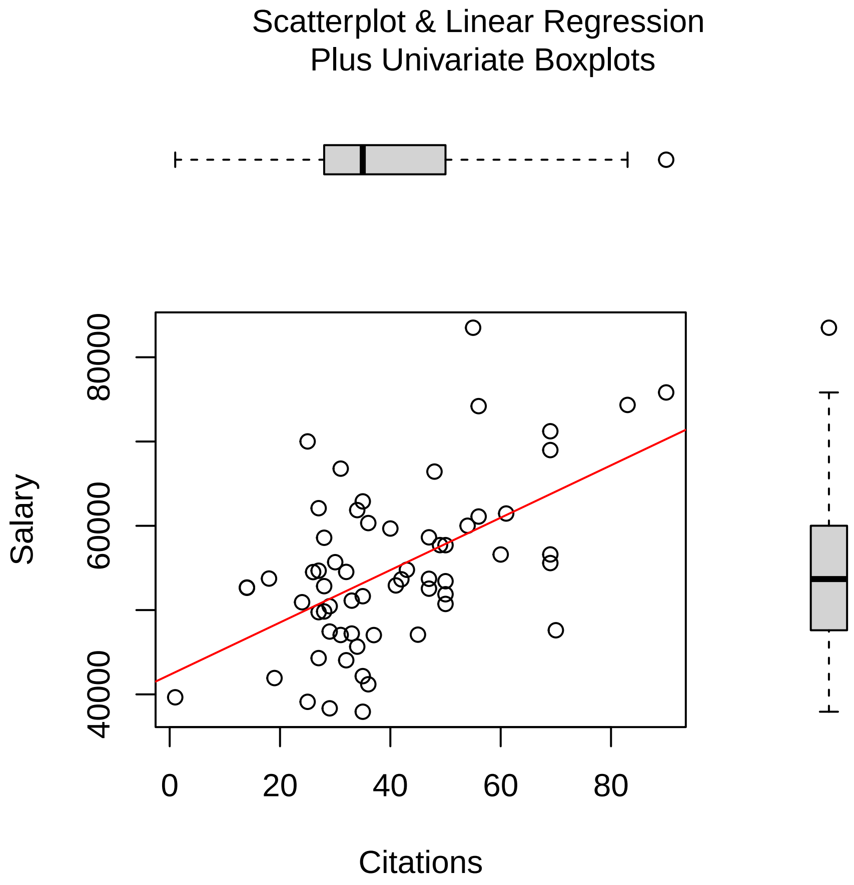
The most important starting point for evaluation of a linear models regression system is visualization of the zero-order bivariate relationships. We do this with scatter plots. The two sections here do this either with base system graphics or ggplot2 in a more sophisticated and perhaps more publication ready format.

Readers are reminded that we earlier examined numerous ways of drawing bivariate scatter plots with R. This was done in the R implementation section when we considered simple regression. It included a base system graphics approach to what I thought might be a publication quality figure. In the present document, I present simple approaches to obtaining maximum information in an exploratory data analysis (EDA) type of approach where speed is important and the investigator is at initial stages of analysis.

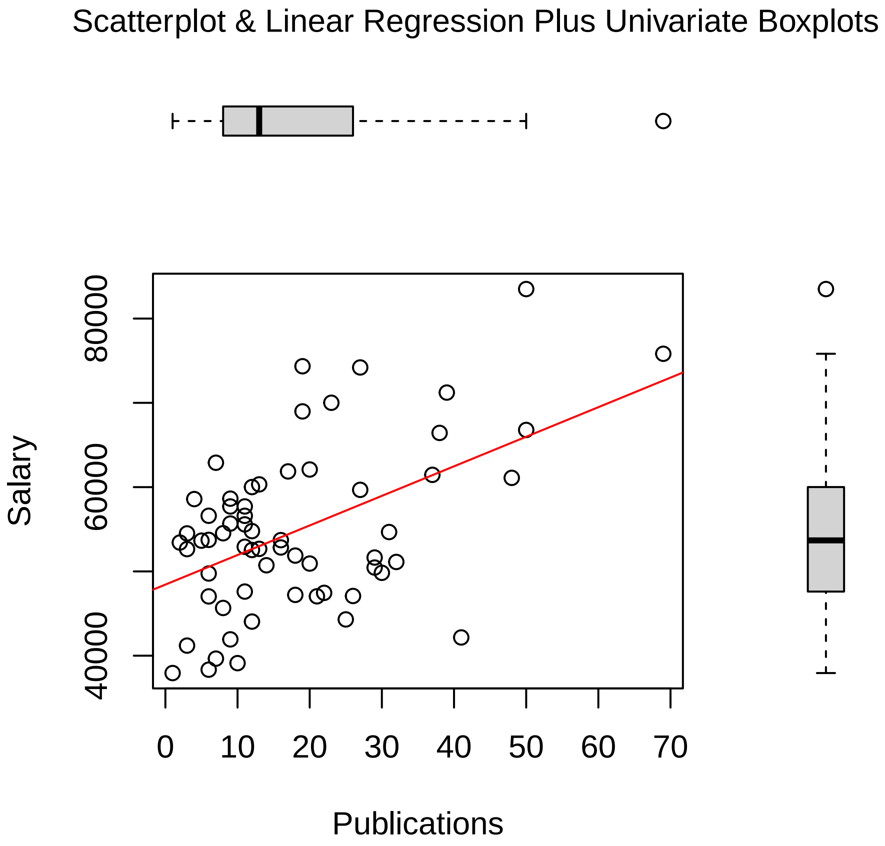
2.3.1 Bivariate Scatterplots and SPLOM with base system functions

Initially, we can draw bivariate scatter plots for all three pairs of variables. Using base system graphics, I have done this in a more complicated fashion by adding boxplots to the margins for each variable

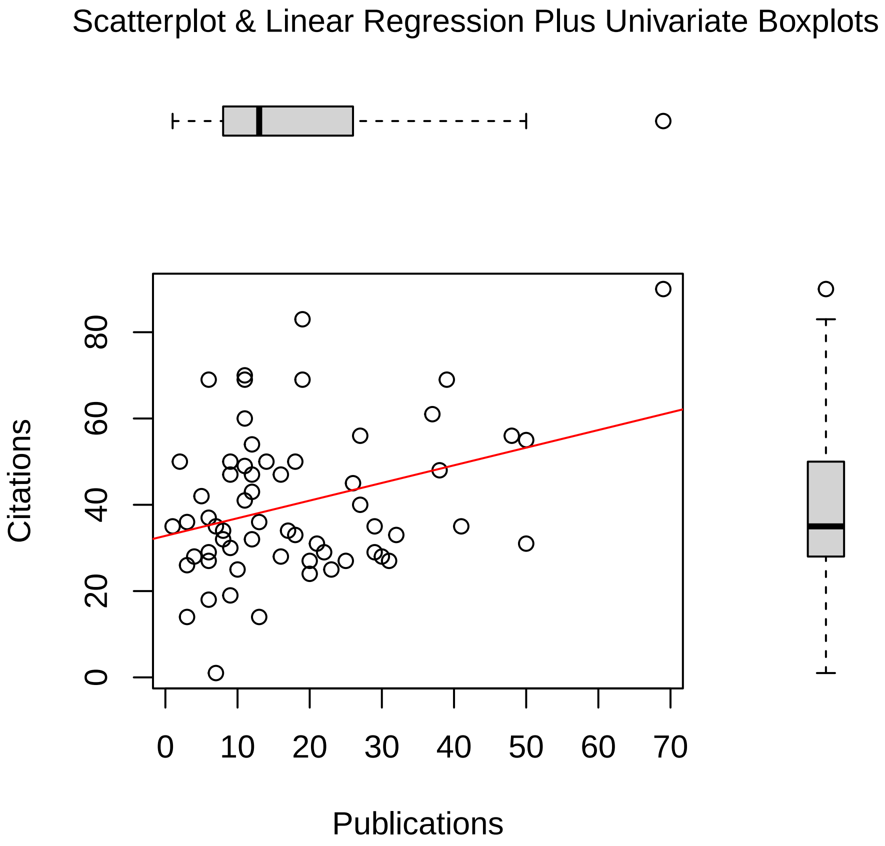
```
par(fig=c(0,0.8,0,0.8))
plot(cits, salary,  xlab="Citations", ylab="Salary")
abline(lm(salary~cits), col="red") # regression line (y~x)
par(fig=c(0,0.8,0.55,1), new=TRUE)
boxplot(cits, horizontal=TRUE, axes=FALSE)
par(fig=c(0.65,1,0,0.8),new=TRUE)
boxplot(salary,axes=FALSE)
mtext("Scatterplot & Linear Regression\n Plus Univariate Boxplots", side=3, outer=TRUE, line=
```



```
par(fig=c(0,0.8,0,0.8))
plot(pubs, salary,  xlab="Publications",
  ylab="Salary")
abline(lm(salary~pubs), col="red") # regression line (y~x)
par(fig=c(0,0.8,0.55,1), new=TRUE)
boxplot(pubs, horizontal=TRUE, axes=FALSE)
par(fig=c(0.65,1,0,0.8),new=TRUE)
boxplot(salary,axes=FALSE)
mtext("Scatterplot & Linear Regression Plus Univariate Boxplots", side=3, outer=TRUE, line=-
```

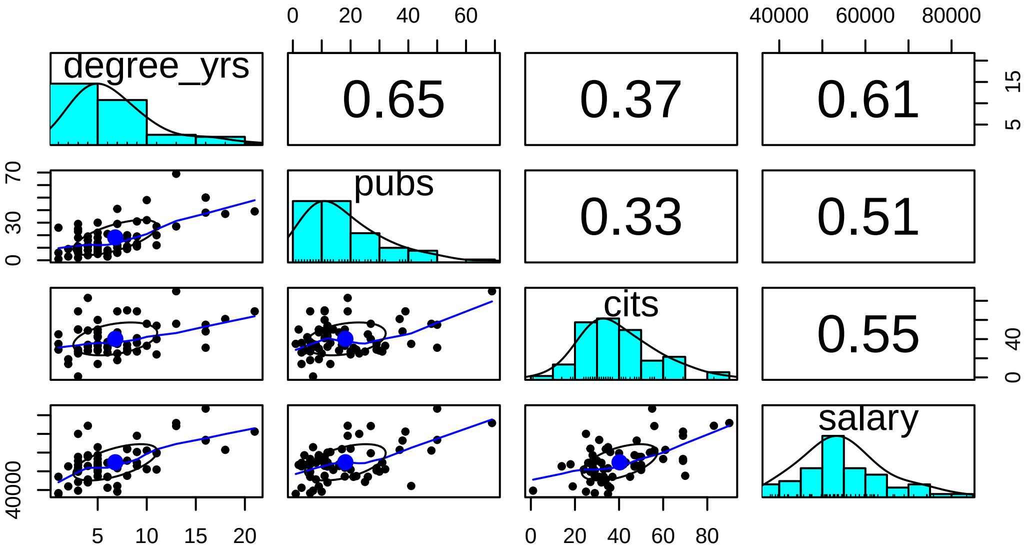


```
par(fig=c(0,0.8,0,0.8))
plot(pubs, cits,  xlab="Publications",
  ylab="Citations")
abline(lm(cits~pubs), col="red") # regression line (y~x)
par(fig=c(0,0.8,0.55,1), new=TRUE)
boxplot(pubs, horizontal=TRUE, axes=FALSE)
par(fig=c(0.65,1,0,0.8),new=TRUE)
boxplot(cits,axes=FALSE)
mtext("Scatterplot & Linear Regression Plus Univariate Boxplots", side=3, outer=TRUE, line=-
```



A preferred approach for the generation of bivariate scatter plots is creation of a scatter plot matrix (SPLOM). There are many ways of doing this in R. Here I use the `pairs.panels` function. I also permitted the 'pairs.panels' function from the **psych** package to include the time variable so that a sense of how it handles larger numbers of variables. Note the "subsetting" syntax for asking to leave out the first and sixth variables from the data frame (case number and gender)

```
psych::pairs.panels(cohen1[c(-1,-6)])
```

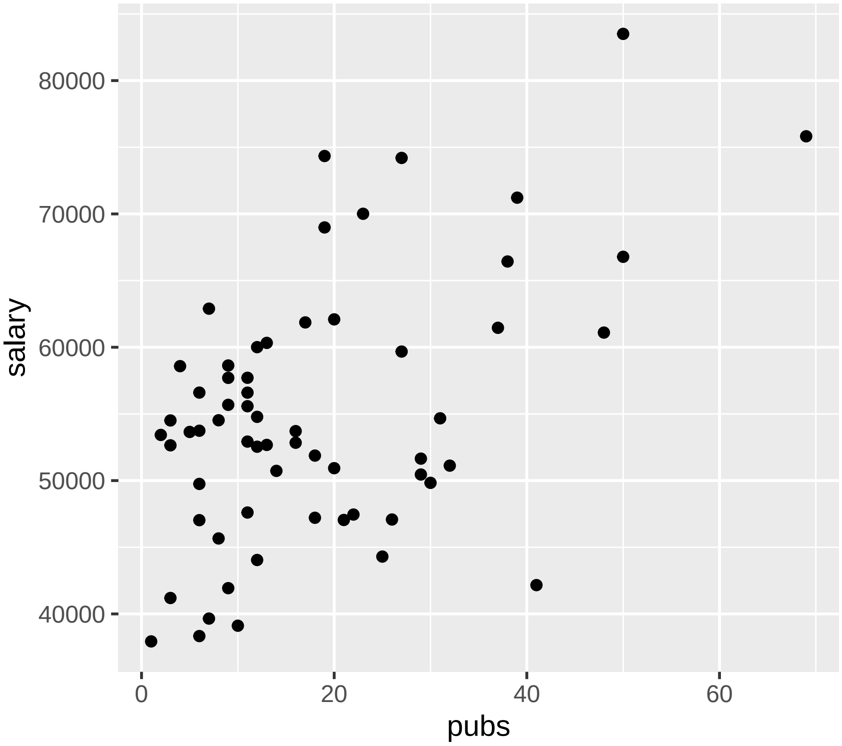


The `pairs.panels` function creates a SPLOM with quite a few attributes beyond the simple bivariate scatter plots. The frequency histograms and kernel density functions as well as the display of the Pearson product-moment zero-order correlation is obvious. However, the added elements on the scatter plots require explanation. Rather than plotting a linear regression function, by default, `pairs.panels` draws a loess fit line. The ellipse that is drawn is based on the 'ellipse' function from John Fox's 'car' package and is implemented by default in pairs.panels. It is called a correlation ellipse and gives a sense of the correlation at limits of one std deviation above and below the mean for each variable. Each of these latter two graph elements can be controlled with arguments passed to 'pairs.panels'. For example, instead of a loess fit, a linear regression line can be chosen. See the help for `pairs.panels` for more information (`?pairs.panels`)

2.3.2 Bivariate Scatterplots and SPLOM with 'ggplot2' and 'lattice'

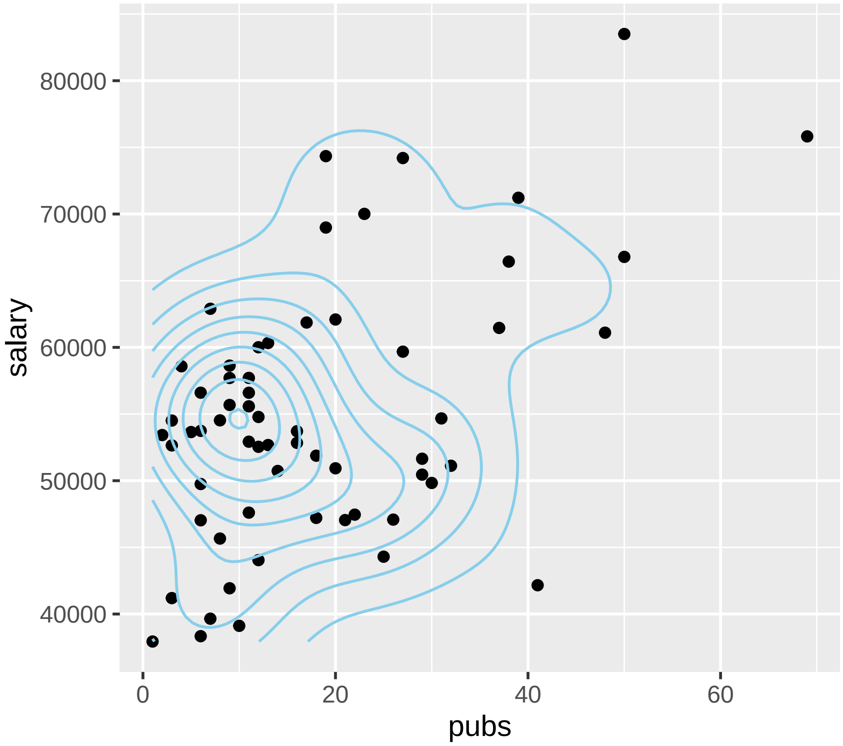
As a reminder of basic work we did with the introduction to using R for simple regression, bivariate scatter plots are drawn here with the `ggplot` function from the **ggplot22* package. The initial two plots are not refined to a publication quality level, and I only take space for the salary-pubs pair of variables. The reader can extend beyond this basic set with only salary and pubs by changing the variables These first two plots should be seen as EDA exploration.

```
# the base plot:
#create base graph entity without displaying anything
pbase <- ggplot(cohen1, aes(x=pubs,y=salary))
#display the scatterplot with the data points
pbase + geom_point()
```



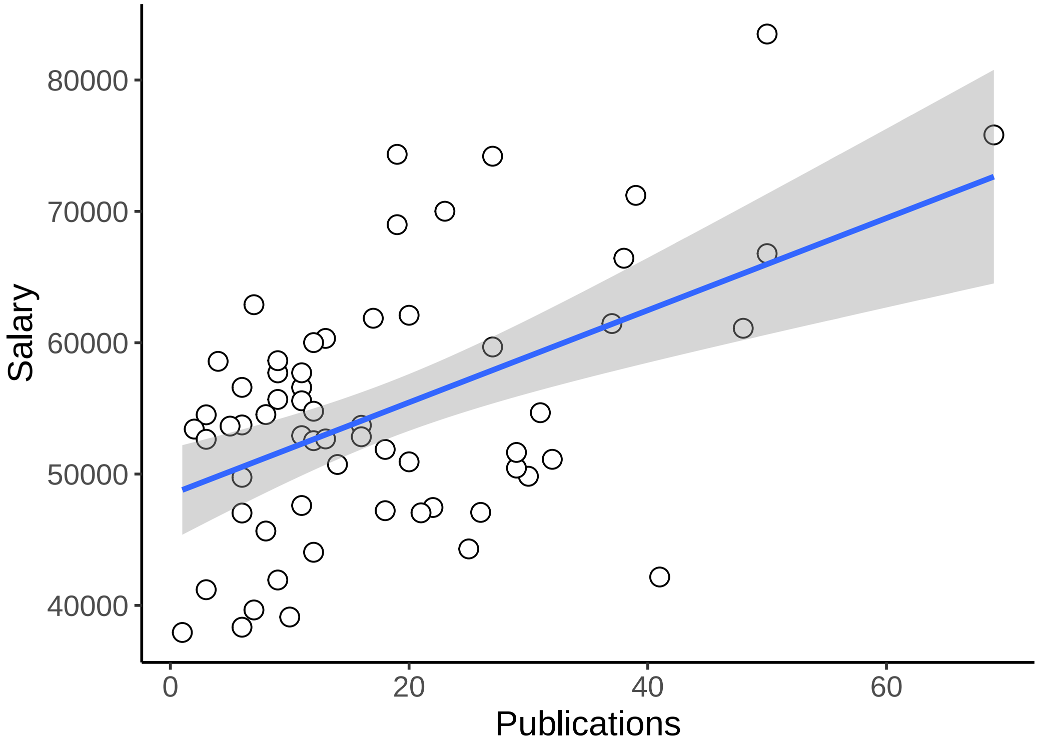
We can add 2D density estimates with contour lines.

```
# add a 2D density using contour lines
pbase + geom_point() + stat_density2d(col="skyblue")
```



A near publication quality figure can be produced with a bit more customization using **ggplot2** tools

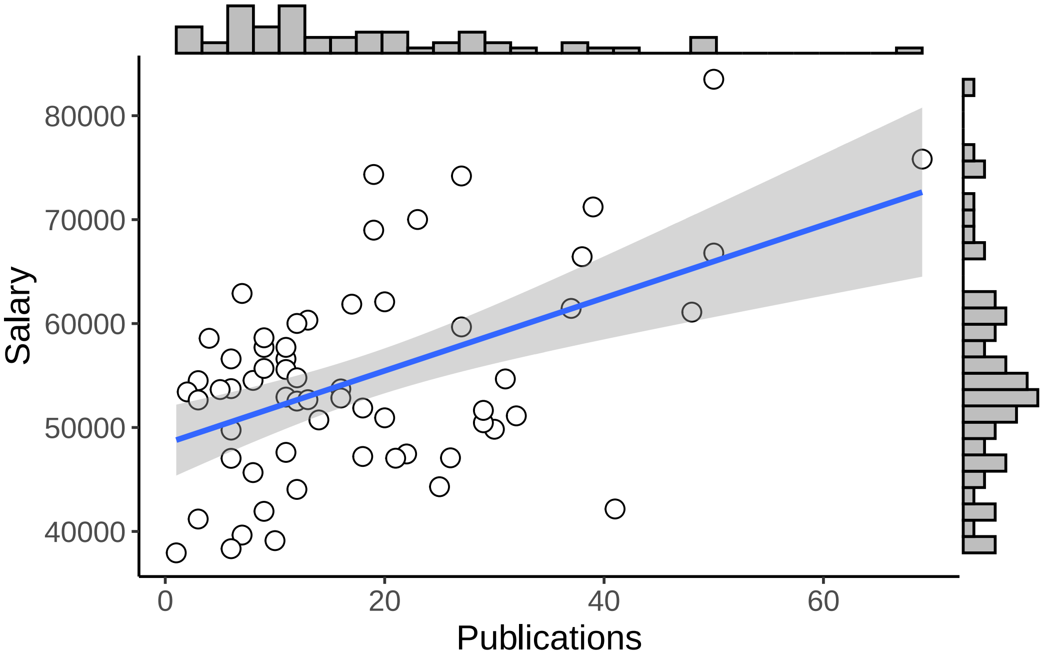
```
#create base graph entity without displaying anything
pbase2 <- ggplot(cohen1, aes(x=pubs,y=salary))
#display the scatterplot with the data points
pbase3 <- pbase2 + geom_point(size=3, shape=21, fill="white", color="black") +
    geom_smooth(method=lm, se=TRUE,fullrange=F) +
    # or:  geom_smooth(method=smooth, se=TRUE,fullrange=F) +
    xlab("Publications") +
    ylab("Salary") +
    theme_classic() +
    theme(axis.text=element_text(size=11),
          axis.title=element_text(size=13))
pbase3
```



The **ggExtra** package provides a facility for producing bivariate scatterplots with frequency histograms, boxplots, or kernel density functions for the univariate distributions in the margins of the scatterplot.

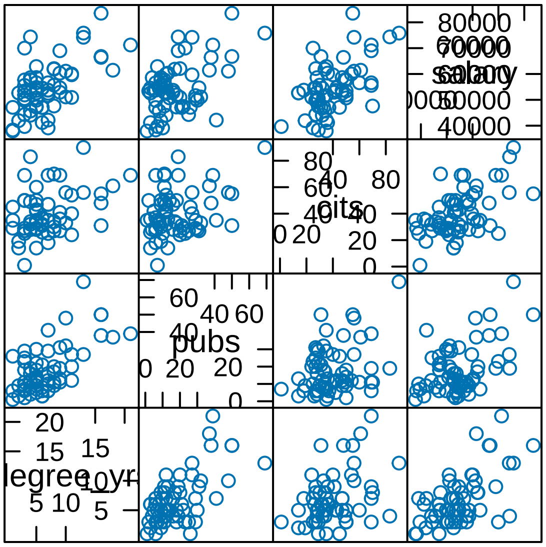
```
#library(ggExtra)
# using base plot pbase3 from above
ggMarginal(pbase3, type="histogram", size=10, fill="gray")
```

```
`geom_smooth()` using formula = 'y ~ x'
`geom_smooth()` using formula = 'y ~ x'
```



Two more ways to obtain a SPLOM are illustrated below. First, we can use the `splom` function from the **lattice** package. Note the exact specification required for requesting the subset of variables from the cohen data frame, and the tilda model symbol used to specify that data frame, and the subsetting to request the same four variables as above.

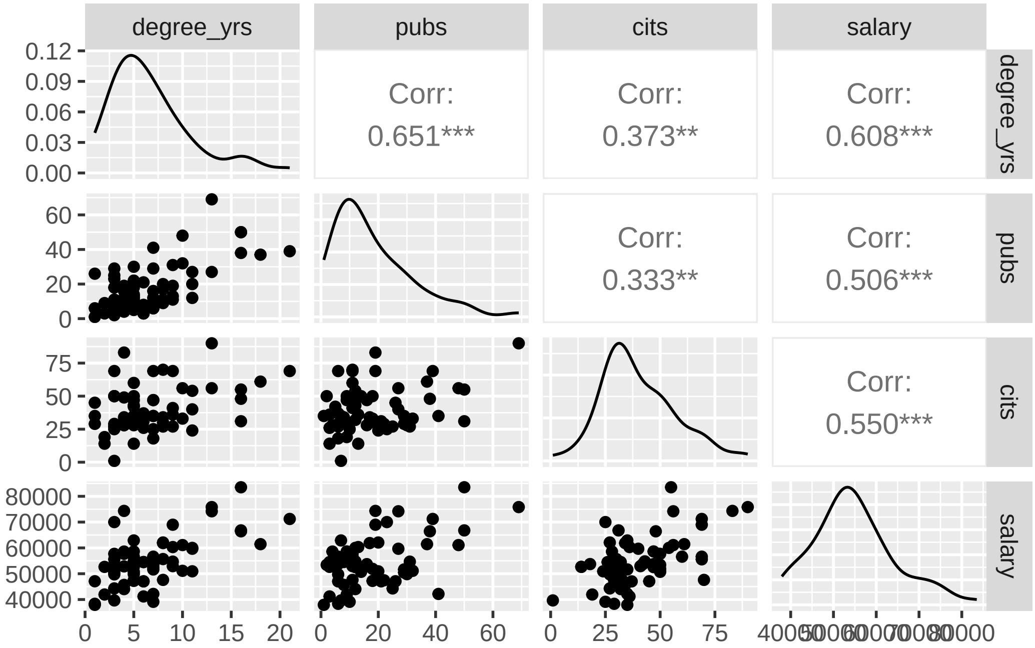
```
lattice::splom(~cohen1[2:5])
```



Scatter Plot Matrix

I have never found that `splom` function to yield a useful or appealing figure. An alternative, and easily implemented, function in the **GGally** package is called `ggpairs`. It is based on ggplot2 and is both efficient and helpful (but I probably still prefer 'pairs.panels')

```
GGally::ggpairs(cohen1[2:5], progress=F)
```



Each of these SPLOMS can be modified/extended to provide more useful information and to tailor the appearance.

2.4 Summary of EDA

The EDA approaches outlined above are views as an essential starting point to any more advanced modeling and inference. “Get to Know Your Data!” It is particularly important to be aware of univariate or bivariate outliers and non-normal distributions of variables. These issues can have major impacts on the ability of the linear modeling results to be accurate, generalizable, and useful. Our three primary variables (salary, pubs and cits) all appeared to have some degree of positive skewness, with pubs having the most. Pubs has an outlier that contributes to this skewness. It will be interesting to examine the residuals-based assumptions for our linear modeling when we know that non-normality in a univariate sense is present. The bivariate EDA approaches did not give any strong sense of multi-dimensional outliers, so this data set set may not have any particularly influential individual cases.

3 Bivariate Correlation and Linear Regression

These bivariate analyses are done largely to reiterate the approaches we established earlier for two variable systems (i.e., simple regression and bivariate correlation). The R implementations are identical to what was covered earlier, except that with more variables, there are more pairs and more than one simple regression possible.

This review establishes some important information to compare to the approaches we took in SPSS, and allows us to remember the meaning of a simple regression regression coefficient and compare its value to coefficients found for the same variables in the multiple regression section of this document that follows this bivariate section.

The sequence of our approach here is to:

a. obtain the variance-covariance matrix for our three variables.
b. obtain the zero-order bivariate correlation matrix (Pearson product-moment correlations).
c. and to test the null hypotheses that the three population correlations are zero.
d. obtain the two different simple regressions of salary on each of the two IVs.
e. graphically evaluate the assumptions of inferences in simple regression.

3.1 Bivariate computations: Covariances and Pearson product-moment correlations

Analyses requested here repeat/extend the implementation approach we learned with R for simple regression.

First, obtain variance-covariance and correlation matrices for the three variables of interest. Recall that we can submit a whole data frame or matrix of data to `cov` and `cor`. That is done here along with subsetting the data frame to use only the three variables for our defined analyses.

```
cat(paste("cov produces a matrix with variances on the leading diagonal
and covariances in the upper and lower triangles.

"))
```

```
cov produces a matrix with variances on the leading diagonal
and covariances in the upper and lower triangles.
```

```r
cov(cohen1[,3:5])
```

```
             pubs       cits      salary
pubs     196.1155    80.1724    68797.68
cits      80.1724   294.8662    91628.81
salary 68797.6830 91628.8096 94206882.35
```

```r
corr1 <- cor(cohen1[,3:5], use="complete.obs", method="pearson")
cat(paste("
cor produces a symmetrical matrix of Pearson correlation coefficients
Here, they are rounded to two places.

"))
```

```
cor produces a symmetrical matrix of Pearson correlation coefficients
Here, they are rounded to two places.
```

```r
round(corr1,2)
```

```
       pubs cits salary
pubs   1.00 0.33   0.51
cits   0.33 1.00   0.55
salary 0.51 0.55   1.00
```

Now we will test each of the three zero-order correlations with the standard t-test and n-2 df:

```r
cor.test(pubs,salary, method = "pearson", alternative = "two.sided",
         exact = FALSE)
```

```

    Pearson's product-moment correlation

data:  pubs and salary
t = 4.5459, df = 60, p-value = 2.706e-05
alternative hypothesis: true correlation is not equal to 0
```

```
95 percent confidence interval:
 0.2934802 0.6710778
sample estimates:
      cor
0.5061468
```

```r
cor.test(pubs,cits, method = "pearson", alternative = "two.sided",
         exact = FALSE)
```

```

    Pearson's product-moment correlation

data:  pubs and cits
t = 2.7392, df = 60, p-value = 0.008098
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.09122073 0.53833356
sample estimates:
      cor
0.3333929
```

```r
cor.test(cits,salary, method = "pearson", alternative = "two.sided",
         exact = FALSE)
```

```

    Pearson's product-moment correlation

data:  cits and salary
t = 5.098, df = 60, p-value = 3.69e-06
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.3477491 0.7030024
sample estimates:
      cor
0.5497664
```

3.2 Test correlations with the `corr.test` function

We need a more efficient way of testing correlations from a whole correlation matrix is to use the `corr.test` and `corr.p` functions from the **psych** package. This method also provides the ability to adjust p values for multiple testing (using the Holm method), and to produce confidence intervals. The down side of using this function is that p values are rounded in appropriately, sometimes to zero, and without exponential notation. An alternative is the `rcorr` function from the **Hmisc** package, but it too has this p value rounding issue.

The commented `adjust.corr` function line from **bcdstats** will produce more interpretable output, especially with p values not rounded to zero.

```
ct1 <- psych::corr.test(cohen1[,3:5])
print(psych::corr.p(ct1$r, n=62), short=FALSE)
```

```
Call:psych::corr.p(r = ct1$r, n = 62)
Correlation matrix
       pubs cits salary
pubs   1.00 0.33   0.51
cits   0.33 1.00   0.55
salary 0.51 0.55   1.00
Sample Size
[1] 62
Probability values (Entries above the diagonal are adjusted for multiple tests.)
       pubs cits salary
pubs   0.00 0.01      0
cits   0.01 0.00      0
salary 0.00 0.00      0

 Confidence intervals based upon normal theory.  To get bootstrapped values, try cor.ci
            lower    r upper    p
pubs-cits    0.09 0.33  0.54 0.01
pubs-salry   0.29 0.51  0.67 0.00
cits-salry   0.35 0.55  0.70 0.00
```

```
#bcdstats::adjust.corr(cohen1[,3:5])
```

3.3 Simple regressions of Salary on each of the two IVs

Next, use the ‘lm’ function from R’s base installation to do linear modeling. Initially, lets do the two simple regressions of salary (the DV) on each of the IV’s separately. We do this for comparability to how we approached the multiple regression problem in conceptual development, but mostly to have the regression coefficients in hand in order to compare them to the regression coefficients for the IVs when included in the two-IV model.

First, we detach the cohen1 data frame so that we can use the “data” argument in the `lm` function, a better practice.

```
detach(cohen1)
```

```
#produce the two single-IV models
fit1 <- lm(salary~pubs, data=cohen1)
fit2 <- lm(salary~cits, data=cohen1)
# obtain relevant inferences and CI's for fit1 with pubs as IV
summary(fit1)
```

```

Call:
lm(formula = salary ~ pubs, data = cohen1)

Residuals:
     Min       1Q   Median       3Q      Max 
-20660.0  -7397.5    333.7   5313.9  19238.7 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 48439.09    1765.42  27.438  < 2e-16 ***
pubs          350.80      77.17   4.546 2.71e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8440 on 60 degrees of freedom
Multiple R-squared:  0.2562,	Adjusted R-squared:  0.2438 
F-statistic: 20.67 on 1 and 60 DF,  p-value: 2.706e-05
```

```
confint(fit1, level=0.95) # CIs for model parameters
```

```
                2.5 %     97.5 %
(Intercept) 44907.729 51970.4450
pubs          196.441   505.1625
```

```
anova(fit1)
```

```
Analysis of Variance Table
```

```
Response: salary
          Df     Sum Sq    Mean Sq F value    Pr(>F)    
pubs       1 1472195326 1472195326  20.665 2.706e-05 ***
Residuals 60 4274424497   71240408                      
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
Anova(fit1,type="III") # note this is Anova, not anova
```

```
Anova Table (Type III tests)

Response: salary
                Sum Sq Df F value    Pr(>F)    
(Intercept) 5.3632e+10  1 752.831 < 2.2e-16 ***
pubs        1.4722e+09  1  20.665 2.706e-05 ***
Residuals   4.2744e+09 60                      
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# obtain relevant inferences and CI's for fit2 with cits as IV

summary(fit2)
```

```

Call:
lm(formula = salary ~ cits, data = cohen1)

Residuals:
     Min       1Q   Median       3Q      Max 
-16462.0  -5822.4   -884.1   5461.7  24096.2 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 42315.71    2662.69  15.892  < 2e-16 ***
cits          310.75      60.95   5.098 3.69e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8175 on 60 degrees of freedom
Multiple R-squared:  0.3022,	Adjusted R-squared:  0.2906 
F-statistic: 25.99 on 1 and 60 DF,  p-value: 3.69e-06
```

```
confint(fit2, level=0.95) # CIs for model parameters
```

```
                 2.5 %     97.5 %
(Intercept) 36989.5395 47641.8738
cits          188.8201   432.6741
```

```
anova(fit2)
```

```
Analysis of Variance Table

Response: salary
          Df     Sum Sq    Mean Sq F value   Pr(>F)    
cits       1 1736876419 1736876419   25.99 3.69e-06 ***
Residuals 60 4009743405   66829057                     
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Anova(fit2,type="III")# note this is Anova, not anova
```

```
Anova Table (Type III tests)

Response: salary
                Sum Sq Df F value    Pr(>F)    
(Intercept) 1.6878e+10  1  252.56 < 2.2e-16 ***
cits        1.7369e+09  1   25.99  3.69e-06 ***
Residuals   4.0097e+09 60                      
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Summary of findings from bivariate analyses/inferences

1. Notice in the above analyses that each regression coefficient has a t value equivalent to what was found for the analogous bivariate correlation test. Not a coincidence - make sure you still recall why this is the case.
2. In addition, the R-squareds reported by the 'summary' function are the square of the zero-order correlations and the F tests are the squares of the t values for testing the regression coefficients.
3. The ANOVA tables produced by `anova` and `Anova` do not give SS total. This is simply SS for Salary, the DV, and can be found other ways (e.g. `var(salary)*(n-1)`). This approach is used below in the multiple regression section.

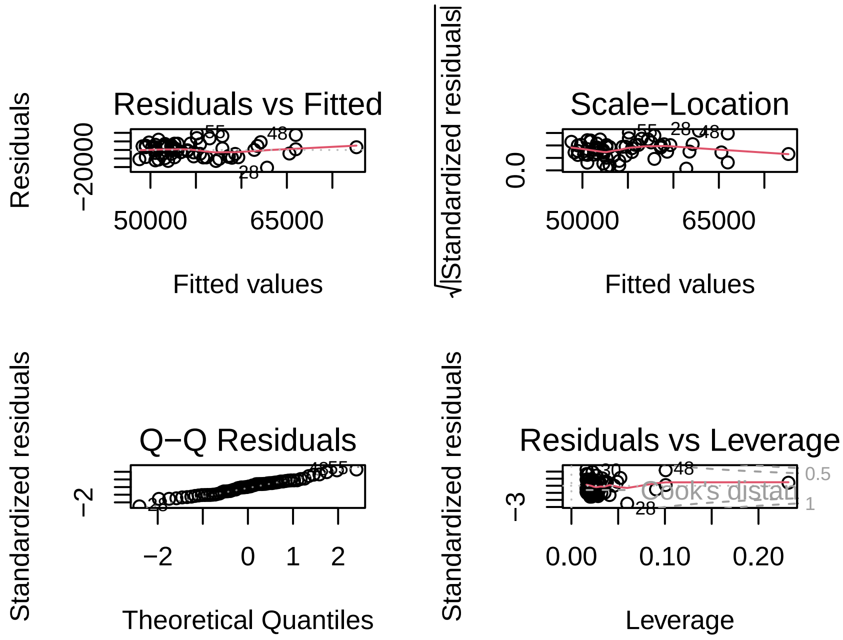
4. The SS regression for each of the fits is labeled as a SS for that IV and not called SS regression as we have done previously. Make sure that understand this equivalence and the labeling choice made in R.
5. Compare the SS for each IV in the `anova` table with the comparable value in the `Anova` table. Ignoring the fact that 'Anova' uses exponential notation, these SS regression values are the same for the two different ANVOA functions. This is the expected outcome in simple regression. But they will not always be equivalent for a particular IV in multiple regression. We explore this below where the differences between `anova` and `Anova` become important in multiple regression.
6. The SS residual value for each IV is the same in the `anova` and `Anova` output for each of the two fits. Again, this is expected in simple regression because the SS regression and SS residual must sum to SS total. This point is emphasized here because it is one area where numerical and conceptual differences occur in multiple regression (see below), compared to these simple regressions.
7. Finally, notice that the F test values obtained by either the `anova` function or the `Anova` function (upper/lower case A) are identical. This is established now for comparison to what happens when we have two IVs below.

3.3.1 Diagnostic plots for simple regressions

Next, we want to obtain the base set of diagnostic plots that R makes available for 'lm' objects. This is the same set that we reviewed in the simple regression section of the course plus an additional normal probability plot of the residuals.

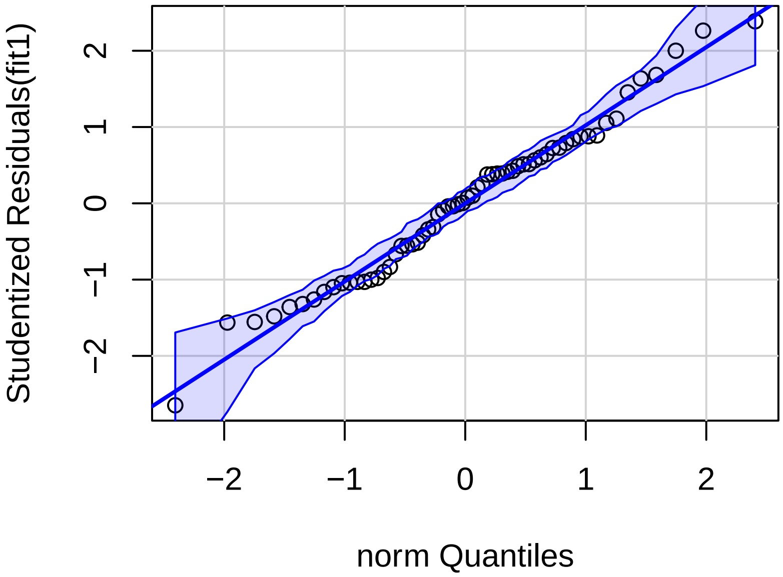
First for fit1:

```r
layout(matrix(c(1,2,3,4),2,2)) # optional 4 graphs/page
plot(fit1)
```



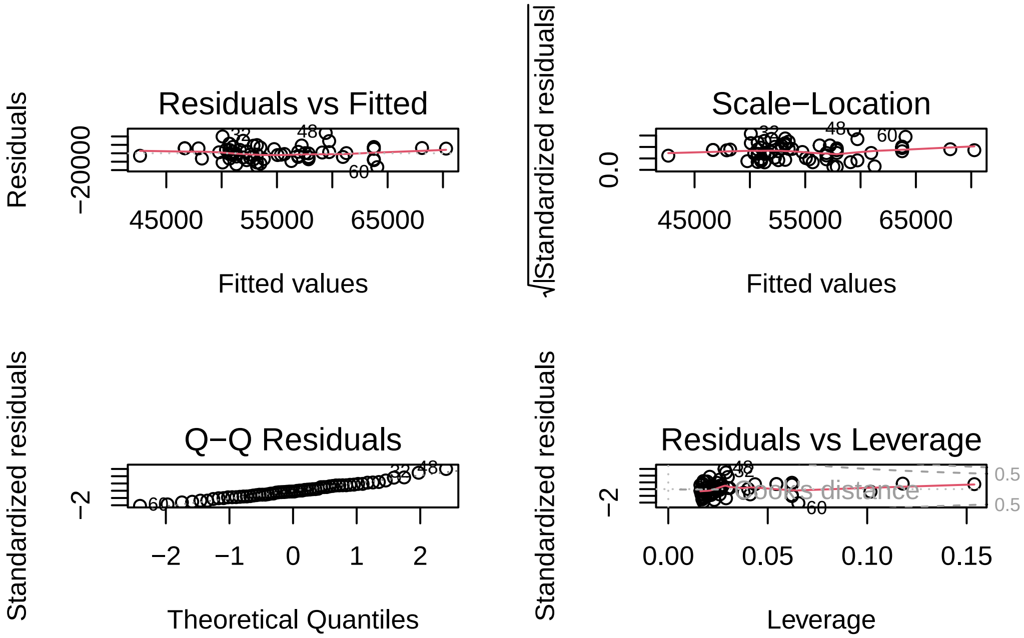
I prefer to draw the qqplot with the `qqPlot` function from the **car** package since it provides confidence bounds. See Fox and Weinberg (2011).

```
car::qqPlot(fit1, distribution="norm", id=F)
```



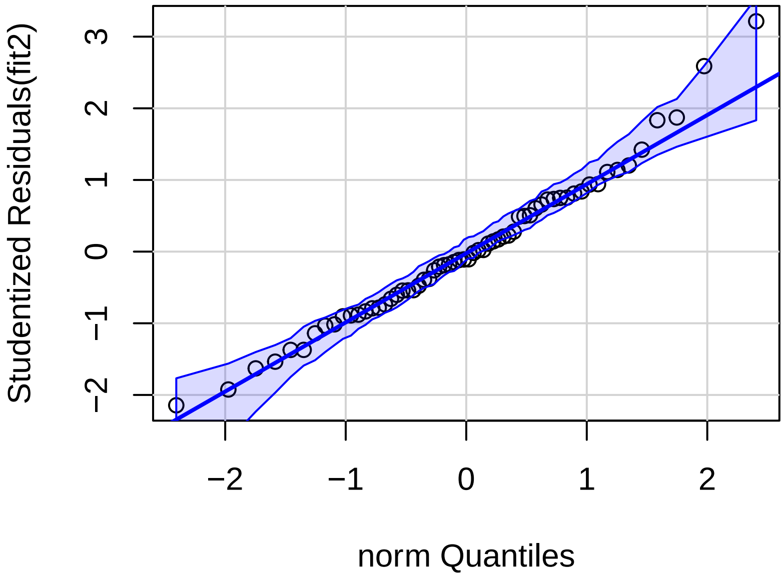
Next, the same plots for fit2:

```
layout(matrix(c(1,2,3,4),2,2)) # optional 4 graphs/page
plot(fit2)
```



And the preferred qqplot for fit2:

```
qqPlot(fit2, distribution="norm", id=FALSE)
```



Since the tests of these two simple regression fits would not typically be used, the diagnostic plots would also not typically be of interest. The analyses seen above are largely here to put in place approaches and numeric values to compare the more interesting multiple regression results found below.

4 Implementation of Standard Multiple Regression Analyses: Linear modeling with multiple IVs

When we covered implementation of simple regression procedures in R, we covered most of the R syntax required for multiple regression. Only a few additional coding strategies are necessary to employ models with more than one IV. We also reviewed those simple regression techniques in the previous section of this document. The illustrations outlined below emphasize a two-IV model for comparability to the example we worked out in SPSS. The major change in analysis from simple regression is the set of topics including diagnostics,model criticism, influence analysis, etc and those comprise later chapters

Some later chapters extend this capability to more than two IVs, but we will emphasize that to a greater degree at a later point in the course and in other documents.

The reader should also note that the following sections have two interrelated goals. One is simply the illustration of coding strategies in R for basic linear modeling. Some sections go beyond that narrow goal and continue to reinforce and expand the conceptual framework we have put in place for multiple regression. Some more advanced R strategies are also employed to obtain useful graphics and output that is formatted well for the markdown approach taken in this document.

The general sequence of the implementation is the following:

a. perform the two-IV multiple regression
b. obtain all the components and tests the we implemented in SPSS for the multiple regression
c. obtain the basic evaluations of residual normality and homoscedasticity
d. obtain additional information such as semi partial correlations, beta weights and CI's
e. reinforce the concepts of unique and common proportions of variance, R squared, SS partitioning, etc
f. perform some basic diagnostics and evaluation of assumptions

4.1 The core linear modeling approach with R

Requesting a multiple regression using `lm` is an easy extension of the basic expressions used for simple regression. The "model" specification still uses the "tilda" symbol to indicate the DV to the left and the IV's to the right. All IV's are named, and the plus symbol is used to essentially list the full set of IV's. Here, I fit two models, each with the same two IVs, but I named them in opposite order. We will explore whether this makes any difference in outcome.

Typically we would NOT fit both models of the different IV orders. It is done here for instructional purposes that play out below, in the next chapter, and later in the semester.

First, establish the two alternative model fits:

```
fit3 <- lm(salary~pubs+cits, data=cohen1)
fit4 <- lm(salary~cits+pubs, data=cohen1)
```

4.1.1 Explore fit3 (`salary~pubs+cits`)

Obtain the same set of additional summaries/analyses/plots we did for simple regression. First for model fit3:

```
summary(fit3)
```

```
Call:
lm(formula = salary ~ pubs + cits, data = cohen1)

Residuals:
     Min       1Q   Median       3Q      Max 
-17133.1  -5218.3   -341.3   5324.1  17670.3 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 40492.97    2505.39  16.162  < 2e-16 ***
pubs          251.75      72.92   3.452  0.00103 ** 
cits          242.30      59.47   4.074  0.00014 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 7519 on 59 degrees of freedom
Multiple R-squared:  0.4195,	Adjusted R-squared:  0.3998 
F-statistic: 21.32 on 2 and 59 DF,  p-value: 1.076e-07
```

```
confint(fit3, level=0.95) # CIs for model parameters
```

```
                 2.5 %     97.5 %
(Intercept) 35479.6889 45506.2540
pubs          105.8395   397.6605
cits          123.3023   361.2931
```

At this point, let's recall the interpretation of the values for the regression coefficients. For comparison's sake, analyses in the previous section found that when, in simple regression, salary was regressed on pubs, the regression coefficient was $350.80. It was interpreted to mean that each publication was related to an increase in salary of $350.80. Now, in the two-IV regression, the regression coefficient is seen to be $251.75. The change is recognized to have come about because the two IV's are correlated and the multiple regression coefficient is said to be a partial regression coefficient. That is, it is the change in the DV for a one unit change in the IV, controlling for the presence of the other IV. So, in fit3, each pub is uniquely associated with about a $251.75 increase in salary. The interpretation of the regression coefficient for cits is similar. The conclusion is that the impact of each IV in a regression model is only interpretable in the context of that model. The presence or absence of other IV's will influence the capability of any IV to predict the DV, when the IVs are correlated.

Next, we produce ANOVA summary tables two ways, using both the `anova` and `Anova` functions. Are they different?

Note that I have formatted the ANOVA tables in a manner that might be unfamiliar, using the 'kable' and 'tidy' functions. This approach permits direct comparison of the output from the two functions since the default listing for the 'Anova' table uses scientific notation for SS values. See the section below on R Markdown formatting capabilities for more detail on this.

The first thing to notice about the ANOVA tables found below is that they do not simply present SS Regression, SS Residual and SS total as you might have expected from how we saw that SPSS produced ANOVA tables. SS total is not shown at all, and SS Regression appears to be broken apart and assigned to the two IVs. Exactly where these SS for pubs and cits come from is a longer discussion. We covered this point earlier in our multiple regression expositions. Reiteration of the importance of this question is begun just below here and in an later section of this document (chapter 5, where we conceptually connect these SS to squared semi-partial correlations).

Make sure you see that the "statistic" column in the kable/tidy version of the ANOVA tables is actually just the F value. Make sure you know how these F's were computed (what divided by what?)

```
# kable function from knitr;  tidy function from broom package
kable(tidy(anova(fit3))) # Type I SS from anova
```

term	df	sumsq	meansq	statistic	p.value
pubs	1	1472195326	1472195326	26.03841	0.0000037
cits	1	938602110	938602110	16.60086	0.0001396
Residuals	59	3335822387	56539362	NA	NA

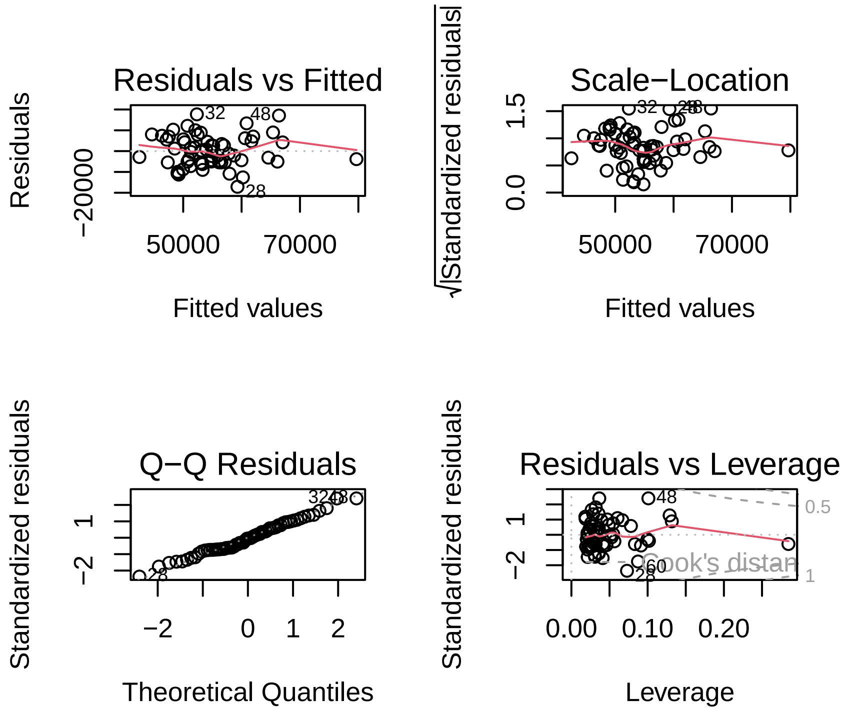
```r
# "unique" ss for each IV produced by the Anova function.
# Can also be found by multiplying
# the squared semi-partial for each IV by SStotal
kable(tidy(Anova(fit3,type="III")))# note this is Anova, not anova
```

term	sumsq	df	statistic	p.value
(Intercept)	14769235188	1	261.22041	0.0000000
pubs	673921018	1	11.91950	0.0010337
cits	938602110	1	16.60086	0.0001396
Residuals	3335822387	59	NA	NA

At this point, in comparing the results from the two anova functions, it is useful to recall that `anova` produces Type I SS and `Anova` was set up to request Type III SS. For the moment, lets just note that the SS and F value for cits is the same in both anova tables. Cits was entered second in the model specification for fit3. This point is expanded below.

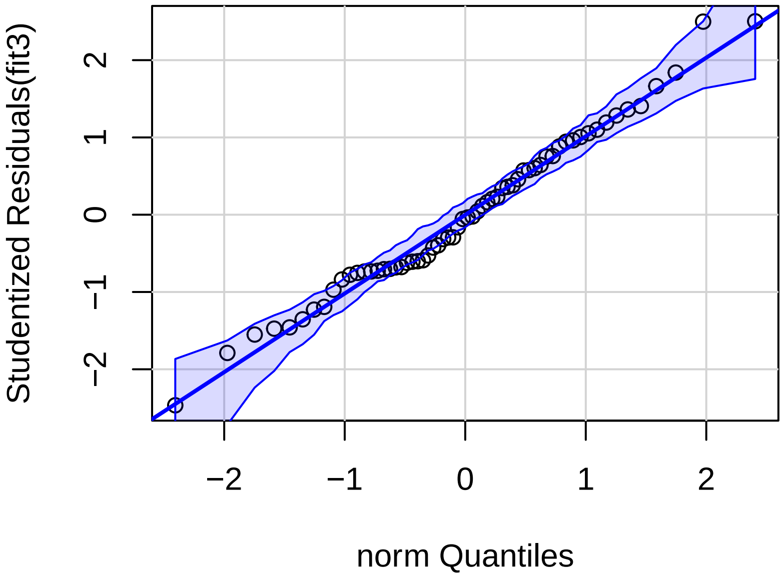
Next, we obtain the standard set of diagnostic plots are obtained as they were for simple regression. Recall that I ask for these plots to be put on one figure by using the 'layout' function. At this point in the course, we have not actually covered two of the four plots, but two are our standard assessments of the normality and homoscedasticity assumptions. There do not appear to be strong violations of either assumption (seen in the two plots on the left)

```r
#influence(fit3)
# regression diagnostics
layout(matrix(c(1,2,3,4),2,2)) # optional 4 graphs/page
plot(fit3)
```



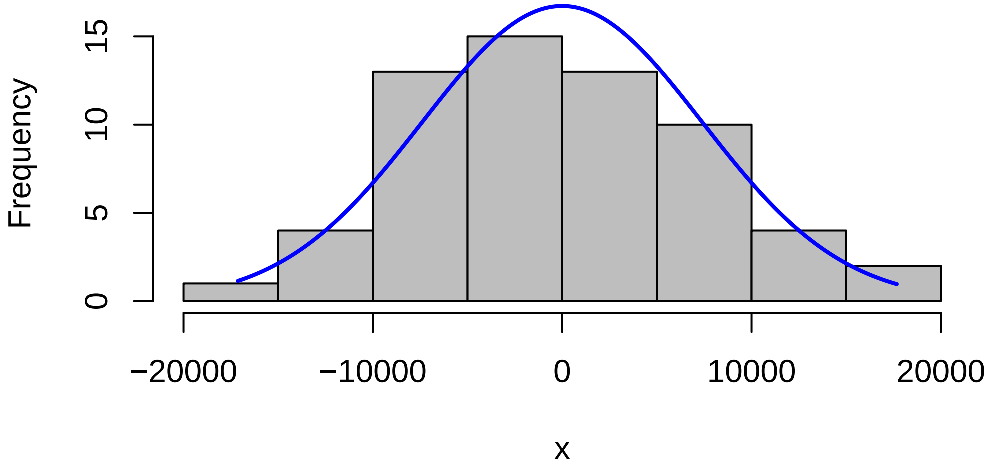
And we can also obtain the preferred qqplot:

```
car::qqPlot(fit3, distribution="norm", id=F)
```



A histogram of the residuals with a normal curve overlaid also provides a helpful visualization.

```
rcompanion::plotNormalHistogram(residuals(fit3))
```



4.1.2 Explore fit4 (salary~cits+pubs)

Next, we examine model fit4 (the one with the opposite order of IVs listed in the model specification):

```r
summary(fit4)
```

```
Call:
lm(formula = salary ~ cits + pubs, data = cohen1)

Residuals:
     Min       1Q   Median       3Q      Max 
-17133.1  -5218.3   -341.3   5324.1  17670.3 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 40492.97    2505.39  16.162  < 2e-16 ***
cits          242.30      59.47   4.074  0.00014 ***
pubs          251.75      72.92   3.452  0.00103 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 7519 on 59 degrees of freedom
Multiple R-squared:  0.4195,	Adjusted R-squared:  0.3998 
F-statistic: 21.32 on 2 and 59 DF,  p-value: 1.076e-07
```

```
confint(fit4, level=0.95) # CIs for model parameters
```

```
                 2.5 %     97.5 %
(Intercept) 35479.6889 45506.2540
cits          123.3023   361.2931
pubs          105.8395   397.6605
```

The ANOVAS for fit4:

```
# kable function from knitr;  tidy function from broom package
kable(tidy(anova(fit4))) # Type I SS produced by anova
```

term	df	sumsq	meansq	statistic	p.value
cits	1	1736876419	1736876419	30.71977	0.0000007
pubs	1	673921018	673921018	11.91950	0.0010337
Residuals	59	3335822387	56539362	NA	NA

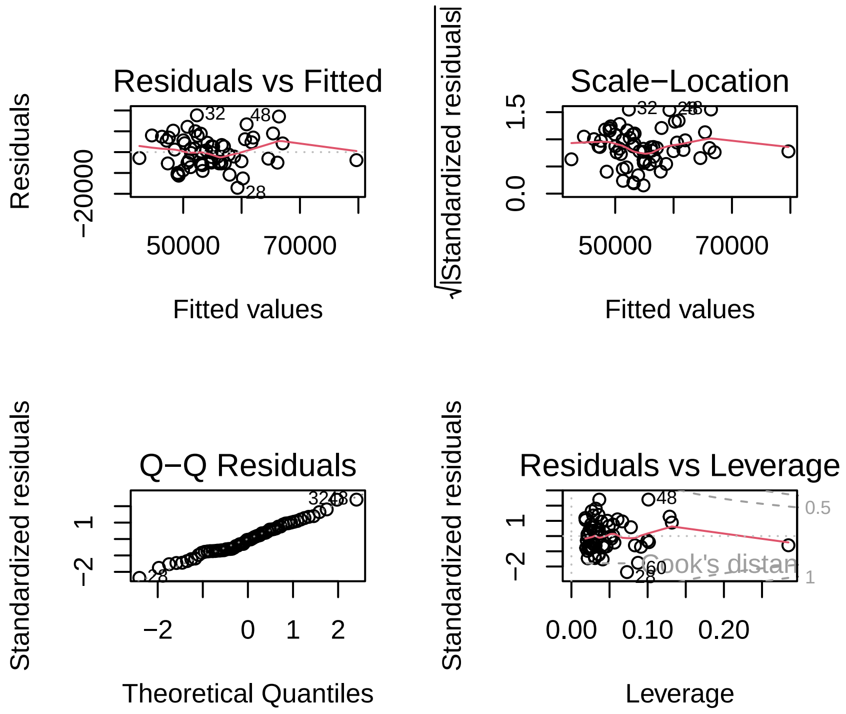
```
# "unique" ss for each IV produced by the Anova function.
# Can also be found by multiplying
# the squared semi-partial for each IV by SStotal
kable(tidy(Anova(fit4,type="III")))# note this is Anova, not anova
```

term	sumsq	df	statistic	p.value
(Intercept)	14769235188	1	261.22041	0.0000000
cits	938602110	1	16.60086	0.0001396
pubs	673921018	1	11.91950	0.0010337
Residuals	3335822387	59	NA	NA

Once again, the SS and F's are identical for pubs, which is the second IV specified in the model specification - is is the "last entered". The SS and F's differ for the first entered variable, cits, as was also the case for fit3. These differences are explored below.

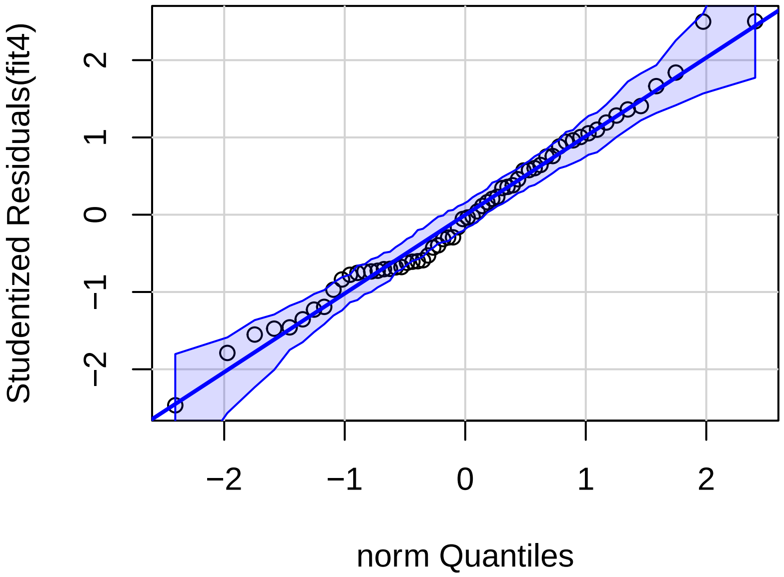
We can also examine diagnostic plots for fit4. They should look identical to those for fit3 since the residuals are the same for each of the two fits.

```r
#influence(fit3)
# regression diagnostics
layout(matrix(c(1,2,3,4),2,2)) # optional 4 graphs/page
plot(fit4)
```



Here is the preferred qqplot for fit4:

```r
car::qqPlot(fit4, distribution="norm", id=F)
```



4.1.3 Comparing fit3 and fit4 results.

We have emphasized how the SS and F-tests for fits 3 and 4 differ, depending on whether we use `anova` or `Anova`. But it is helpful to ask which components are identical for the two Fits. These items/statistics are identical:

1. The Multiple R-squared
2. SS Regression (although not printed above). It is found by adding the SS for the two IVs from the `anova`-produced table.
3. SS Residual
4. df regression/residual
5. The regression coefficients, confidence Intervals, and t-tests of the coefficients are also identical for the two models
6. Below, we also see that beta weights, partial r's, semi-partial r's, tolerances and unique/common variance proportions are identical for the two models

The primary difference emerges when examining the SS partitioning. Lets do one more comparison here. These SS computations will be revisited in a later chapter. If we rerun the fit3 model here it will permit comparing the SS, F's and t-tests of the regression coefficients when using either anova or Anova. From this analysis, make note of the t-values. I saved them out of the model fit to work with in a section below.

```
fit3b <- lm(salary~pubs+cits,data=cohen1)
summary(fit3b)
```

```
Call:
lm(formula = salary ~ pubs + cits, data = cohen1)

Residuals:
     Min       1Q   Median       3Q      Max 
-17133.1  -5218.3   -341.3   5324.1  17670.3 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 40492.97    2505.39  16.162  < 2e-16 ***
pubs          251.75      72.92   3.452  0.00103 ** 
cits          242.30      59.47   4.074  0.00014 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 7519 on 59 degrees of freedom
Multiple R-squared:  0.4195,	Adjusted R-squared:  0.3998 
F-statistic: 21.32 on 2 and 59 DF,  p-value: 1.076e-07
```

```
tvals <- summary(fit3b)$coefficients[2:3,3] # subsetting to extract the t's for the two IVs
tvals
```

```
    pubs     cits 
3.452463 4.074415 
```

Next, re-obtain the SS and F tests from the Anova function, and save the F values out to a vector.

```
Anova(fit3b, type=3)
```

```
Anova Table (Type III tests)

Response: salary
```

```
                Sum Sq Df F value    Pr(>F)    
(Intercept) 1.4769e+10  1 261.220 < 2.2e-16 ***
pubs        6.7392e+08  1  11.919 0.0010337 ** 
cits        9.3860e+08  1  16.601 0.0001396 ***
Residuals   3.3358e+09 59                      
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
fvals <- Anova(fit3b, type=3)[2:3,3] # subsetting to extract F's for the two IVs
fvals
```

```
[1] 11.91950 16.60086
```

Now we can take the square roots of the F values and compare those to the t values. Since each F is based on 1 and n-k-1 df, the square roots will produce a t value with those n-k-1 df.

```
fvals^.5
```

```
[1] 3.452463 4.074415
```

```
tvals
```

```
    pubs     cits 
3.452463 4.074415 
```

Since the quantities are identical, we can conclude that the t-tests of the regression coefficients are equivalent to tests of Type III SS for each IV. Recalling comparisons of fit3 and fit4 above (the two opposite IV entry orders), we also recall that the coefficients and their t-tests are the same. The conclusion is that tests of the regression coefficients for any model are actually tests of type III SS for those IVs.

The type I SS seen from the `anova` analyses lead us to conclude that those SS are found by computing the unique SS for each variable at the point it enters the equation. That is why the SS for each variable differs, depending on whether it is entered first or second. The type III SS are found by treating each variable as if it were the second (last) to enter. Further implications of this are found in a later chapter.

4.2 Efficiently obtaining important descriptive components of a multiple-IV linear model

It has been argued that several core components of multiple regression should be produced whenever any linear model is fit. Some of those components have not yet been requested/produced with the methods outlined above (e.g. beta weights). I have written my own R function that obtains many of these additional items that we have previously discussed. The function is called `mrinfo` (multiple regression information) and I just call it Mr. Info. It is available in the **bcdstats** package.

The syntax is to just pass a `lm` model fit object to the `mrinfo` function. We do so here with our fit3 model and we will examine the output. The unique proportion of variance accounted for by each IV are equivalent to the square of the semi-partial correlation of that IV with the DV, after residualizing that IV on all other IVs. The argument `minimal` permits request of only a set of supplemental information or a longer list of regression information that has been covered above in piecemeal fashion.

```
bcdstats::mrinfo(fit3, minimal=TRUE)
```

```
[[1]]
NULL

$`supplemental information`
     beta wt structure r partial r semipartial r tolerances  unique  common
pubs 0.36323     0.78145   0.40996       0.34245    0.88885 0.11727 0.13891
cits 0.42867     0.84880   0.46860       0.40414    0.88885 0.16333 0.13891
       total
pubs 0.25618
cits 0.30224

$`var infl factors from HH:vif`
   pubs    cits
1.12505 1.12505

[[4]]
NULL
```

We can obtain these statistics for fit4 as well, but all are identical to those from fit3.

```
bcdstats::mrinfo(fit4, minimal=TRUE)
```

```
[[1]]
NULL

$`supplemental information`
     beta wt structure r partial r semipartial r tolerances  unique  common
cits 0.42867     0.84880   0.46860       0.40414    0.88885 0.16333 0.13891
pubs 0.36323     0.78145   0.40996       0.34245    0.88885 0.11727 0.13891
       total
cits 0.30224
pubs 0.25618

$`var infl factors from HH:vif`
   cits    pubs
1.12505 1.12505

[[4]]
NULL
```

The beta weights, partial correlations, semi-partial correlations, and tolerances should be values that the reader understands and expects, given our prior conceptual and SPSS work. But what are the "unique", "common", and "total" values. The reader should examine those, keeping in mind core ways of thinking about the Multiple R squared (which you know from above and from SPSS work). Recall that the multiple R-squared from both fits 3 and 4 was about .42. How might we describe the "unique" quantities, and how do they relate to the semi-partials? (Hint: sum the two unique proportions and add the common proportion)

4.2.1 A caveat on using 'mrinfo'

The original data file from the cohen text called the "years since degree" variable "time". We will examine a 3 IV model using that variable later in this document. At present, the word time can't be used for a variable in a `lm` model that is submitted to the 'mrinfo' function. It thinks it is an R function called 'time'. So I changed it to degree_yrs.

4.2.2 Information Criteria

Information criteria are usually used in comparing models and that is covered in a later chapter and with the use of the `tidy` function. They are not useful as standalone values in characterizing a single model. But it is worthwhile here to demonstrate the ease of obtaining AIC and BIC, the most commonly used of those statistics.

```
AIC(fit3)
```

```
[1] 1287.601
```

```
BIC(fit3)
```

```
[1] 1296.11
```

4.2.3 Another method to obtain partial and semi partial correlations

The **olsrr** package has many very useful functions for characterizing `lm` models. Some of those are covered in a later chapter, including a great many visualizations. But it is worth mentioning here, one additional way to obtain partial and semi-partial correlations of each IV with the DV, controlling for the other IV's. Note the same outcome as that produced by `mrinfo`.

```
olsrr::ols_correlations(fit3)
```

```
               Correlations
------------------------------------------
Variable    Zero Order    Partial    Part
------------------------------------------
pubs             0.506      0.410    0.342
cits             0.550      0.469    0.404
------------------------------------------
```

4.3 Diagnostic and Added-variable plots for the full two-IV model

Many graphical techniques can be explored for evaluation of assumptions, influence, and partial relationships in multivariable systems. A few are explored here, and others are presented in later chapters (e.g., see the chapter on the **olsrr** package).

4.3.1 Working with the residuals and predicted scores (yhats)

Residuals and yhats from any `lm` model can be quickly obtained.

```
fit3.resid <- resid(fit3)
fit3.pred <- predict(fit3)
```

These vectors can also be standardized.

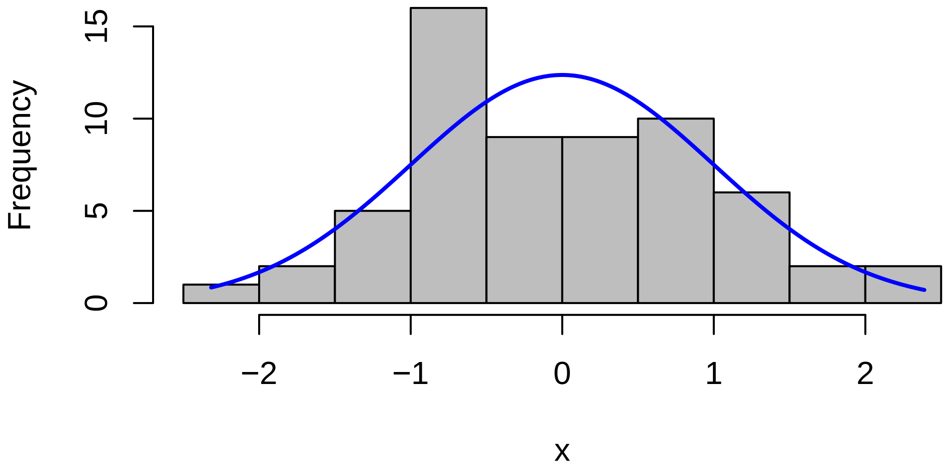
```
fit3.zresid <- scale(fit3.resid)
fit3.zpred <- scale(fit3.pred)
```

We might also place these new vectors into a data frame along with the original variables, in order to save them for other future work.

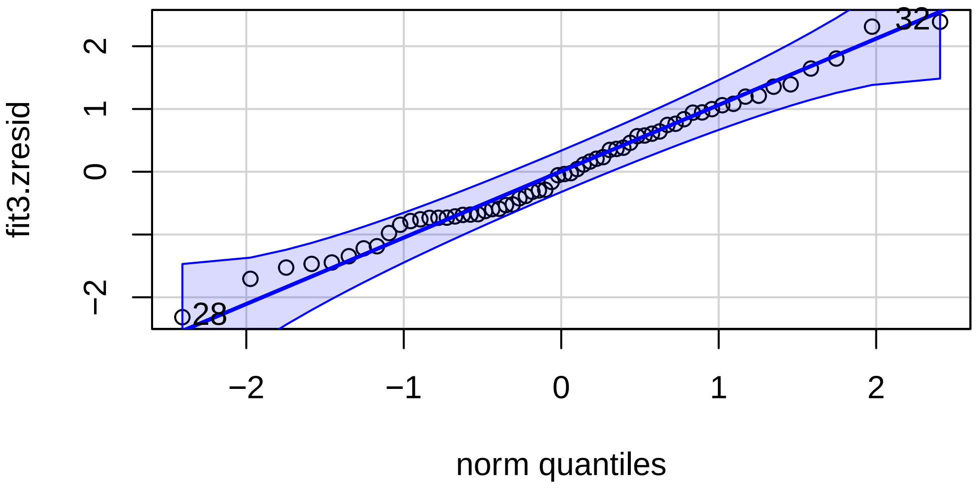
```
# cohen1 was already detached
#detach(cohen1)
cohen1new <- cbind(cohen1, fit3.resid, fit3.pred, fit3.zresid, fit3.zpred)
# and then reattach cohen1 for possible use later
#attach(cohen1new)
```

Now we can work directly with these new residual and yhat vectors. Some of these plots duplicate what was shown above, but it is useful to have the standardized residuals and yhats available.

```
rcompanion::plotNormalHistogram(fit3.zresid)
```

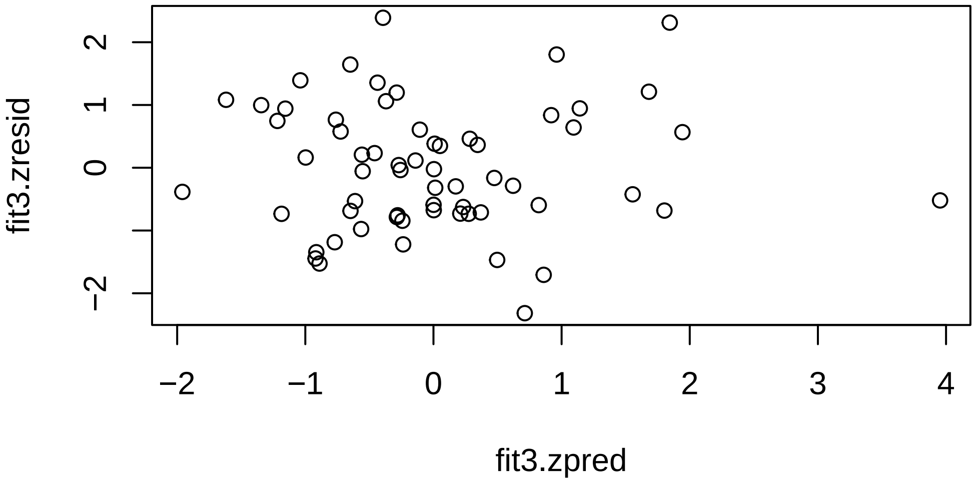


```
car::qqPlot(fit3.zresid)
```



```
[1] 32 28
```

```
plot(fit3.zpred,fit3.zresid)
```

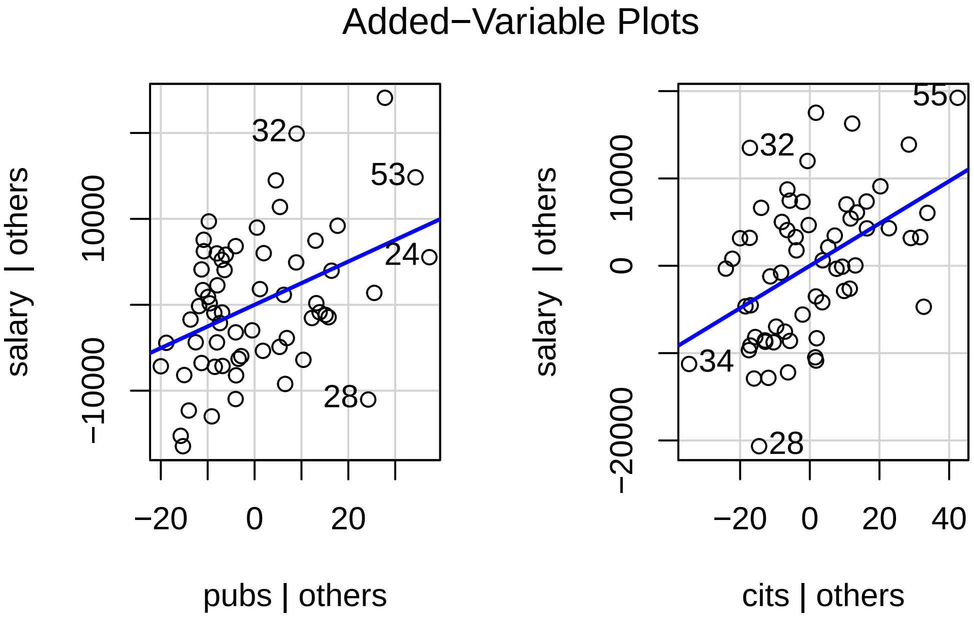


4.3.2 Added Variable Plots

Added variable plots permit evaluation of the so-called partial regressions. These are actually just plots of the partial correlations of each IV (residualized on the other IVs) with Y (also residualized on the other IV). Note that the "test" of these partial correlations is equivalent to the test of the semi-partial correlations and also equivalent to the tests of the individual regression coefficients found in the `summary` output.

See the later chapters on Extensions and use of the **olsrr** package for further treatment of this.

```r
car::avPlots(fit3)
```


Added−Variable Plots
salary | others
10000
−10000
32
53
24
28
−20
0
20
pubs | others
salary | others
10000
0
−20000
55
32
34
28
−20
0
20
40
cits | others

5 Unique and common proportions of variance and Type I vs III Sums of Squares

We can think about an ability to partition the SS of the DV, not only into Regression and Residual components, but to further break down the Regression SS into unique and common parts. This builds on the concepts just reviewed above and connects to earlier work.

THIS SECTION AND THE NEXT ARE THE CRITICAL CONCEPTUAL FRAMEWORK THAT HAS BEEN DEVELOPED FOR MULTIPLE REGRESSION IN THE 510/511 COURSE. WE COVERD THIS WHEN WE WORKED THROUGH THE SPSS IMPLEMENTATION. IF YOU DID NOT CONSOLIDATE IT THEN, OR ON ASSIGNMENTS/EXAM PREP, THEN THIS IS THE TIME/PLACE TO BE CERTAIN ABOUT IT.

In this section we will:

1. find unique SS for each IV, and the common SS
2. Utilize semi-partial correlations and the derived unique/common proportions from ‘mrinfo’
3. use these quantities to reinforce concepts that we have repeatedly covered in slightly different ways
4. use R as a calculator to accomplish some of these things
5. Understand that since we are going to use some rounded quantities from ‘mrinfo’ some error is introduced into our computations. So, some things may not match exactly as the should if the concepts are correct.

5.1 SS components, semi-partial correlations, unique and common proportions/SS

A few components are computed “manually” to establish some basic quantities. SS_{total} is found two different ways (the quantities are returned after this code chunk), and they match. Other quantities are calculated, but not printed here - they are used below.

```
# compute a few SS manually.....
# first obtain a few values to use, SS Total, MSresid, and R squared
SStotal <- var(cohen1$salary)*61 # variance times n-1 gives SS for that variable
# name an object that is the anova summary table for the fit3 model
```

```r
a <- anova(fit3)
#extract the MS Residual and multiple R squared from that table
MSresid <- a$`Mean Sq`[3]
rsquared <- summary(fit3)$r.squared
# use the above quantities to compute SS Regression and SS Residual
SSregression <- rsquared*SStotal
SSresidual <- MSresid*59 # MS resid times its df
# sum them to doublecheck
SSregression + SSresidual # should equal SStotal
```

```
[1] 5746619823
```

```r
SStotal  # show SStotal to doublecheck
```

```
[1] 5746619823
```

Focus on the information value of the semi-partial correlations (we found them with `mrinfo` above). By squaring them, we have the unique proportions of variance accounted for by each IV - and the values match what was reported by `mrinfo` above.

```r
semipartialpub <- .3424509
semipartialcit <- .4041425
# square them to produce "unique" proportions of explained variance
# see that it matches the table from mrinfo
part1sqrd <- semipartialpub^2
part2sqrd <- semipartialcit^2
part1sqrd
```

```
[1] 0.1172726
```

```r
part2sqrd
```

```
[1] 0.1633312
```

Now, we can use the squared semi-partials to find the common proportion of variance shared between the two IVs. Does it match what was returned by `mrinfo`?

```
# now that we know the unique fractions we can find the common proportion
# and it should match the mrinfo value
# recall that rsquared was defined at the beginning of this section
commonprop <- rsquared -part1sqrd - part2sqrd
commonprop
```

```
[1] 0.138912
```

Now compute the unique and common SS. It would be useful to compare these SS to the table found above in the basic model section for fit3 and fit4 and the section that compared the ANOVA summary tables from `anova` and `Anova`.

```
# now compute the unique and common SS
uniqueSSpub <- part1sqrd*SStotal
uniqueSScit <- part2sqrd*SStotal
uniqueSSpub
```

```
[1] 673921157
```

```
uniqueSScit
```

```
[1] 938602084
```

These do match the SS for the two IVs found by using the `Anova` function (with type III SS) from either fit3 or fit4. What is their sum? It should be less than SS regression since each is the unique component and the full $SS_{regression}$ includes the common SS where the two IVs overlap in the salary space.

```
# should these two values add up to SSregression?
# No.   should be something less
uniqueSSpub + uniqueSScit
```

```
[1] 1612523240
```

```
SSregression
```

```
[1] 2410797436
```

Lets find a common SS by using the common proportion seen above.

```
# how much less??????? the amount due to the shared SS
# its proportion is the common proportion
commonprop
```

```
[1] 0.138912
```

```
SScommon <- commonprop*SStotal
SScommon
```

```
[1] 798274196
```

Now verify that all three components sum to $SS_{regression}$, as they should:

```
# do all three sum to SSregression now? compare:
SScommon + uniqueSSpub + uniqueSScit
```

```
[1] 2410797436
```

```
SSregression
```

```
[1] 2410797436
```

5.2 F tests on the unique components?

Now that we have derived unique SS for each IV, can we do an F test? That is, can we test a null hypothesis that an individual IV does not uniquely contribute to the R squared (also phrased as the model fit)? Yes. Just like any F test:

First, find the MS for each IV, based on the unique SS.

```
# can we do tests of these unique SS? Sure, F tests with MSresid as "error" term
# We need to convert the unique SS to MS, so we need df
# each has 1 df, since each represents one variable
# the MS are the SS divided by 1
MSuniquepub <- uniqueSSpub/1
MSuniquecit <- uniqueSScit/1
MSuniquepub
```

```
[1] 673921157
```

```
MSuniquecit
```

```
[1] 938602084
```

Now find the F values by using MSresid as established above.

```
# and now the two F's
Fpubunique <- MSuniquepub/MSresid  #(df are 1,59)
Fcitunique <- MSuniquecit/MSresid  #(df are 1,59)
Fpubunique
```

```
[1] 11.9195
```

```
Fcitunique
```

```
[1] 16.60086
```

```
# can you find these F's anywhere above?
# hint: look in the table of the Anova function (not anova)
```

Yes, they match the F's from the `Anova` tables above, using Type III SS. And, we can take their square roots to compare to the t values for each regression coefficient. This duplicates the illustration seen in a previous chapter.

```
# Now take the square roots of these F's
Fpubunique^.5
```

```
[1] 3.452463
```

```
Fcitunique^.5
```

```
[1] 4.074415
```

```
# look familiar???
# yes, they are the t's that provided the test stat for the regression coefficients
```

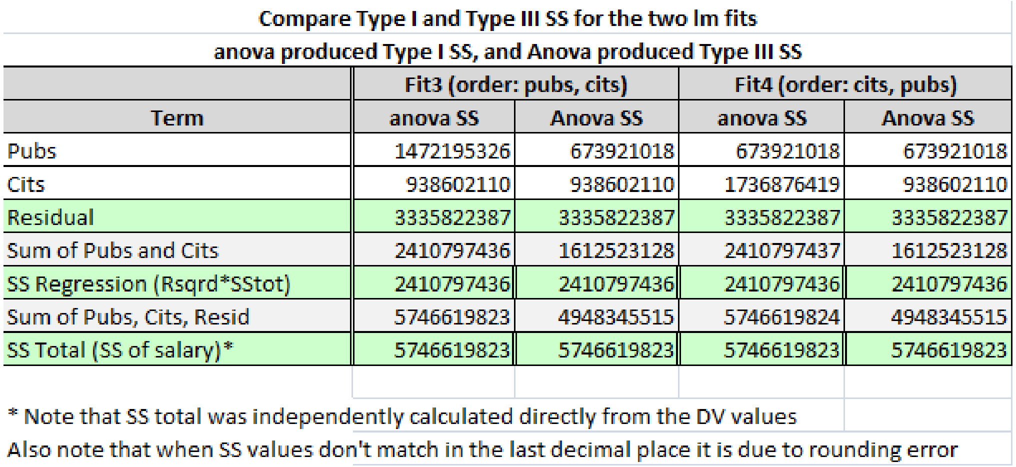
Conclusions?

1. tests of semipartial correlations can be done as F tests by converting to SS and MS
2. these tests are equivalent to the F tests done on TYPE III SS in the Anova function
3. since the F's are the squares of the t's for the two regression coefficients, we conclude that tests of the regression coefficients are tantamount to tests of the semi-partial correlations. This is why we can call them partial (actually semi-partial) regression coefficients.
4. The unique SS are seen to be equivalent to the SS computed above by the 'Anova' function (rather than 'anova'), and are thus equivalent to what are called Type III SS. To be contained……

5.3 Compare and Contrast information from fit3 and fit4

Take some time again to ponder the output from above, comparing comparable values from fit3 and fit4 as we did in the previous chapter. Recall that fit3 and fit4 differed only in the order of entry of the two IVs, with pubs first in fit3. Many/most of the components are the same. The primary differences seem to emerge in the 'anova' vs 'Anova' output. Take the square root of each of the F tests from 'anova' and 'Anova'. Which ones match the t's for tests of the comparable regression coefficients and which differ? This was explored briefly earlier in this document. This section reviews that information again in a more succinct tabular form.

I have created a table of SS produced by both 'anova' and 'Anova' for each of fits 3 and 4 to facilitate this comparison. Make certain you can see, in the above output, where these values came from.



Compare Type I and Type III SS for the two lm fits

anova produced Type I SS, and Anova produced Type III SS

	Fit3 (order: pubs, cits)		Fit4 (order: cits, pubs)	
Term	**anova SS**	**Anova SS**	**anova SS**	**Anova SS**
Pubs	1472195326	673921018	673921018	673921018
Cits	938602110	938602110	1736876419	938602110
Residual	3335822387	3335822387	3335822387	3335822387
Sum of Pubs and Cits	2410797436	1612523128	2410797437	1612523128
SS Regression (Rsqrd*SStot)	2410797436	2410797436	2410797436	2410797436
Sum of Pubs, Cits, Resid	5746619823	4948345515	5746619824	4948345515
SS Total (SS of salary)*	5746619823	5746619823	5746619823	5746619823

* Note that SS total was independently calculated directly from the DV values

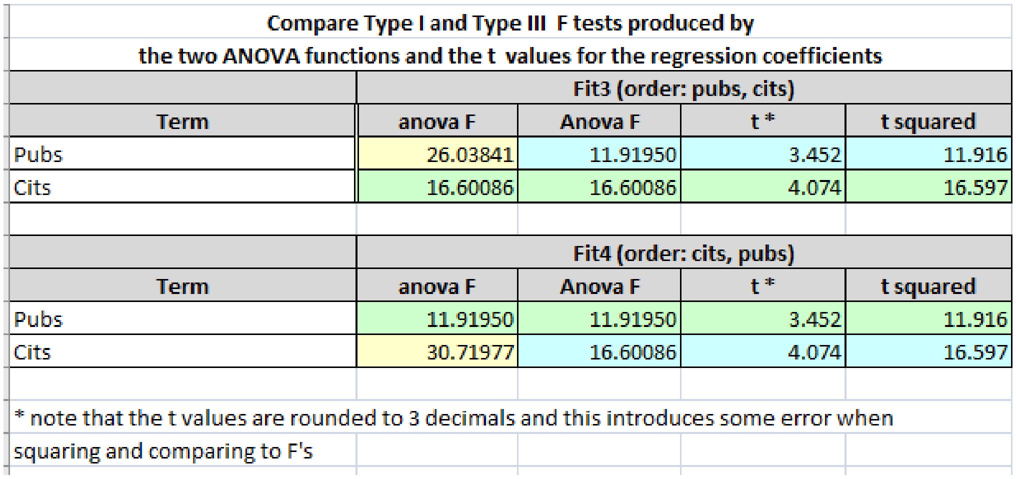
Also note that when SS values don't match in the last decimal place it is due to rounding error

For the moment, notice a few things from this table, recalling that both fit3 and fit4 have the same intercept, regression coefficients and their t's, multiple R squared, SS total, SS residual, SS regression, and F-tests of the whole equation with 2 and 59 df. The SS for pubs and cits should be a partitioning of SS regression.

1. In anova output, the sum of the SS for pubs and cits matches SS regression both for fit3, and fit4 which also match each other.
2. In Anova output the sum of the SS for pubs and cits is always less than SS Regression, and adding them to SS residual yields a quantity less that SS total.
3. For anova output, summing SS for pubs, cits, and residual DOES yield the SS total, as expected.
4. Sometimes SS for pubs (or for cits) from 'anova' and 'Anova' match and sometimes they don't. It depends on which fit one examines. So, we must conclude that the 'order' of model specification has something to do with the difference in Type I and Type III SS. Type III SS are found by treating each IV as if it were the last to enter the model, thus controlling for all other IVs.
5. The reader might also compare the SS for each IV under each model to the SS Regression from when that IV was used in simple regression in this document (in the bivariate computations section)

A similar table compares F test values from 'anova' and 'Anova' for the two fits. The pattern here follows what was seen for SS above since the F's are derived from those SS. The more interesting comparison is the comparison of those F's to the square of the t's found from testing the two regression coefficients in the two models. This comparison lets us get to the heart of what is being tested with the F tests *vis a vis* the way we discussed the null hypotheses for the t tests of the regression coefficients.

The t's test a null hypothesis that the "unique" contribution of that predictor is zero, when controlling for the other predictor(s). And the squares of these t's don't always match the F's from the 'anova' function. They do match for the Anova function. Further discussion of this took place above,in chapter 4, connecting the findings to concepts deriving from our understanding of semi-partial correlations.



Compare Type I and Type III F tests produced by
the two ANOVA functions and the t values for the regression coefficients

	Fit3 (order: pubs, cits)			
Term	**anova F**	**Anova F**	**t ***	**t squared**
Pubs	26.03841	11.91950	3.452	11.916
Cits	16.60086	16.60086	4.074	16.597

	Fit4 (order: cits, pubs)			
Term	**anova F**	**Anova F**	**t ***	**t squared**
Pubs	11.91950	11.91950	3.452	11.916
Cits	30.71977	16.60086	4.074	16.597

* note that the t values are rounded to 3 decimals and this introduces some error when squaring and comparing to F's

This general topic is further addressed below entitled "Work with Sums of Squares a bit more, along with unique and common proportions of variance."

Typically we would not evaluate both models fit3 and fit4 which entered the IVs in different orders since the final model is largely the same (with the caveat about unique vs sequential or Type I vs Type III SS). If you examined the coefficients tables for fit3 and fit4, you found the regression coefficients to be the same (and the t-tests of them as well) So much of the above work is duplicative. Perhaps we would only run fit3. In that case, the kind of comparison in the next section on formally comparing models might still be of interest.

The issue with computation of SS types is revealing in this comparison of fit3 and fit 4 which have only two IVs. The SS computation issue will be addressed several times later in the semester and becomes more involved when the number of IVs exceed two (see also, the "Extensions" chapter) For now, focus on the comparability of the fit3 and fit4 models with regard to overall fit, and with regard to coefficients.

Additional illumination on fit3 and fit4 differences in SS has been connected to semi-partial correlations and unique vs common SS issues addressed here, and that is the part of this topic that is the important conceptual framework. The primary conclusion from the SS comparison work here, is that order of entry of the IVs into a model can affect the SS computation, depending on whether the analyst chooses Type I (sequential) or Type III (unique) SS.

6 Formally Comparing Models

Often we wish to compare regression models that are "nested", meaning that the IV's in one model are a superset of the IV(s) found in another model. One example of that with the simple two-variable system covered so far would be comparison of fit3 (or fit4), the two-IV models, to fit1 (or fit3), the single IV models. This can be done descriptively, inferentially, or with information criteria comparisons.

6.1 Descriptively comparing models

The most common indices for comparing fit of two models are R-squared, adjusted R-squared, and RMSE (root mean squared error). The latter is simply the square root of MSresidual, sometimes called the std error of the estimate, or residual standard error in R. Each are provided by the `summary` output on `lm` fits.

For Illustration, we can compare the simple regression model with only publications as an IV (fit1) with the two-IV model that also includes citations (fit3). As expected with any model that has a superset of IVs that are in smaller model, fit3 has higher R-squared and adjusted R-squared values and a smaller RMSE. The RMSE is in the scale of the dv (salary) and so can be directly interpreted - fit3 std error is over $900 better. The proportions of variance indexed by the two R-squared values also reflect a better fit of about 16% of the variation.

```
cat("Fit1 Rsquared=",summary(fit1)$r.squared, "\n")
```

```
Fit1 Rsquared= 0.2561846
```

```
cat("Fit1 Adjusted Rsquared=",summary(fit1)$adj.r.squared, "\n")
```

```
Fit1 Adjusted Rsquared= 0.2437876
```

```
cat("Fit1 RMSE=",summary(fit1)$sigma, "\n")
```

```
Fit1 RMSE= 8440.403
```

```
cat("Fit3 Rsquared=",summary(fit3)$r.squared, "\n")
```

```
Fit3 Rsquared= 0.4195157
```

```
cat("Fit3 Adjusted Rsquared=",summary(fit3)$adj.r.squared, "\n")
```

```
Fit3 Adjusted Rsquared= 0.3998383
```

```
cat("Fit3 RMSE=",summary(fit3)$sigma, "\n")
```

```
Fit3 RMSE= 7519.266
```

Two additional indices are more commonly used in Econometrics, but can fit these types of OLS models as well.

MAE (mean absolute error) is the average absolute difference between observed Dependent Variable scores and the Yhats. Smaller is, of course, indicative of a better fit. The index is in the scale of the dependent variable, so it is simple to interpret - dollars here.

```
Metrics::mae(cohen1$salary, predict(fit1))
```

```
[1] 6804.567
```

```
Metrics::mae(cohen1$salary, predict(fit3))
```

```
[1] 6056.539
```

MAPE (mean absolute percentage error) is another index often used. Like MAE, it involves comparing observed dv values and yhats. In this index each absolute value of a yhat difference from the observed dv score for that case is calculated (essentially an unsigned residual), Then that difference is expressed as a percentage of the dv score. The mean is then taken across all cases to produce the MAPE index.

```
MLmetrics::MAPE(predict(fit1), cohen1$salary)
```

```
[1] 0.1290218
```

```
MLmetrics::MAPE(predict(fit3), cohen1$salary)
```

```
[1] 0.1139376
```

6.2 Inferentially comparing models

In an attempt to explore how to think about tests of each IV, *vis a vis* the kinds of things implied just above in chapters 4-5, lets use the `anova` function in a different way.

We can pass two linear models to 'anova' and ask it to compare them. The second model in each specification has to contain a superset of the IV's in the first (i.e., the IV used in the first plus at least one more). It is somewhat like the "stepping" idea we introduced in SPSS and 'anova' essentially tests the improvement in fit of the second model over the first. Carefully look at the F tests for these model differences in Fit, and compare them to what you just examined above for the 'anova' vs 'Anova' approaches in the two different orders of the two-IV fit models. The F's that compare the two models are the same as the test of the R squared "increment" that we covered in SPSS work earlier.

We consider beginning with fit1 (only pubs was an IV in that simple regression), and compare fit 3 (which also included cits as an IV) to test the increment in the R-squared produced by the inclusion of cits. The F value matches the Type III SS F test for cits and is the square of the t value that tested the regression coefficient of cits.

```
# compare model 3 to model 1 - stepping approach, evaluating a new variable (cits)
anova(fit1,fit3)# note this is anova, not Anova
```

```
Analysis of Variance Table

Model 1: salary ~ pubs
Model 2: salary ~ pubs + cits
  Res.Df        RSS Df Sum of Sq      F    Pr(>F)
1     60 4274424497
2     59 3335822387  1 938602110 16.601 0.0001396 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The next illustration compares the full model to the simple (restricted) regression that only contained pubs (fit1). The test is therefore the test of an hypothesis that increment in SS accounted for by cits is zero. And this F also matches the F test of the Type III SS seen above. You may have encountered these types of F tests in other software where they are termed tests of R-squared change.

```
# compare model 3 to model 2 - stepping approach, evaluating a new variable (pubs)
anova(fit2,fit3)# note this is anova, not Anova
```

```
Analysis of Variance Table

Model 1: salary ~ cits
Model 2: salary ~ pubs + cits
  Res.Df        RSS Df Sum of Sq      F   Pr(>F)
1     60 4009743405
2     59 3335822387  1 673921018 11.919 0.001034 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This approach can also be used when more than one IV is added in a full model compared to a restricted model. See the "Extensions" chapter.

6.3 Information criteria

It has become standard to compare models with information criteria. In a later chapter, we consider using this approach in deciding variable-selection strategies. Here, a simple application permits introduction of AIC and BIC approaches in comparing the single IV model with the 2-IV model just tested above. Recall that fit3 contained both IV's, fit1 used only "pubs", and fit2 used only "cits".

………………………………

```
AIC(fit1,fit3)
```

```
     df      AIC
fit1  3 1300.973
fit3  4 1287.601
```

```
AIC(fit2,fit3)
```

```
     df      AIC
fit2  3 1297.010
fit3  4 1287.601
```

```
BIC(fit1,fit3)
```

```
     df      BIC
fit1  3 1307.354
fit3  4 1296.110
```

```
BIC(fit2,fit3)
```

```
     df      BIC
fit2  3 1303.391
fit3  4 1296.110
```

7 Further work with residuals and yhats

This chapter is structured with three goals:

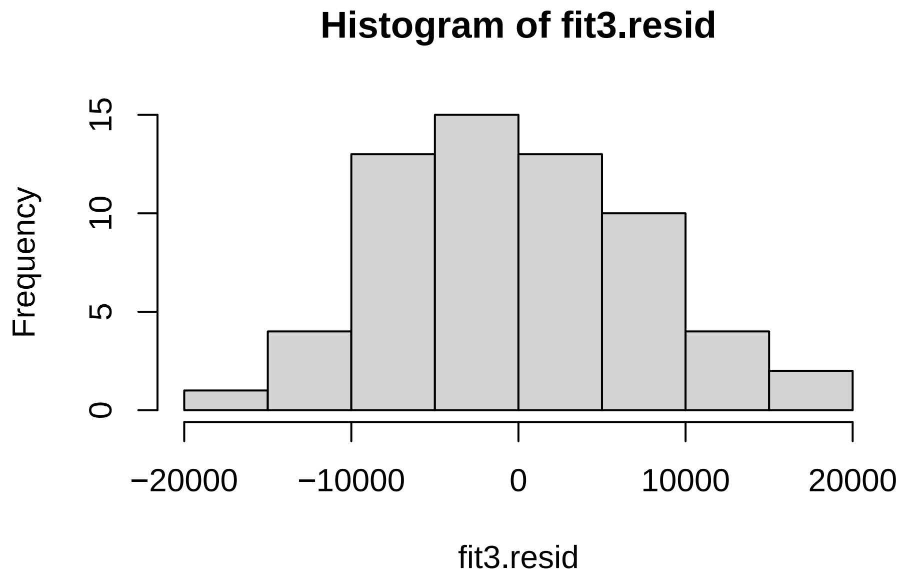
1. Learning how to extract the residuals and the yhats from a 'lm' fit object in order to do additional work with them. You may recall that we learned to "save" residuals and yhats from an SPSS regression. That is also done here.
2. The diagnostic plots from above did not give us a frequency histogram of the residuals or even a simple boxplot. Once extracted, we can easily produce those types of graphs.
3. We can revisit some primary concepts in linear modeling by further examining the meaning of the yhats.

First, extract the residuals and yhats from the model fit (fit3 in this instance). We could add them to the data frame containing the original variables for later potential use (although we can work with the newly created vectors directly here.

```r
fit3.resid <- residuals(fit3)
fit3.pred <- predict(fit3)
#create a new data frame with the original variables plus the extracted residuals and yhats
cohen2 <- cbind(cohen1,fit3.resid,fit3.pred) #note that fit3.pred is the yhats
```

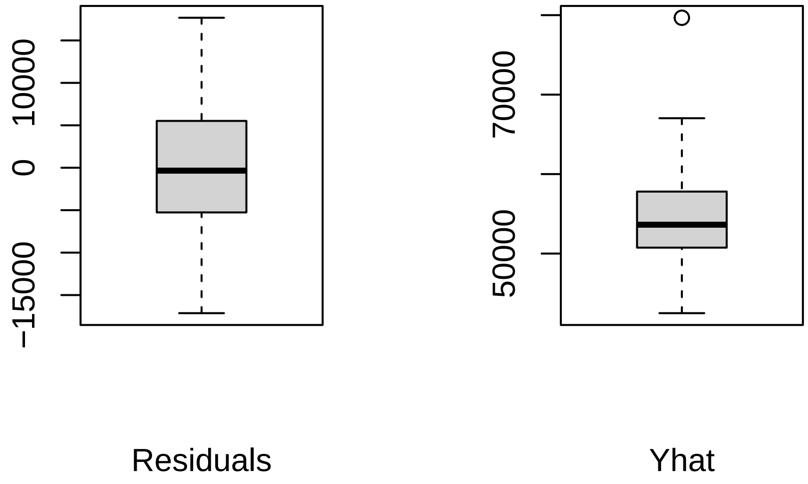
Examine the frequency histogram of the residuals. It looks only slightly positively skewed even though the original DV was somewhat skewed. The normality assumption may not be seriously violated.

```r
# simple frequency histogram of the residuals from fit3
hist(fit3.resid)
```



Boxplots of the residuals and yhats are produced next, revealing one potential outlier in the yhats.

```
# boxplots
layout(matrix(c(1,2),1,2)) #optional 2graphs/page
boxplot(fit3.resid,xlab="Residuals",data=cohen2,col="lightgrey")
boxplot(fit3.pred, xlab="Yhat",data=cohen2,col="lightgrey")
```



Numerical summaries of the residuals and yhats are provided by the `describe` function.

```
psych::describe(fit3.resid)
```

```
   vars  n mean      sd  median trimmed     mad       min      max    range
X1    1 62    0 7394.97 -341.27 -160.91 7519.46 -17133.14 17670.34 34803.48
   skew kurtosis     se
X1 0.22     -0.4 939.16
```

```
psych::describe(fit3.pred)
```

```
   vars  n     mean      sd   median  trimmed     mad      min      max range
X1    1 62 54815.76 6286.59 53630.77 54275.57 4982.75 42497.52 79670.52 37173
   skew kurtosis    se
X1 1.13     2.45 798.4
```

Now that we have the yhat values available, we can do one more analysis that will be familiar. If we correlate the yhats with salary (the DV), we find an interesting quantity, when we square it.

```
# correlate the original DV (salary) with the yhats from the two-IV linear model
r_y_yhat <- with(cohen2, cor(salary, fit3.pred))
r_y_yhat
```

```
[1] 0.6477003
```

```
# square that value
r_y_yhat^2 #does the value look like something you recognize?
```

```
[1] 0.4195157
```

When covering this characteristic at earlier points, it was emphasized that the yhats are the best linear combination of the IVs, and carry all the information about the predictability of the DV from those IVs. It is a core feature of regression that the relationship between the DV and the yhats reveals the strength of the prediction, the multiple R (or R-squared).

8 Examine the fit of the plane in a 3D wireframe plot:

R has useful facilities for creating a three dimensional scatter plot. It is also possible to plot the model fit onto that 3D scatter plot. The two-IV model fits a plane. This section creates the model, generates the scatter plot, adds the plane, and permits rotation of the plot if the code is executed in R or RStudio.

Initially we have to relabel the variables in order to use the 3d scatter plot function below.

```
# to simplify the use of the scatter3D function, we re-label the variables.
# scatter3D expects the "Y" variable from a linear model to be in the "z" dimension of the 3
z <- cohen1$salary
x <- cohen1$pubs
y <- cohen1$cits
```

```
# Again fit the two-IV model, but using the x,y,z variables
#  x,y, and z are used for purposes related to the 3D surface plotting below.
# This model is the same as fit3 examined above
fit3d1 <- lm(z~x+y)
#summary(fit.dep1b)
tidyfit.3d1 <- tidy(fit3d1) # nicer table than with summary
kable(tidyfit.3d1, format="markdown")
```

term	estimate	std.error	statistic	p.value
(Intercept)	40492.9715	2505.39437	16.162314	0.0000000
x	251.7500	72.91896	3.452463	0.0010337
y	242.2977	59.46809	4.074415	0.0001396

Drawing the 3d scatter plot is somewhat complicated and the student in 510/511 is not expected to master the code at this point. However, one should be able to take this code and modify it to be used in and two-IV model fit with other data sets.

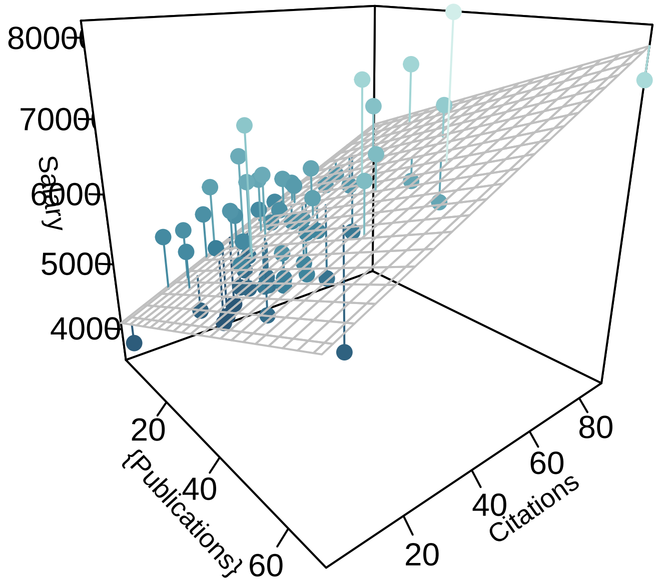
In order to draw the 3D scatter plot we need to do the following:

1. Set up a grid of hypothetical points along a plane (I chose 21 lines for visual style reasons). The grid is established with minimal and maximal values in the x and y dimensions by using the minimal and maximal values for the x and y variables (pubs and cits in our case)
2. Create a matrix of predicted scores (in the y scale), using the prediction equation that we called fit3d1 above.
3. use the 'scatter3D' function from the 'plot3D' package. This draws the basic scatter plot, permits some color control of points, allows text and label editing, and permits addition of a surface.

```
# set up a grid required for the plane drawn by the surface argument of scatter3D
grid.lines = 21
x.pred <- seq(min(x), max(x), length.out = grid.lines)
y.pred <- seq(min(y), max(y), length.out = grid.lines)
xy <- expand.grid( x = x.pred, y = y.pred)

z.pred <- matrix(predict(fit3d1, newdata = xy),
                 nrow = grid.lines, ncol = grid.lines)
# fitted points for droplines to surface if we want to draw them
fitpoints <- predict(fit3d1)
```

This 3D scatter plot capability is useful in one important manner not demonstrated in the html or pdf versions of this document. If the code is run in R (preferable RStudio), the 'scatter3D' function can be followed by a 'plotrgl' function (commented out here). This will pop up a graphics window and the user can use the mouse to manually rotate the 3D image. This provides superb visualization capability. In the scatter3D implementation here the theta and phi values are chosen to produce what I thought was the best rendition of the 3D image for the 2 dimensional rendering required for print documents.

```
# scatter plot with regression plane
scatter3D(x, y, z, pch = 20, cex = 1.5, cex.lab=.8,
          theta = 50, phi = 20, ticktype = "detailed",
#          theta = 50, phi = 18, ticktype = "detailed",
          zlim=c(35000, 82000),
          #cex.axis=.9,
          xlab = "{Publications}", ylab = "Citations",
          zlab = "Salary",
          surf = list(x = x.pred, y = y.pred, z = z.pred,
                      facets = NA,
                      col="grey75",
                      fit = fitpoints),
          colkey=F,
          #col=terrain.colors(length(z)),
          #col=paletteer_c("ggthemes::Classic Blue", 30),
          col=paletteer_c("grDevices::Teal", 30),
          main = "",
          plot=T)
```



```
# execute the plotrgl function outside of the pdf or html doc to interactively work with the
# plotrgl()
```

The comment in the last lines of the code chunk indicate how this 3D scatterplot can be used interactively with RGL tools (`plotrgl`). The reader might also see how this type of scatterplot can by dynamically rotated in a shiny app at this url:

Shiny app for 2-IV linear models

9 Inferential tests for assumptions

There are three primary assumptions that linear modeling with the standard NHST t and F tests rely on:

1. The relationships among variables are best described by linear functions.
2. The residuals from a model are normally distributed.
3. Homoscedasticity.

Graphical evaluation of these assumptions has been the priority in previous considerations, as well in this document. Now we can turn to inferential tests regarding these assumptions.

9.1 Evaluation of the Residual Normality Assumption

Prior work with SPSS and R for linear models has only employed a graphical assessment to the normality assumption for residuals (and skewness/kurtosis computation). Frequency histograms and normal probability plots are useful, but at times one may wish to do an inferential test of a null hypothesis that residuals are normally distributed. Quite a few such inferential tests have been developed, and many are available in R.

While we can easily accomplish these tests, the analyst should be concerned about a binary outcome decision process in evaluation of the assumption. The real question should be something like "with the degree of non-normality present, how much of an impact on the tests of regression coefficients and the test of the whole model is there?" Strong guidance on this is lacking. However, some discussion can be found in standard regression textbooks and the reader is advised to consult Fox (2016), or any number of other sources (e.g., Cohen et al. (2003), Cook & Weisberg (1999), Darlington (1990), Howell (2013), Weisberg (2014), or Wright & London (2009)).

The options available in R will be presented here without regard to that more important question. The Anderson-Darling test may be the test that is most recommended. Historically, the Shapiro test is probably the most commonly used one, based on availability in other software. But with the 'nortest' package in R, many others are also available.

First, we can implement the historically common Shapiro-Wilk test:

```
shapiro.test(residuals(fit3))
```

```

    Shapiro-Wilk normality test

data:  residuals(fit3)
W = 0.98639, p-value = 0.7239
```

This next code chunk shows code executing five different tests from the **nortest** package. But only one (Anderson-Darling) is executed here to keep this document small. The reader may recall from graphical approaches accomplished above that there is a slight degree of non-normality in the residuals from the basic two-IV model (fit3 or fit4). Is it a significant departure from normality, based on the sample size employed in this study?

```
# tests for normality of the residuals
#library(nortest)
ad.test(residuals(fit3)) # Anderson-Darling test (from 'nortest' package )
```

```

    Anderson-Darling normality test

data:  residuals(fit3)
A = 0.34507, p-value = 0.4742
```

```
#cvm.test(residuals(fit3)) #Cramer-von Mises test ((from 'nortest' package )
#lillie.test(residuals(fit3)) #Lilliefors (Kolmogorov-Smirnov) test (from 'nortest' package )
#pearson.test(residuals(fit3)) #get Pearson chi-square test (from 'nortest' package )
#sf.test(residuals(fit3)) #get Shapiro-Francia test (from 'nortest' package )
```

Neither of these tests reach significance thresholds at α=.05 and this is not surprising given the small amount of skewness visualized in the graphical assessments of the residuals seen in above sections of this document.

9.1.1 Evaluation of the normality assumption by testing skewness and kurtosis

Another approach to inference regarding the normality assumption is to test whether skewness and/or kurtosis of the residuals deviates from a null hypothesis that they are zero (as would be the case for a normal distribution). Once again, this document is more concerned with showing how such tests can be accomplished in R. Discussions about their interpretation and whether they should be used are left to other parts of the course and to the recommendations of the relevant textbooks.

R has two packages (**tseries** and **moments**) that contain functions to test skewness and/or kurtosis:

```
# also....... another way to evaluate normality is to test for skewness and kurtosis
#library(tseries)
jarque.bera.test(residuals(fit3)) # Jarque Bera test for normality of a variable
```

```

	Jarque Bera Test

data:  residuals(fit3)
X-squared = 0.77247, df = 2, p-value = 0.6796
```

```
#library(moments)
agostino.test(residuals(fit3))
```

```

	D'Agostino skewness test

data:  residuals(fit3)
skew = 0.22341, z = 0.78060, p-value = 0.435
alternative hypothesis: data have a skewness
```

```
anscombe.test(residuals(fit3)) # Anscombe-Glynn test of kurtosis
```

```

	Anscombe-Glynn kurtosis test

data:  residuals(fit3)
kurt = 2.68478, z = -0.25858, p-value = 0.796
alternative hypothesis: kurtosis is not equal to 3
```

Again, none of these tests would allow us to reject a null hypothesis of residual normality.

We might also simply perform a Z test by taking the ratio of skewness and kurtosis coefficients to their std errors (called a large-sample approximation in using the std normal Z here). We can obtain the coefficients and std errors in a couple ways:

We can write our own function to test skewness and kurtosis as the Z approximation called the large sample approximation suggested above. First, create the function and test skewness:

```
# we can build our own tests, since we know the
# std errors of skewness and kurtosis (fall semester work)
# these functions test the G1 and G2 statistics with their
# large sample std errors against nulls that the parameters are zero (as in normal distribs)
skewness.test <- function (lmfit) {
  require(psych)
  G1 <- psych::describe(residuals(lmfit),type=2)$skew # pull G1 from psych:::describe
  skewness <- G1
  N <- length(residuals(lmfit))
  stderrskew <- sqrt(6/N)
  teststat <- skewness/stderrskew
  pvalue <- 2*(pnorm(abs(teststat), lower.tail=F))
  outskew <- data.frame(cbind(N,G1,stderrskew,teststat,pvalue))
  colnames(outskew) <- c("N","G1","Std Error", "Z Test Stat","2-tailed p")
  rownames(outskew) <- c("")
  return(outskew)
}

# use this approach (z test) with large N. It returns a two-tailed p value
skewness.test(fit3)
```

```
  N        G1 Std Error Z Test Stat 2-tailed p
 62 0.2289903 0.3110855   0.7361008   0.4616693
```

And now test kurtosis:

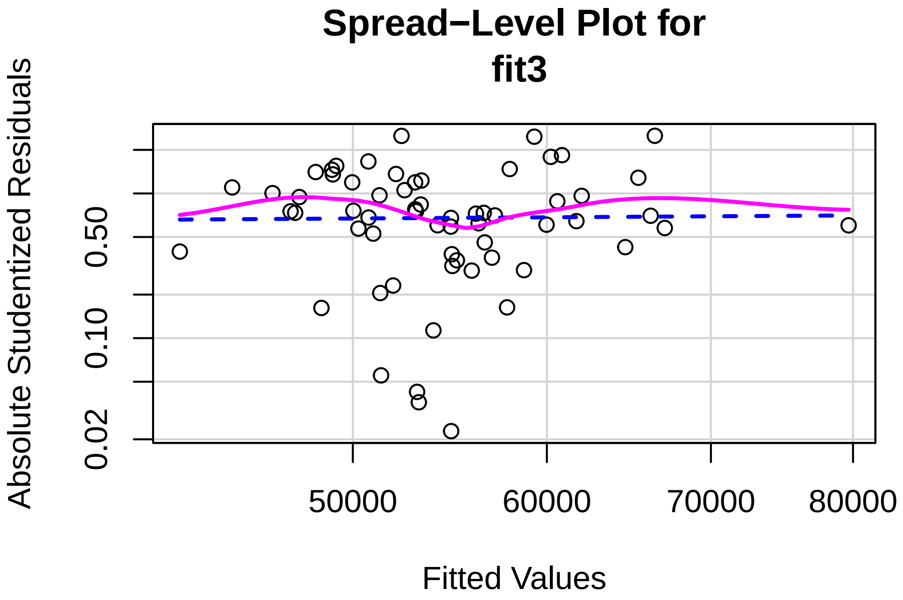
```
# now kurtosis
kurtosis.test <- function(lmfit) {
  require(psych)
  G2 <- psych::describe(residuals(lmfit),type=2)$kurtosis # pull G2 from psych:::describe
  kurtosis <- G2
  N <- length(residuals(lmfit))
  stderrkurt <- sqrt(24/N)
  teststat <- kurtosis/stderrkurt
  pvalue <- 2*(pnorm(abs(teststat), lower.tail=F))
  outk <- data.frame(cbind(N,G2,stderrkurt,teststat,pvalue))
  colnames(outk) <- c("N","G2","Std Error", "Z Test Stat","2-tailed p")
  rownames(outk) <- c("")
  return(outk)
}

kurtosis.test(fit3) # use this approach (z test) with large N. It returns a two-tailed p valu
```

```
  N         G2 Std Error Z Test Stat 2-tailed p
 62 -0.2388151  0.622171  -0.3838415  0.7010959
```

9.2 Inferential test of Homoscedasticity

Evaluation of homoscedasticity shares the same graphical-approach history in our prior work as the normality assumption discussed above. There is also the question of relative impact of varying decrees of violation of the assumption that needs to be considered (we have discussed this previously in the context of the 2-sample location test assumption of homogeneity of variance which is essentially the same assumption). But the narrow goal here is the illustration of approaches in R. First, we redo a plot of residuals against yhats using the `spreadLevelPlot` function from the **car** package. The plot gives a hint of larger residual spread for mid range values of yhats, but the pattern is carried by only a few data points.

```
# also a plot slightly different from the base system plot on the fit
# from the car package
spreadLevelPlot(fit3)
```



```
Suggested power transformation:  0.8964362
```

The `ncvTest` function (ncv standing for non-constant variance), also from the **car** package provides chi-square based test statistic that evaluates a null hypothesis of homoscedasticity. The test is more typically called the “Score” test.

```
# tests of homoscedasticity
# the ncvTest function is a test of non-constant error variance, called the Score test from
ncvTest(fit3)
```

```
Non-constant Variance Score Test
Variance formula: ~ fitted.values
Chisquare = 0.2914146, Df = 1, p = 0.58932
```

Another test of homoscedasticity is one that is frequently used. The Breusch-Pagan test is implemented in the `bptest` function found in the **lmtest** package. Neither it, nor the Score test yielded a result that would challenge the null hypothesis of homoscedasticity.

```
#library(lmtest)
bptest(fit3)
```

```

    studentized Breusch-Pagan test

data:  fit3
BP = 2.665, df = 2, p-value = 0.2638
```

9.3 Global Evaluation of Linear Model Assumptiions using the ‘gvlma’ package.

A paper in JASA (E. A. Pena & Slate, 2006) outlined an approach to model assumption evaluation that is novel (and mathematically challenging). An R implementation is now available in the **gvlma** package. Initially, the approach evaluates a global null hypothesis that all assumptions of the linear model hypothesis tests are satisfied:

1. Linearity
2. Normality evaluation of skewness)
3. Normality (evaluation of kurtosis)
4. Link Function (linearity and normality)
5. Homoscedasticity

If this global null is rejected, then evaluation of each of four component assumptions is evaluated individually.

Fortunately, the ‘gvlma’ function is simple to use and the output is largely the standard NHST p value for each test.

A caveat here is that this type of approach is even more likely to be the target of the kinds of discussion we have had before on whether binary decision inferential tests are the way to act on knowledge of whether model assumptions are satisfied. I have not yet found a literature commenting on or assessing the Pena & Slate procedure (2006), so use of this method is only recommended with reservation. It does serve as a good example of how new/novel methods become rapidly available in R and are sometimes quite simple to implement.

The function provides tests, in this order, of:

- The global test of the four assumptions
- Skewness
- Kurtosis
- Linearity (link function test)
- homoscedasticity

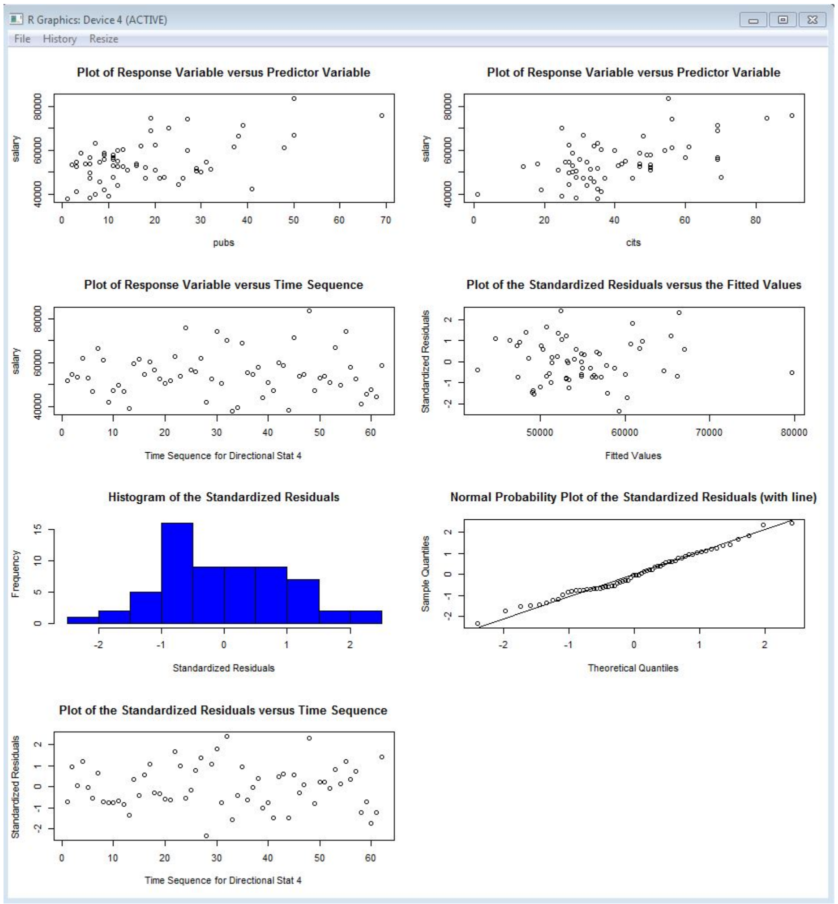
```
# both the package and the function are called 'gvlma'
#library(gvlma)
# the gvlma function only requires a 'lm' model fit object as a single argument to yield the
gvlmafit3 <- gvlma(fit3)
#print out the tests
gvlmafit3
```

```

Call:
lm(formula = salary ~ pubs + cits, data = cohen1)

Coefficients:
(Intercept)         pubs         cits  
    40493.0        251.8        242.3  


ASSESSMENT OF THE LINEAR MODEL ASSUMPTIONS
USING THE GLOBAL TEST ON 4 DEGREES-OF-FREEDOM:
Level of Significance =  0.05

Call:
 gvlma(x = fit3)
```

```
                    Value p-value                Decision
Global Stat         1.8686  0.7599 Assumptions acceptable.
Skewness            0.5158  0.4727 Assumptions acceptable.
Kurtosis            0.2567  0.6124 Assumptions acceptable.
Link Function       0.2047  0.6509 Assumptions acceptable.
Heteroscedasticity 0.8914  0.3451 Assumptions acceptable.
```

```
#plot relevant information
# note: this plot does not play nice in the R markdown format.
#       So I generated it outside of the .Rmd file, saved the image and displayed the image
#plot(gvlmafit3)
```

It is possible to generate a set of very useful plots by passing the `gvlma` object just created to the base system `plot` function (`plot(gvlmafit3)` as was commented out in the code chunk just above). However, the figure doesn't play nice with `rmarkdown`, so I created it separately, and saved the figure to a .png file that is displayed here.


R Graphics: Device 4 (ACTIVE)
File
History
Resize
Plot of Response Variable versus Predictor Variable
salary
pubs
40000
60000
80000
0
10
20
30
40
50
60
70
Plot of Response Variable versus Predictor Variable
salary
cits
0
20
40
60
80
Plot of Response Variable versus Time Sequence
salary
Time Sequence for Directional Stat 4
Plot of the Standardized Residuals versus the Fitted Values
Standardized Residuals
Fitted Values
50000
60000
70000
80000
Histogram of the Standardized Residuals
Frequency
Standardized Residuals
0
5
10
15
-2
-1
0
1
2
Normal Probability Plot of the Standardized Residuals (with line)
Sample Quantiles
Theoretical Quantiles
Plot of the Standardized Residuals versus Time Sequence
Standardized Residuals
Time Sequence for Directional Stat 4

10 A Transition Chapter

10.1 Summary up to this point and code

One of the recognized features in R that both infuriates and satisfies is that there are always many different ways to do the same thing. Some of that is reflected in the above chapters. In order to facilitate translation of the work in the chapters above into a usable approach for the reader's own data sets, I provide a series of four code chunk templates here. The core methods are executed (leaving out EDA things) followed by some assumption evaluation, model criticism and influence analysis. This set of steps can be viewed as a fairly full implementation of the OLS approach although a few followup approaches have been left out.

```
cohen1 <- read.csv("data/cohen.csv", stringsAsFactors=T)
#changing to numeric may not be necessary if they are integer
cohen1$degree_yrs <- as.numeric(cohen1$degree_yrs)
cohen1$pubs <- as.numeric(cohen1$pubs)
cohen1$cits <- as.numeric(cohen1$cits)
cohen1$salary <- as.numeric(cohen1$salary)
fit3 <- lm(salary~pubs+cits, data=cohen1)
summary(fit3)
confint(fit3, level=0.95) # CIs for model parameters

kable(tidy(anova(fit3))) # Type I SS
kable(tidy(Anova(fit3,type="III")))# Type III SS

bcdstats::mrinfo(fit3, minimal=TRUE)
# olsrr::ols_correlations(fit3)
AIC(fit3)
BIC(fit3)
```

```
rcompanion::plotNormalHistogram(residuals(fit3))
car::qqPlot(fit3, distribution="norm", id=F)
layout(matrix(c(1,2,3,4),2,2)) # optional 4 graphs/page
plot(fit3)

ad.test(residuals(fit3)) # Anderson-Darling test (from 'nortest' package )
```

```
lmtest::bptest(fit3)

car::avPlots(fit3)
```

```
# Influence measures from chapter 11
influence.measures(fit3)
olsrr::ols_coll_diag(fit3)
# Cook's D plot
# identify D values > 4/(n-k-1)
threshold  <- 4/((nrow(cohen1)-length(fit3$coefficients)-2))
plot(fit3, which=4, cook.levels=threshold)
car::influencePlot(fit3, id=TRUE,
              main="Influence Plot", sub="Circle area is proportial to Cook's Distance" )
```

```
# olsrr functions from chapter 14
ols_regress(fit3)
ols_correlations(fit3)
ols_coll_diag(fit3)
ols_test_normality(fit3)
ols_plot_resid_fit(fit3)
ols_plot_cooksd_chart(fit3)
ols_plot_dfbetas(fit3)
ols_plot_resid_lev(fit3)
ols_plot_resid_stud_fit(fit3)
```

10.2 Commentary

At this point in this document, we have covered essentially all of the methods that we first utilized in SPSS and have added several others into the arsenal of tools. The document simultaneously tried to weave conceptual integration/review into the templates for accomplishing the analyses with R. The remainder of the document extends the set of analyses to things not yet covered in the course. This includes resampling methods, regression diagnostics and influence analysis, model criticism, models with more than two IVs, and models with categorical variables. Much of the following are presented with the dual goals of R instruction and further consolidation of the conceptual framework that comprises linear modeling with multiple IV's and serves as a template for conceptual elaboration of those topics at later points in time.

11 Regression diagnostics and Influence Analysis

In a future section of the course, we will look more carefully at procedures for diagnosing problem issues in linear model fits, including the violation of assumptions. We will also build concepts that address how influential specific cases and variables are for the particular outcome that our linear modeling produces. This section of the current document will simply present code for a long list of these things without much comment, annotation, or explanation. It serves to put these capabilities in place for the point in time when we will need them The illustrations are all based on the fit3 model evaluated above:

11.1 Standardized, Studentized and Deleted Residals

It is common to examine a variety of standardized residuals as a first step in evaluating Influence of individual cases. Residuals can be calculated either using the full data set (full model) or a modified model fit that does not include the specific case in question. If one suspects that a case is an outlier, then the logic is that the error estimate should, perhaps not include that case. These types of residual calculations would be called “deleted” residuals. Residuals calculated on the full model fit are thus calculated “internally” and those on the reduced model lacking one case at a time are called “internal” computations.

Another distinction is between what I will call “regular” residuals and studentized residuals, depending on whether the residual error estimate used in the denominator is adjusted for leverage for that case (studentized), or not (regular). The reader should perhaps read a standard text such as Fox for perhaps a more lucid and full explanation.

However, there is a major terminology/nomenclature discrepancy among various software implementations of the same indices. Since the course has covered the SPSS implementations, this section will point out differences between inconsistently named indices in SPSS and R.

When standardizing residuals, each residual is scaled to a std deviation (regular) or a std deviation weighted by leverage (studentized). In addition these denominators can be calculated either using all of the cases (internal standardization) or by excluding the case in question (this is typically called external standardization). Unfortunately, labels for these entities are non-standard (but the “internal” and “external” labels do have standard meaning). I will list here what each of them are called and how calculated, separately for SPSS and R.

SPSS

- Unstandardized Residual (Raw Residual, called RES when “saved” from the dialog box or RESID in syntax)
- Deleted Residual (Raw Residual, called DRE when saved or DRESID in syntax
- Standardized Residual (“Internal” - all cases used for SD; called ZRES when “saved” or ZRESID in syntax)
- Studentized Residual (“Internal” - all cases used for SD; called “SRE” when saved or SRESID in syntax)
- Studentized Deleted Residual (“External - deleting the individual case in question; called”SDR” when saved or SDRESID in syntax)

R

Only Two indices are provided as base R functions:

- `rstandard` calculates what are called standardized residuals in R, but they use the leverage adjustment in the standardizer. Thus they match what SPSS calls studentized residuals
- `rstudent` calculates what are called studentized residuals, but they use externally calculated estimates of residual error. They are studentized in the sense that leverage is included in the denominator (standardizer). This function produces what SPSS calls Studentized Deleted Residuals.
- I have not found a direct function in R for computing what SPSS calls standardized (zresid) or deleted unstandardized (dresid) residuals, but the former is calculated more or less manually as the second object in this code chunk.

```
#  Residual Examination
#  create the whole suite; look at each by doing, e.g., print(sresids)
unstandardized <- residuals(fit3) # raw residuals

# internally calculated standardized residuals - manual computation to match SPSS
# denominator is square root of MSresidual, or std error of the estimate
zresids <- residuals(fit3)/sqrt(sum(residuals(fit3)^2)/df.residual(fit3))

# what R calls "standardized" residuals are internally standardized and studentized.
zresidsr <- rstandard(fit3)

# what R calls "studentized" with `rstudent` are studentized and externally calculated (targe
studresids1 <- rstudent(fit3)

# A function from the **MASS** package does the same studentized-deleted (external) computati
studresids2 <- MASS::studres(fit3)
```

```
# examples of exploration
headtail(zresids)
```

```
Warning in headtail(zresids): headtail is deprecated.  Please use the headTail
function
```

```
  1                    2                    3
h "-0.699982866797094" "0.926026284899153" "0.0417119267530487"
  "...       ..."      "...       ..."     "...       ..."
t "-0.676009297297"    "-1.67796447498741" "-1.20061722275395"
  4
h "1.17726328968242"
  "...       ..."
t "1.36950767166633"
```

```
max(zresids)
```

```
[1] 2.350008
```

```
min(zresids)
```

```
[1] -2.278566
```

```
# headtail(zresidsr)
# headtail(studresids1)
# headtail(studresids2)
```

11.2 Influence Indices

Several indices are calculated here that produce values for each case in the data set. I show how, with subsetting, to obtain individual case values for some of these (dfbetas and dffits), but I don't print all values for any of them. The next to last code line in this chunk reminds us how to use `headtail` to examine the first few and last few elements of an object - dfbetas are shown. The commented last line would produce all Cooks D values for the whole data set.

```
#  Residual Examination
#  create the whole suite; look at each by doing, e.g., print(sresids)

leverage_hats <- hatvalues(fit3) #get the leverage values (hi)
cooksD <- cooks.distance(fit3) #get Cook's distance
dfbetas <- dfbetas(fit3) #calculate all dfbetas
dfbetas_4_0 <- dfbetas(fit3)[4,1] #dfbeta for case 4, first coefficient (i.e., b_0, the inte
dffits <- dffits(fit3) #All dffits
dffits_4 <- dffits(fit3) [4] #dffits for case 4
# examine dfbetas, for example
headtail(dfbetas)
```

```
Warning in headtail(dfbetas): headtail is deprecated.  Please use the headTail
function
```

```
#print(cooksD)
```

```
    X.Intercept.  pubs  cits
1           0.01  0.02 -0.06
2           0.16 -0.11 -0.06
3              0 -0.01  0.01
4           0.11  0.01 -0.05
...          ...   ...   ...
59         -0.07  0.06  0.01
60          0.23  0.28 -0.48
61         -0.14 -0.13  0.16
62          0.22 -0.15 -0.07
```

A more efficient method would be to use the `influence.measures` function from the base R **stats** package.

```
influence.measures(fit3)
```

```
Influence measures of
	 lm(formula = salary ~ pubs + cits, data = cohen1) :

      dfb.1_ dfb.pubs dfb.cits    dffit cov.r   cook.d    hat inf
1   0.008129  0.01960 -0.05551 -0.10620 1.049 3.79e-03 0.0222
2   0.160844 -0.10563 -0.06110  0.19120 1.047 1.22e-02 0.0394
3   0.000932 -0.00799  0.00567  0.01024 1.114 3.56e-05 0.0548
```

```
4   0.107281  0.00601 -0.05468  0.16285 0.997 8.78e-03 0.0183
5  -0.002124  0.00261 -0.00107 -0.00532 1.075 9.59e-06 0.0213
6  -0.049875  0.05867 -0.00742 -0.09073 1.068 2.78e-03 0.0287
7  -0.026369  0.11332 -0.00172  0.14604 1.083 7.18e-03 0.0490
8   0.080617 -0.18206 -0.02083 -0.22213 1.129 1.66e-02 0.0913
9  -0.148543  0.02480  0.10377 -0.15437 1.069 8.01e-03 0.0423
10 -0.083773 -0.05091  0.07723 -0.12682 1.051 5.40e-03 0.0276
11 -0.069015 -0.10187  0.09357 -0.14877 1.078 7.45e-03 0.0460
12 -0.084348 -0.04385  0.06963 -0.13013 1.040 5.67e-03 0.0235
13 -0.218418  0.05384  0.12922 -0.24033 0.989 1.90e-02 0.0306
14  0.011574  0.03139 -0.01104  0.05591 1.071 1.06e-03 0.0236
15  0.051671 -0.05597 -0.04532 -0.10408 1.105 3.66e-03 0.0564
16  0.069421 -0.04461 -0.01862  0.09308 1.062 2.92e-03 0.0258
17  0.090380 -0.04154 -0.01774  0.14520 1.013 7.01e-03 0.0186
18  0.032304  0.06033 -0.08303 -0.09826 1.164 3.27e-03 0.0998   *
19 -0.006172  0.02484 -0.02349 -0.05022 1.074 8.54e-04 0.0247
20 -0.057447 -0.08102  0.07453 -0.12229 1.078 5.04e-03 0.0412
21 -0.036962 -0.07507  0.04825 -0.11265 1.066 4.27e-03 0.0317
22  0.174364 -0.15952 -0.00875  0.27497 0.940 2.45e-02 0.0266
23  0.216307 -0.06124 -0.14042  0.22349 1.049 1.66e-02 0.0471
24  0.257508 -0.25732 -0.16319 -0.37964 1.444 4.86e-02 0.2846   *
25  0.009834  0.02041 -0.03011 -0.03848 1.110 5.02e-04 0.0527
26  0.100073 -0.04771 -0.03938  0.12352 1.049 5.12e-03 0.0258
27  0.174261  0.07261 -0.15269  0.23376 0.986 1.80e-02 0.0286
28 -0.086936 -0.60221  0.29487 -0.69170 0.842 1.47e-01 0.0729   *
29  0.273447 -0.08859 -0.17975  0.27913 1.053 2.59e-02 0.0605
30 -0.124739  0.08218  0.17987  0.33383 0.917 3.57e-02 0.0319
31  0.002373  0.04870 -0.06672 -0.11809 1.052 4.69e-03 0.0258
32  0.336704  0.22122 -0.34618  0.48675 0.804 7.25e-02 0.0366   *
33 -0.183200  0.24198 -0.02250 -0.32065 0.972 3.35e-02 0.0410
34 -0.130803  0.00208  0.11464 -0.13338 1.162 6.02e-03 0.1017   *
35 -0.129871 -0.06761  0.22397  0.25732 1.076 2.21e-02 0.0668
36  0.072456  0.09240 -0.15929 -0.18335 1.127 1.13e-02 0.0833
37 -0.002423 -0.00372  0.00342 -0.00529 1.109 9.49e-06 0.0512
38  0.002705 -0.04443  0.04141  0.07201 1.082 1.75e-03 0.0346
39 -0.109550  0.03749  0.04423 -0.14335 1.025 6.86e-03 0.0214
40 -0.048498 -0.11486  0.07640 -0.15461 1.069 8.03e-03 0.0425
41 -0.004516 -0.09430 -0.01858 -0.21803 0.963 1.55e-02 0.0214
42 -0.011109 -0.04497  0.06025  0.08860 1.080 2.65e-03 0.0359
43  0.015572 -0.06537  0.05092  0.10612 1.065 3.80e-03 0.0301
44 -0.210905  0.13106  0.07318 -0.26062 0.975 2.22e-02 0.0310
45 -0.102923  0.07575  0.09629  0.16787 1.122 9.50e-03 0.0781
46 -0.003300  0.01153 -0.01793 -0.04205 1.070 5.99e-04 0.0202
```

```
47  0.005028 -0.00769  0.00479  0.01661 1.075 9.35e-05 0.0211
48 -0.283943  0.71171  0.03684  0.83871 0.861 2.15e-01 0.1010   *
49 -0.073208 -0.01361  0.04441 -0.10915 1.041 4.00e-03 0.0193
50  0.026590  0.00231 -0.01861  0.03253 1.077 3.58e-04 0.0246
51  0.015143 -0.03114  0.01331  0.04325 1.086 6.34e-04 0.0339
52 -0.008137 -0.00340  0.00755 -0.01044 1.090 3.70e-05 0.0344
53  0.038030  0.31517 -0.16647  0.34355 1.165 3.95e-02 0.1317   *
54  0.025432 -0.01368 -0.01072  0.02976 1.087 3.00e-04 0.0328
55 -0.297768 -0.14408  0.46139  0.49343 1.111 8.03e-02 0.1288
56  0.003171 -0.03246  0.03242  0.05962 1.077 1.20e-03 0.0290
57  0.170509  0.01465 -0.14748  0.18095 1.082 1.10e-02 0.0547
58 -0.128062  0.16523 -0.01901 -0.22926 1.015 1.74e-02 0.0356
59 -0.074222  0.05679  0.01127 -0.10925 1.054 4.02e-03 0.0251
60  0.225126  0.27734 -0.48439 -0.55343 0.982 9.84e-02 0.0874
61 -0.144962 -0.12638  0.15843 -0.23754 1.011 1.86e-02 0.0361
62  0.218534 -0.15068 -0.07280  0.26981 0.987 2.39e-02 0.0355
```

11.3 Multicollinearity Indices

Variance Inflation factors are useful indices for assessing multicolinearity. In a two-IV model such as this one, these will not be very interesting unless the two IVs are extremely highly correlated. And for a two-IV model the VIFs will be the same for each IV, since there is only one other IV to be correlated with for each of them (and it is the same correlation). So, this illustration is not very informative beyond showing how to obtain the VIFs.

```
#library(car) #load the package car if not still loaded from above
vif <- vif(fit3) #variance inflation factors
vif
```

```
   pubs    cits
1.12505 1.12505
```

```
# one rule of thumb is to look for square root of vif values  > 2
vif(fit3)^.5
```

```
    pubs     cits
1.060684 1.060684
```

```
vif(fit3)^.5 > 2
```

```
 pubs  cits
FALSE FALSE
```

```
olsrr::ols_coll_diag(fit3)
```

```
Tolerance and Variance Inflation Factor
---------------------------------------
  Variables Tolerance     VIF
1      pubs 0.8888492 1.12505
2      cits 0.8888492 1.12505


Eigenvalue and Condition Index
------------------------------
  Eigenvalue Condition Index  intercept       pubs       cits
1 2.68529631         1.00000 0.01850054 0.03808100 0.01741131
2 0.23613502         3.37222 0.11732112 0.95108275 0.07094028
3 0.07856867         5.84617 0.86417834 0.01083626 0.91164841
```

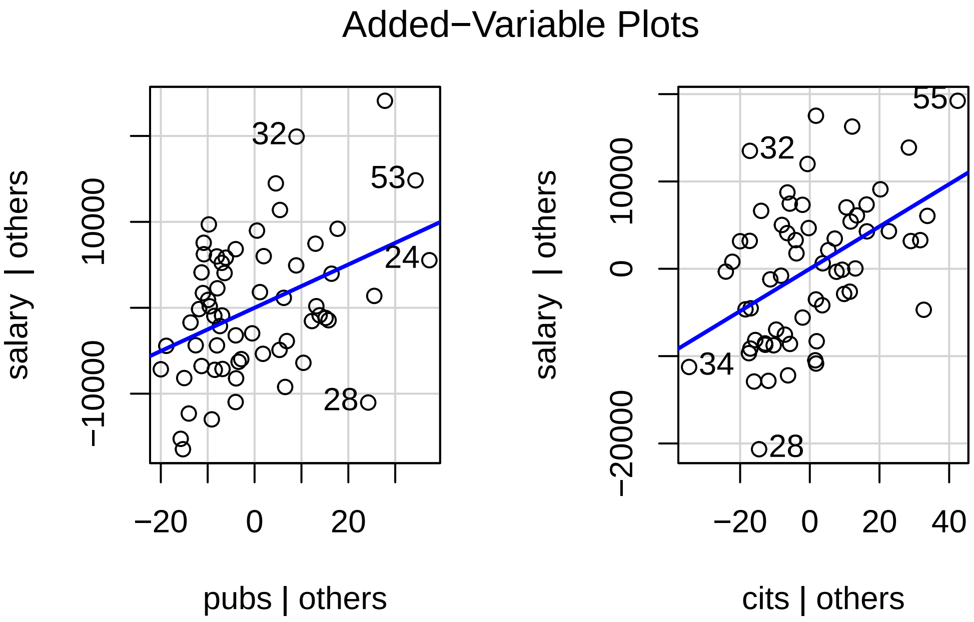
11.4 Plots for model criticism

Several of the influence statistics described above are incorporated into easily produced visualizations. Other examples are in later chapters (e.g. the chapter on **OLSRR**).

11.4.1 Added Variable Plots

Added variable plots are a tool that permits examination of the partial correlations between pairs of variables. Each of the two variables in a plot has been residualized on all other variables. This was also seen in chapter 4.

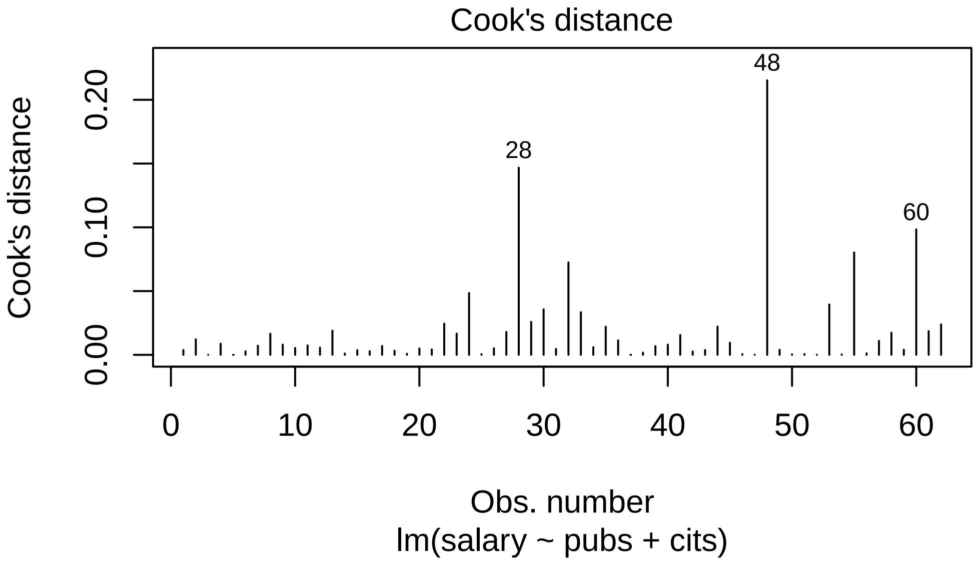
```
avPlots(fit3) #added variable plots
```



11.4.2 A Cooks D plot

This code chunk creates a plot of Cooks D values (an influence statistic) as a function of sequential case number in the data set. It also identifies extreme cases (case numbers labeled) based on a threshold set in the threshold object. As discussed in class, it is best to look for Cook's D values divergent from others rather than those exceeding a numerical criterion, but some criteria have been proposed and one of those defines the threshold here.

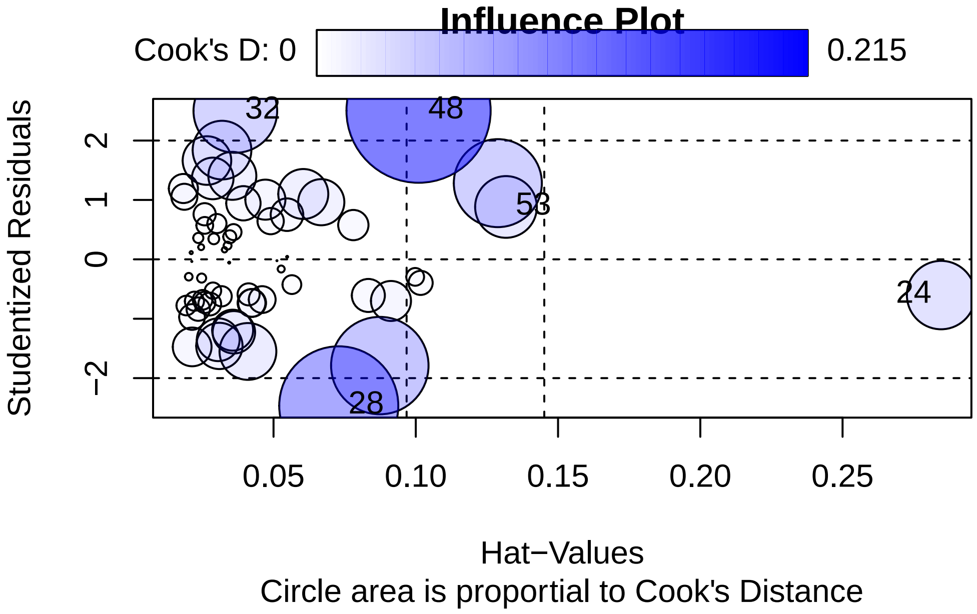
```
# Cook's D plot
# identify D values > 4/(n-k-1)
threshold  <- 4/((nrow(cohen1)-length(fit3$coefficients)-2))
plot(fit3, which=4, cook.levels=threshold)
```


Cook's distance
Cook's distance
0.00
0.10
0.20
0
10
20
30
40
50
60
28
48
60
Obs. number
lm(salary ~ pubs + cits)

11.4.3 A standard Influence Plot from the car package

This influence plot is a standard scatterplot of residuals against yhats, and is the plot used to visually assess heteroscedasticity. But the added benefit here is that the size (area) of each point is scaled to the Cook's D statistic thus permitting a visualization of the most influential cases in their residual/yhat position.

```
# Influence Plot
car::influencePlot(fit3, id=TRUE,
              main="Influence Plot", sub="Circle area is proportial to Cook's Distance" )
```



```
      StudRes        Hat      CookD
24 -0.6018576 0.28463388 0.04856737
28 -2.4663093 0.07292105 0.14683221
32  2.4982801 0.03657164 0.07253089
48  2.5025997 0.10097428 0.21527380
53  0.8821127 0.13170293 0.03949029
```

11.5 Additional functions for examining outliers, normality, and leverage

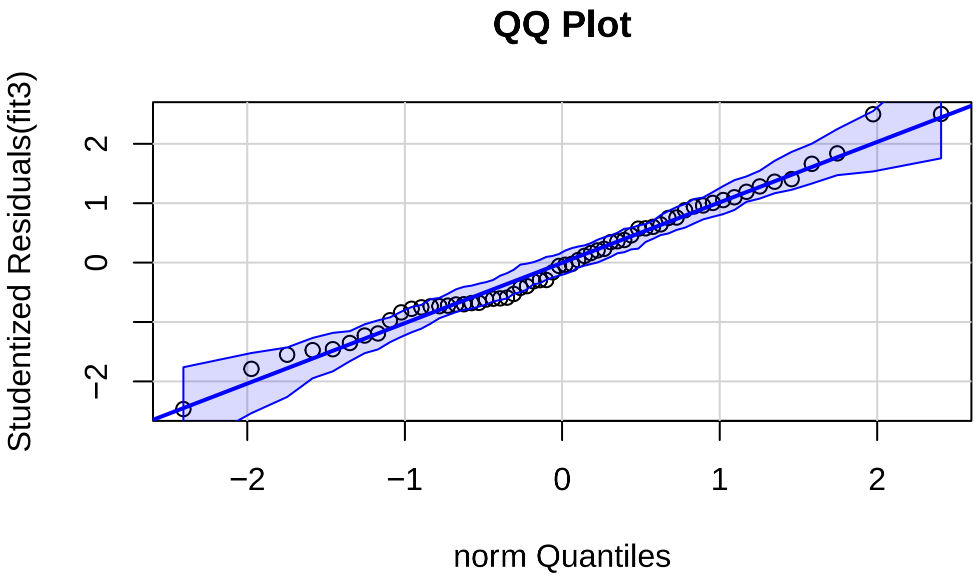
Fox' **car** package has a useful function that identifies outliers and evaluates their likelihood of occurrence with a p value. But doing this for N cases creates a multiple testing problem, so a Bonferroni adjusted p value is also presented. In this data set no extreme outliers were detected. This `outlierTest` function is evaluating outlier status relative to th studentized residuals.

```
#library(car) # if not loaded previously
outlierTest(fit3) # Bonferonni p-value for most extreme obs\
```

```
No Studentized residuals with Bonferroni p < 0.05
Largest |rstudent|:
   rstudent unadjusted p-value Bonferroni p
48   2.5026           0.015169      0.94046
```

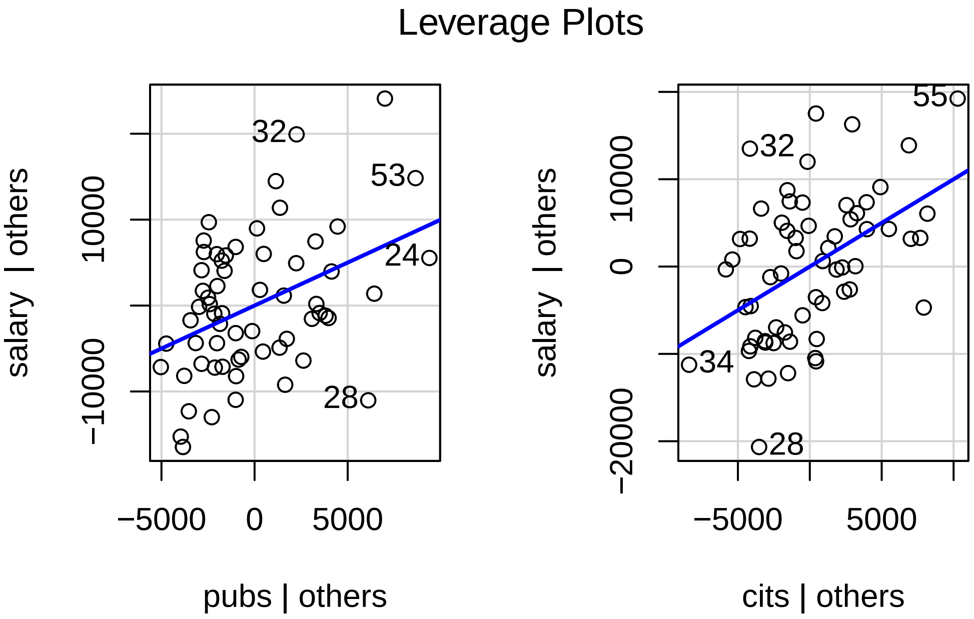
We have already used the qq normal plot of residuals in many places. We should not forget that it is a diagnostic tool and thus properly included in this chapter.

```
qqPlot(fit3, main="QQ Plot", distribution="norm", id=F) #qq normal plot for studentized resi
```



The leverage plots shown here appear to be identical to the added variable plots shown above, and they are. The illustration here is done as a placeholder for later analyses where some IVs (such as categorical IVs with more than two levels) would have more than 1 df. In that case, this plot is useful, in addition to added variable plots.

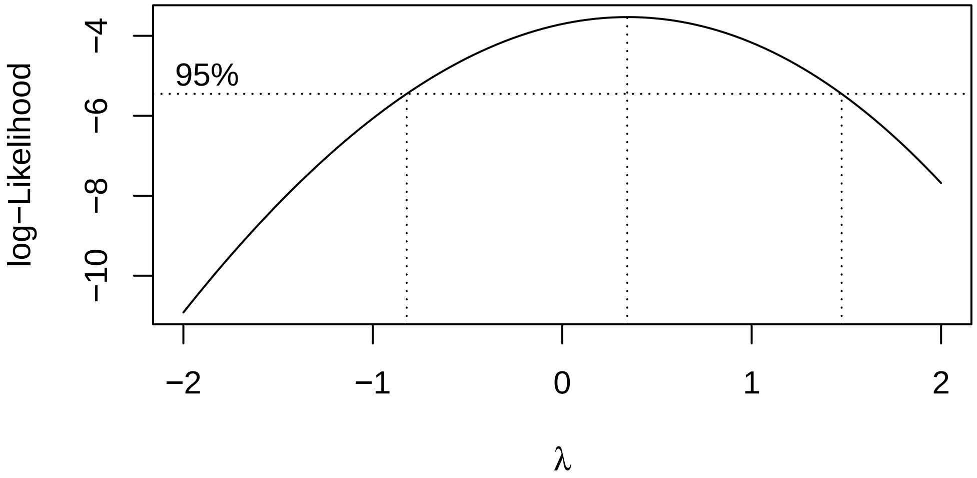
```
# library(car)  # the function is from the car package
leveragePlots(fit3) # leverage plots
```



11.6 Scale transformations when assumptions are violated

A method of choice for addressing any violations of assumptions is often simple scale transformation of either the DV or the IVs, or both. We covered this idea with the household water use data set. We will revisit the scale transformation topic again at a later point in the course and will introduce Tukey's ladder of scales and Box-Cox transformation. Here a quick implementation of Box-Cox transformation capabilities is introduced. In this instance, the goal is to find an exponent to use to transform the DV to help produce normality and homoscedasticity of the residuals. The peak of the curve in the plot yields λ, the scaling component.

```
# scale transformations of the DV and/or IVs can help alleviate the impact of many issues
#  #evaluate possible Box-Cox transformations (MASS package must be installed)
# look for the lambda at the log likelihood peak
#library(MASS)
boxcoxfit3 <- boxcox(fit3,plotit=T)
```



```
#boxcoxfit3
# also see `boxCox` from **car**
```

To find the exact λ value the output from the `boxcox` function was placed in a data frame and then sorted according to the log-likelihood value. The first case in the resorted data frame gives the correct λ value. Note that the X and Y values are those used to create the plot above.

```
cox = data.frame(boxcoxfit3$x, boxcoxfit3$y)            # Create a data frame with the resul
#str(cox)
cox2 <- cox[with(cox, order(-cox$boxcoxfit3.y)),] # Order the new data frame by decreasing y
#str(cox2)
cox2[1,] # Display the lambda with the greatest log likelihood
```

```
   boxcoxfit3.x boxcoxfit3.y
59    0.3434343    -3.531357
```

For this model, λ is was found to be about .343. This value would be used to transform the DV and the analysis rerun. But recall that the normality and homoscedasticity assumptions didn't appear to be violated so this process would be unnecessary for the current data set. I show the code just to provide a template. Notice that an exponent of 1.0 would not be a scale change. The exponent of .343 is approximately equivalent to the cubed root. Also note that with the scale change, the values of the regression coefficients would become less readily interpretable.

```
#tsalary <- salary**(cox2[1,]$boxcoxfit3.x)
#hist(tsalary)
#fit3t <- lm(tsalary~cohen1$pubs+cohen1$cits)
#summary(fit3t)
#qqPlot(fit3$residuals, distribution="norm")
```

12 Resampling and Robust methods for Linear Models

Two classes of alternative approaches to linear modeling are briefly described in this section. The first is bootstrapping which is an alternative to std error calculation when faced with non-normal residuals. Although there are many "flavors" of bootstrapping, only two are illustrated here. The second class is a broad range of methods falling under the rubric of Robust Regression. The ones emphasized here are those that can help when heteroscedasticity is present in an OLS model. The code is not accompanied by much commentary or explanation. It is presumed that background on bootstrapping methods is obtained elsewhere.

12.1 Ordinary nonparametric bootstrapping with the `boot` function

The **boot** package has extensive facilities for bootstrapping many kinds of analyses. The essential logic is to tell the `boot` function which statistic from the analysis needs to be bootstrapped, what kind of bootstrapping (e.g., casewise vs residual), and then requests for summaries of the bootstrapping. This first section is based somewhat on the methods outlined in the Quick-R website. The methods presented here are nonparametric bootstrapping methods.

Each statistic of interest can be bootstrapped. But this requires writing a function to extract that statistic from the `lm` fit so that it can be repeated with the bootstrap samples. This section will do bootstrap analyses on three statistics:

1. The Multiple R-Squared from the two-IV model used throughout this doc.
2. The pvalue for the F test of that Multiple R-squared (test of the "whole" equation).
3. The regression coefficients from the two-IV equation.

In these illustrations, I ran the `boot` function (creating the results object) within a call to the `system.time` function to record how long it took to do the bootstrap re-sampling. This will vary across platforms.

12.1.1 Bootstrap the multiple R-squared statistic

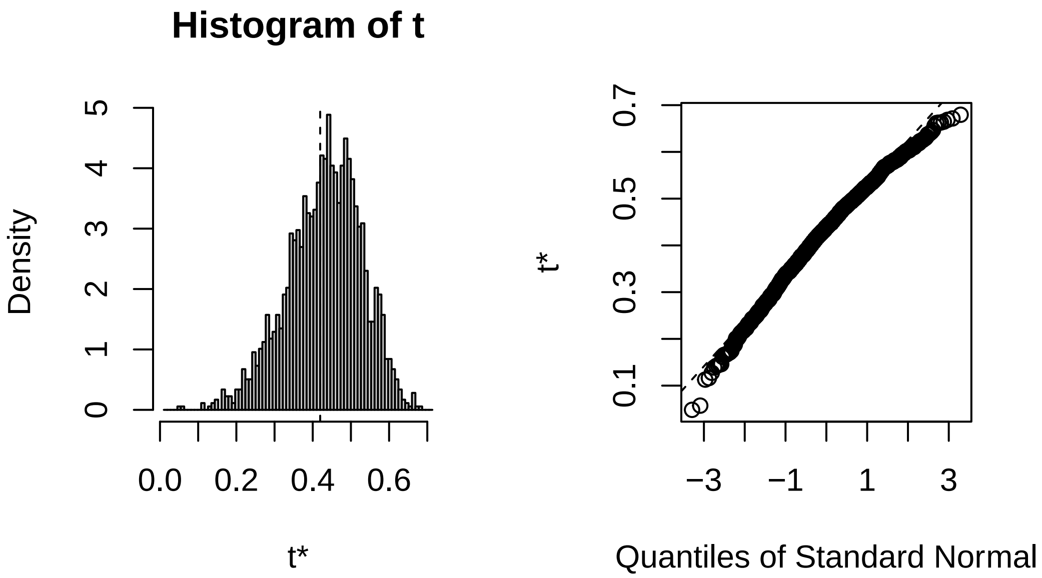
```
#library(boot)
# Bootstrap 95% CI for R-Squared
# function to obtain R-Squared from the data
multrsq <- function(formula, data, indices) {
     d <- data[indices,] # allows boot to select sample
     fit <- lm(formula, data=d)
     return(summary(fit)$r.square)
}
# bootstrapping with 2000 replications
# for other kinds of statistics, `boot` may need a "maxit" argument to  set the upper limit
set.seed(1234) # for reproducibility within this document
system.time(results1 <- boot(data=cohen1, statistic=multrsq,
   R=2000, formula=salary~pubs+cits))
```

```
   user  system elapsed 
   0.64    0.00    0.66 
```

```
# view results
results1
```

```


ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = cohen1, statistic = multrsq, R = 2000, formula = salary ~ 
    pubs + cits)


Bootstrap Statistics :
     original    bias    std. error
t1* 0.4195157 0.0118493  0.09665115
```

```
plot(results1)
```



```
# show 95% confidence interval
boot.ci(results1, type=c("norm","basic","perc","bca"))
```

```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 2000 bootstrap replicates

CALL :
boot.ci(boot.out = results1, type = c("norm", "basic", "perc",
    "bca"))

Intervals :
Level      Normal              Basic
95%   ( 0.2182,  0.5971 )   ( 0.2378,  0.6161 )

Level     Percentile            BCa
95%   ( 0.2230,  0.6013 )   ( 0.1806,  0.5768 )
Calculations and Intervals on Original Scale
```

12.1.2 Bootstrap the omnibus F test p value

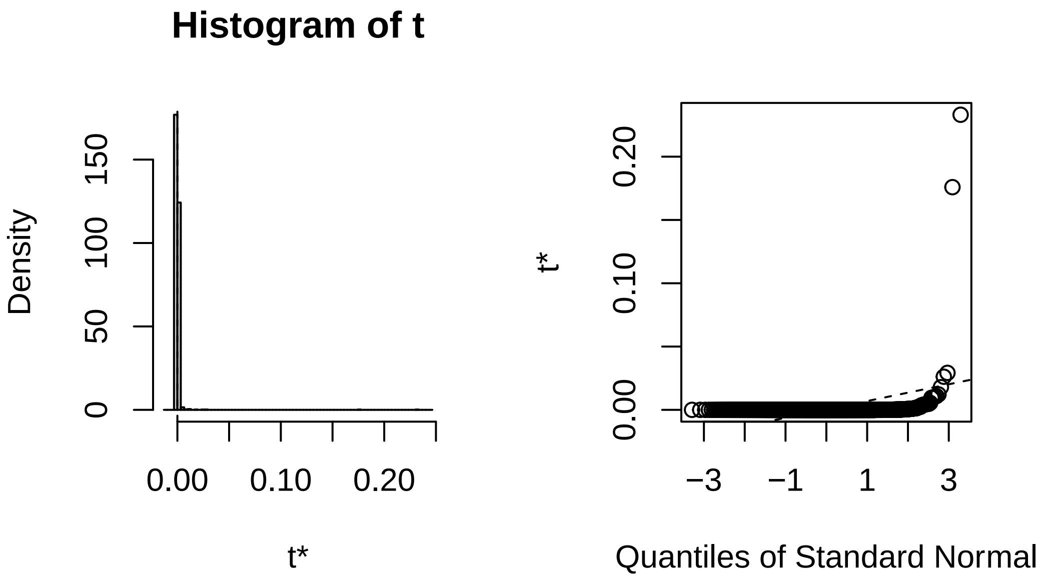
```
fpval <- function(formula, data, indices) {
     d <- data[indices,] # allows boot to select sample
     fit <- lm(formula, data=d)
     return(pf(summary(fit)$fstatistic[1],
               df1=summary(fit)$fstatistic[2],
               df2=summary(fit)$fstatistic[3],lower.tail=F))
}
# bootstrapping with 2000 replications
# for other kinds of statistics, `boot` may need a "maxit" argument to  set the upper limit
set.seed(1234) # for reproducibility within this document
system.time(results2 <- boot(data=cohen1, statistic=fpval,
   R=2000, formula=salary~pubs+cits))
```

```
   user  system elapsed
   0.75    0.02    0.77
```

```
# view results
results2
```

```


ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = cohen1, statistic = fpval, R = 2000, formula = salary ~
    pubs + cits)


Bootstrap Statistics :
        original        bias    std. error
t1* 1.076015e-07 0.0003175022 0.006625147
```

```
plot(results2)
```



```
# show 95% confidence interval
boot.ci(results2, type=c("bca"))
```

```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 2000 bootstrap replicates

CALL :
boot.ci(boot.out = results2, type = c("bca"))

Intervals :
Level       BCa
95%   ( 0.0000,  0.0041 )
Calculations and Intervals on Original Scale
```

12.1.3 Bootstrap the regression coefficients

Next, we will bootstrap the regression coefficients

```
# Bootstrap 95% CI for regression coefficients
#library(boot)
# function to obtain regression weights
```

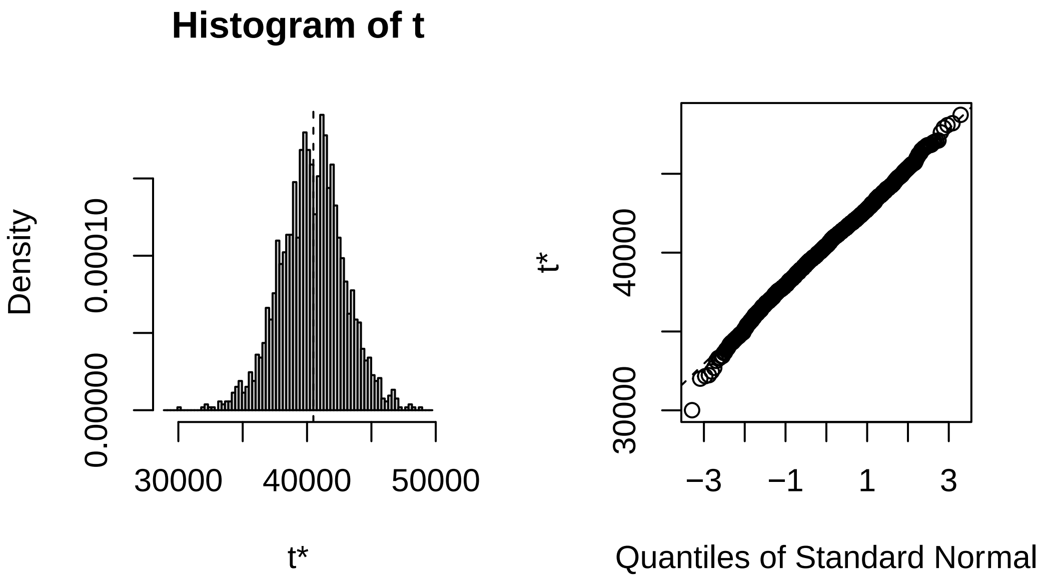
```r
coeffs <- function(formula, data, indices) {
  d <- data[indices,] # allows boot to select sample
  fit <- lm(formula, data=d)
  return(coef(fit))
}
# bootstrapping with 2000 replications
# for other kinds of statistics, `boot` may need a "maxit" argument to  set the upper limit
set.seed(1234) # for reproducibility within this document
system.time(results3 <- boot(data=cohen1, statistic=coeffs,
        R=2000, formula=salary~pubs+cits))
```

```
   user  system elapsed 
   0.53    0.00    0.54 
```

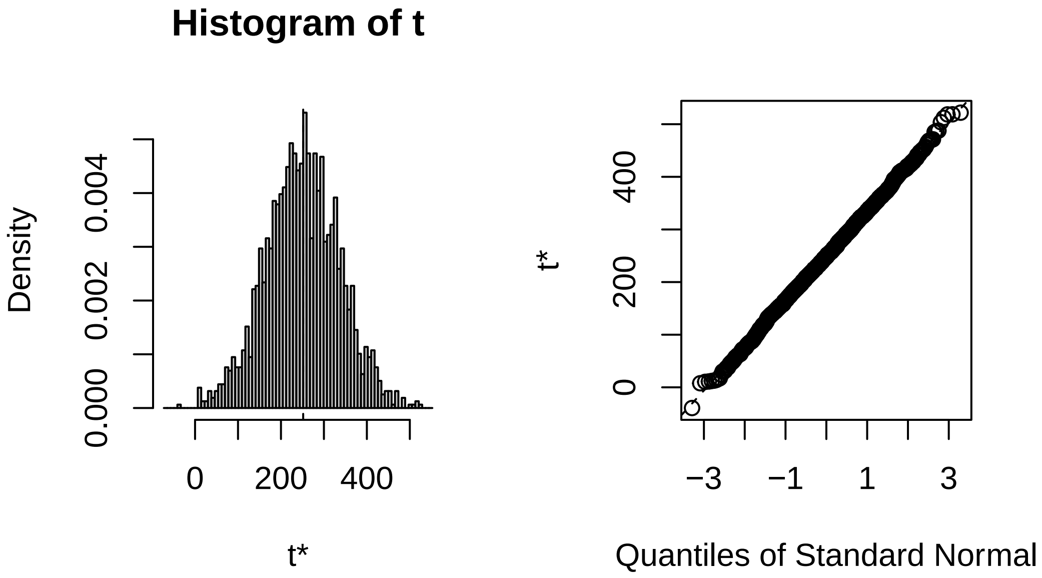
```r
# view results
results3
```

```


ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = cohen1, statistic = coeffs, R = 2000, formula = salary ~ 
    pubs + cits)


Bootstrap Statistics :
      original      bias    std. error
t1* 40492.9715 -92.356087  2481.39738
t2*   251.7500  -2.286656    85.13455
t3*   242.2977   2.522431    58.03025
```

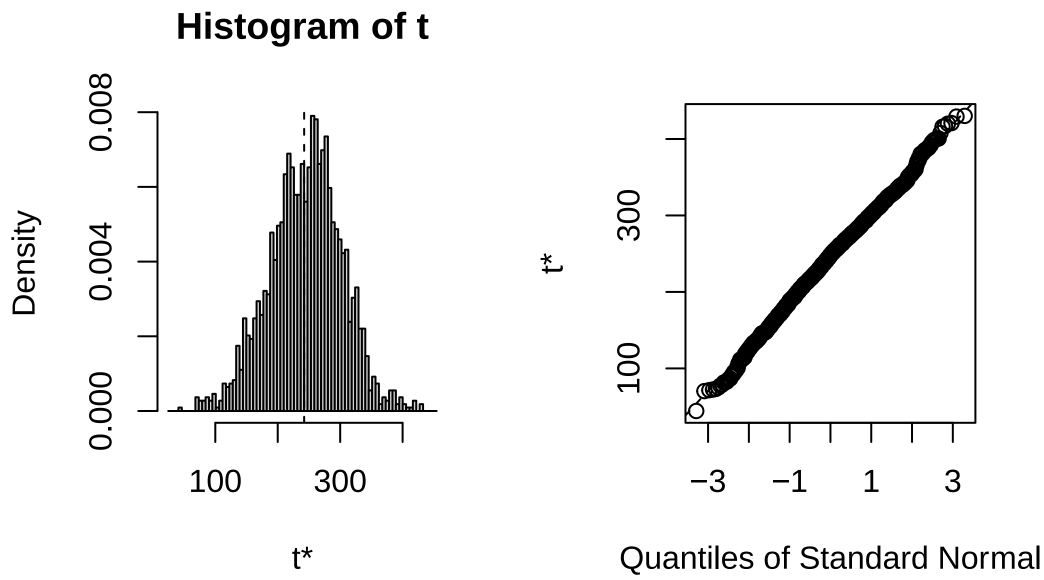
```r
plot(results3, index=1) # intercept
```



```r
plot(results3, index=2) # wt
```



```
plot(results3, index=3) # disp
```



```
# get 95% confidence intervals
boot.ci(results3, type="bca", index=1) # intercept
```

```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 2000 bootstrap replicates

CALL :
boot.ci(boot.out = results3, type = "bca", index = 1)

Intervals :
Level       BCa
95%   (35539, 45437 )
Calculations and Intervals on Original Scale
```

```
boot.ci(results3, type="bca", index=2) # pubs
```

```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 2000 bootstrap replicates
```

```
CALL :
boot.ci(boot.out = results3, type = "bca", index = 2)

Intervals :
Level       BCa
95%   ( 86.9, 424.7 )
Calculations and Intervals on Original Scale
```

```
boot.ci(results3, type="bca", index=3) # cits
```

```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 2000 bootstrap replicates

CALL :
boot.ci(boot.out = results3, type = "bca", index = 3)

Intervals :
Level       BCa
95%   (112.6, 340.1 )
Calculations and Intervals on Original Scale
```

12.2 An alternative bootstrap approach from car package functions

John Fox has produce the **car** package that we have seen contains many useful functions for linear modeling (Fox, 2016). One function, `Boot`, provides a flexible approach to non-parametric bootstrapping that may be easier to use. I also find that the `Confint` function is a quicker way to get a full picture of bootstrapped confidence intervals.

```
# Bootstrap 95% CI for regression coefficients
#library(boot)
# function to obtain regression weights
coeffs <- function(formula, data, indices) {
  d <- data[indices,] # allows boot to select sample
  fit <- lm(formula, data=d)
  return(coef(fit))
}
# bootstrapping with 2000 replications
# for other kinds of statistics, `boot` may need a "maxit" argument to  set the upper limit
set.seed(1234) # for reproducibility within this document
system.time(results4 <- car::Boot(fit3, f=coef,
        R=2000, method=c("case"))) # can specify casewise or residual sampling with the meth
```

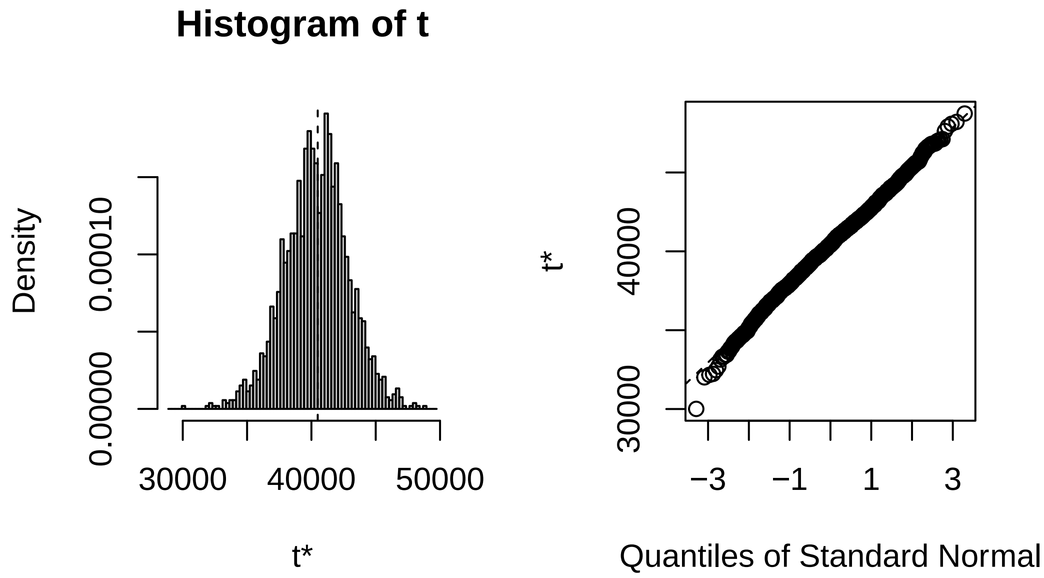
```
   user  system elapsed 
   0.72    0.00    0.74 
```

```
# view results
summary(results4, high.moments=T)
```

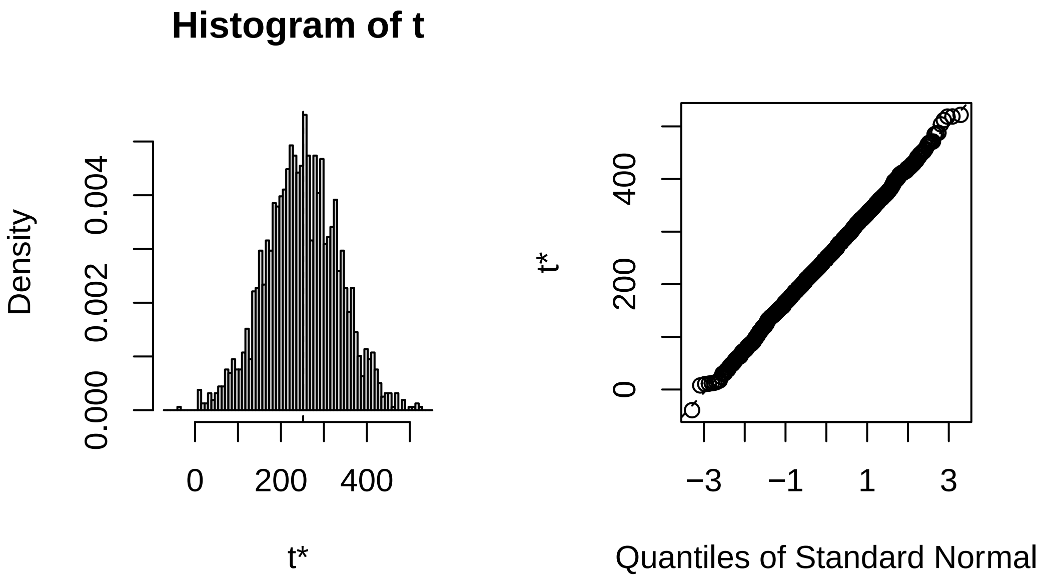
```
Number of bootstrap replications R = 2000 
            original bootBias   bootSE  bootMed   bootSkew bootKurtosis
(Intercept) 40492.97 -92.3561 2481.397 40430.85 -0.1058512     0.314662
pubs          251.75  -2.2867   85.135   248.05  0.0016417     0.013766
cits          242.30   2.5224   58.030   248.02 -0.1261283     0.134424
```

Bootsrapped empirical distributions for each coefficient are produced with the application of the base system `plot` function to the bootstrapped results.

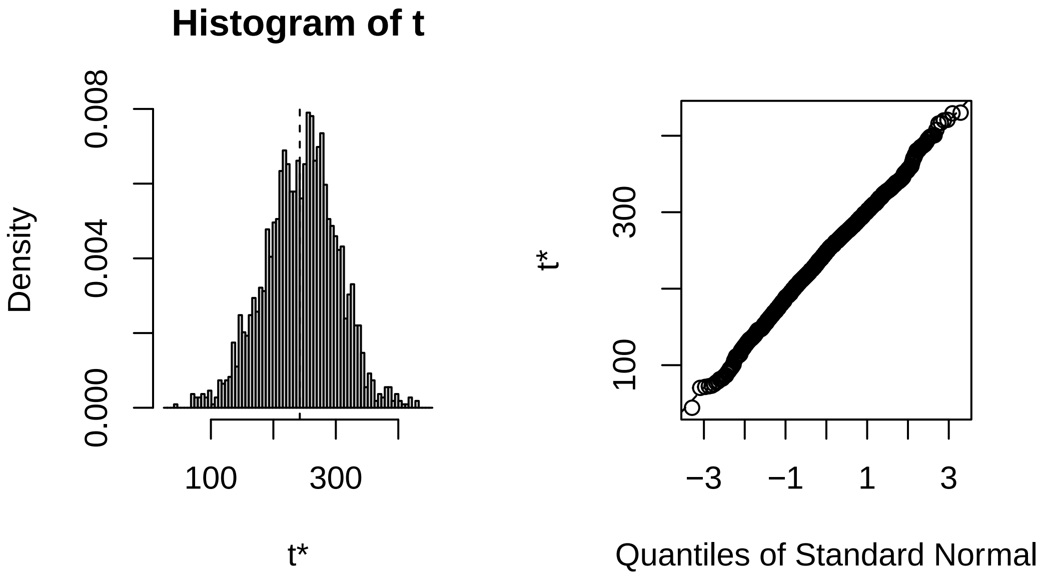
```
plot(results4, index=1) # intercept
```



```
plot(results4, index=2) # wt
```



```r
plot(results4, index=3) # disp
```



And confidence intervals of multiple types are readily produced. Percentile (“perc”) and “bca” are typically recommended.

```r
# get 95% confidence intervals
Confint(results4, type="norm")
```

```
Bootstrap normal confidence intervals

               Estimate       2.5 %     97.5 %
(Intercept) 40492.9715 35721.87804 45448.7770
pubs          251.7500    87.17604   420.8973
cits          242.2977   126.03808   353.5125
```

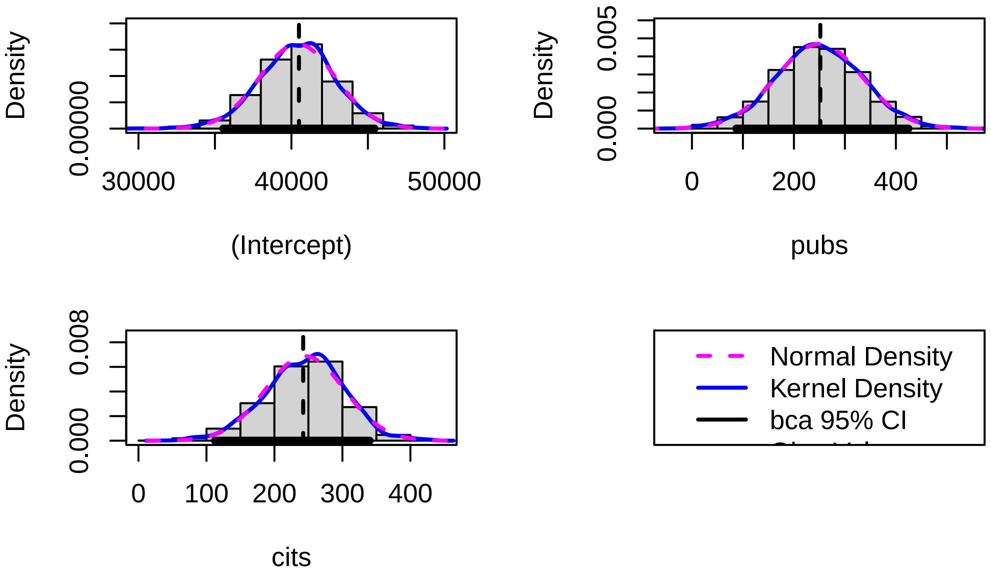
```r
Confint(results4, type="perc")
```

```
Bootstrap percent confidence intervals

               Estimate       2.5 %     97.5 %
(Intercept) 40492.9715 35280.22022 45274.3103
pubs          251.7500    77.56269   415.7893
cits          242.2977   127.95169   353.4045
```

```r
Confint(results4, type="bca")
```

```
Bootstrap bca confidence intervals

               Estimate       2.5 %     97.5 %
(Intercept) 40492.9715 35539.48858 45436.7048
pubs          251.7500    86.91784   424.6978
cits          242.2977   112.56818   340.1057
```

```r
# obtain a useful set of histograms for each coefficient distribution
hist(results4, legend="separate")
```



```
vcov(results4)
```

```
            (Intercept)        pubs        cits
(Intercept)  6157332.98 -47605.756 -112384.905
pubs          -47605.76   7247.891   -1593.342
cits         -112384.90  -1593.342    3367.510
```

12.3 Robust Regression for heteroscedastic data sets

Bootstrapping methods are primarily employed as a tool to handle situations where normality assumptions are violated. They are not necessarily insensitive to heteroscedasticity issues. In this section, I outline tools available from the **car**, **lmtest**, **MASS**, and **sandwich** packages for robust regression analyses. Also included is a brief introduction to a suite of robust methods for handling data sets with outliers and influential cases.

12.3.1 A robust Standard Errors approach from the car Package when heteroscedasticity is present.

The Salary-pubs-cits data set was not one that showed much heteroscedasticity or non-normality, so submitting it to robust methods may not be necessary. Another data set we have see previously does have heteroscedasticity and this is what is used here. This is a data set provided by Dr. James Boswell on baseline scores of anxiety-diagnosed patients on scales of depression severity (bods, treated as the DV), anxiety severity (boasev), and positive affect (bpa). A standard two-IV linear model is fit and then corrected. The data file is available baselinevars.csv

```
boswell <- read.csv("data/baselinevars.csv")
```

The two-iv regression produced a multiple R-squared of about .42 and the standard tests of the two IVs are shown here. Note that the data set has a fairly large N (200) and thus substantial power for effects of the size seen.

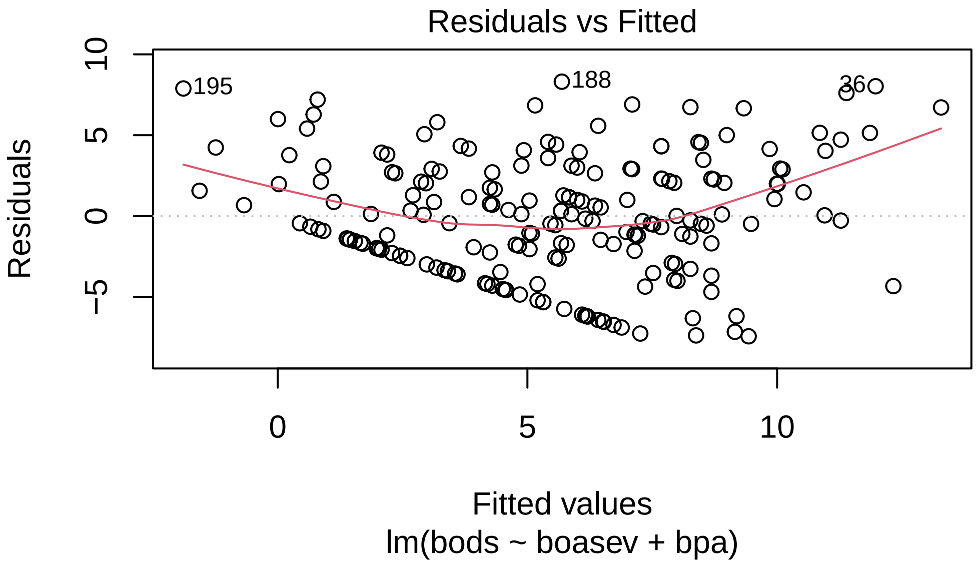
```
fit10<- lm(bods~boasev+bpa, data=boswell)
kable(tidy(fit10), caption=
           "Regression Coefficients and Regular Parametric Std Errors")
```

Table 12.1: Regression Coefficients and Regular Parametric Std Errors

term	estimate	std.error	statistic	p.value
(Intercept)	5.7363514	1.6676431	3.439796	0.0007105
boasev	0.5829990	0.0775505	7.517665	0.0000000
bpa	-0.2104816	0.0405446	-5.191359	0.0000005

The plot of residuals vs yhats reveals an odd pattern, that reflects a truncated variable (boasev). But it also reveals a modest amount of heteroscedasticity.

```
plot(fit10, which=1)
```



```
#qqPlot(fit10$residuals, distribution="norm", id=F)
```

Testing a null hypothesis of no heteroscedasticity with the Breusch-Pagan test finds a p value in the rejection range.

```
#library(lmtest)
bptest(fit10)
```

```

	studentized Breusch-Pagan test

data:  fit10
BP = 16.966, df = 2, p-value = 0.0002069
```

Several different methods of estimating corrected covariance matrices based on heteroscedasticity are available. The one illustrated here (“HC0”) is often simply called the White Std Error. See the help doc on `hccm` for more details. The `hccm` function is from the **car** package and `coeftest` is from the **lmtest** package. You can see that the standard errors are different than the regular standard errors, but they are not much larger owing to the fact that heteroscedasticity is not severe in this data set.

```
cov <- hccm(fit10, type="hc0") #needs package 'car'
fit10.HC0 <- coeftest(fit10, vcov.=cov)
kable(tidy(fit10.HC0), caption=
        "Robust (HC0) standard errors and modified t-tests")
```

Table 12.2: Robust (HC0) standard errors and modified t-tests

term	estimate	std.error	statistic	p.value
(Intercept)	5.7363514	1.9687184	2.913749	0.0039840
boasev	0.5829990	0.0984717	5.920470	0.0000000
bpa	-0.2104816	0.0454911	-4.626872	0.0000067

12.3.2 The Wald test and an alternative to adjusted t-tests for linear models with heteroscedasticity

The **lmtest** package has a function called `waldtest` that permits a commonly used inferential method, the Wald Test, which also employs covariance matrix correction for heteroscedasticity. It uses a model comparison strategy. First we created an intercept only model (mod0) and the the full two-IV model (modfull). `waldtest` then compares those two models in a standard model comparison methodology and the F test evaluates the improvement in fit, but using corrected standard errors of the White type (HC0). This F value is somewhat smaller than the one seen above for the regular OLS method, and is the alternate method for testing the null about the overall fit of the model. The `vcovHC` function comes from the **sandwich** package and is a common one for estimating the covariance structure in the presence of heteroscedasticity.

```
#library(lmtest)
#library(sandwich)
mod0 <- lm(bods~1 , data=boswell)
modfull <- lm(bods~boasev+bpa , data=boswell)
waldtest(mod0, modfull, vcov = vcovHC(modfull, type = "HC0"))
```

```
Wald test

Model 1: bods ~ 1
Model 2: bods ~ boasev + bpa
  Res.Df Df      F    Pr(>F)
1    199
2    197  2 57.232 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The **lmtest** package also has a function to test the individual regression coefficients with the adjusted standard errors. This approach is identical to the one shown above with the **car** package tools. The `vcovHC` function comes from the **lmtest** package. Adjusted standard errors and t's are the same as above since the same White Std. Errors are chosen.

```
#library("lmtest")
#library("sandwich")
# Robust t test
coeftest(modfull, vcov = vcovHC(modfull, type = "HC0"))
```

```
t test of coefficients:

             Estimate Std. Error t value  Pr(>|t|)    
(Intercept)  5.736351   1.968718  2.9137  0.003984 ** 
boasev       0.582999   0.098472  5.9205 1.406e-08 ***
bpa         -0.210482   0.045491 -4.6269 6.719e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

12.4 Robust regression methods when outliers or highly influential cases are present

When outliers or highly influential cases are detected via influence analysis it is possible to use that information to refine the linear model. The general idea is to take an influence statistic such as Cook's D and use it to weight the importance of cases in the regression. One commonly recommended method is called "Iteratively Reweighted Least Squares". The `rlm` function in the **MASS** package is a recommended method to perform such weighted/reweighted least squares. Rather than take additional space in this document, the reader is referred to a very good tutorial page on IRLS using `rlm`. This tutorial page is from the UCLA Statistical Consulting unit which has created many useful tutorials.

13 Extensions to larger models and to categorical IVs

In this chapter, the goals are to extend the suite of methods to models with additional characteristics. This includes a model with more than two IVs but we will also consider using a categorical IV toward the end. At the same time, this chapter provides some emphasis on strong capabilities in rmarkdown and the "tidyverse" to generate nicer tables. Some of that has been seen in other chapters with the `kable` and `gt` functions, but it is expanded here.

The capability of **quarto**, **rmarkdown**, **bookdown** and **knitr** to produce a document such as this one is accompanied by increased options for formatting the output. The typical text output from R functions can be converted to style/fonts that are more readable. A few brief examples are illustrated in this section. The kable function permits enhancement of tables such as those coming from `summary` and `anova`. But capabilities in the **broom** package also permit enhancement of the standard output coming from 'summary. The reader should recognize that whole documents/manuscripts can be written in Quarto/rmarkdown, even in APA style

13.1 Evaluate a 3-IV model

We will use the same data set that we used above with the enhanced output and formatting. However we will extend the analysis to three IVs. The reader should take note of the fact that adding a third variable is simply done in the model specification argument for the `lm` function and that such functions as `summary` and `anova` or `Anova` would be done the same way as previously executed:

```
fit6 <- lm(salary~pubs+cits+degree_yrs, data=cohen1)  # add degree_yrs as a 3rd IV to our ori
summary(fit6)
```

```

Call:
lm(formula = salary ~ pubs + cits + degree_yrs, data = cohen1)

Residuals:
     Min       1Q   Median       3Q      Max
```

```
-13907.1  -4313.8   -649.5   4366.1  21165.3 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 38967.85    2394.31  16.275  < 2e-16 ***
pubs           93.61      85.35   1.097 0.277273    
cits          204.06      56.97   3.582 0.000699 ***
degree_yrs    874.46     283.89   3.080 0.003160 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 7031 on 58 degrees of freedom
Multiple R-squared:  0.5011,	Adjusted R-squared:  0.4753 
F-statistic: 19.42 on 3 and 58 DF,  p-value: 7.727e-09
```

```r
mrinfo(fit6, minimal=TRUE)
```

```
[[1]]
NULL

$`supplemental information`
           beta wt structure r partial r semipartial r tolerances  unique
pubs       0.13506     0.71500   0.14254       0.10172    0.56721 0.01035
cits       0.36102     0.77662   0.42559       0.33219    0.84665 0.11035
degree_yrs 0.38541     0.85873   0.37495       0.28567    0.54940 0.08161
            common   total
pubs       0.24584 0.25618
cits       0.19190 0.30224
degree_yrs 0.28793 0.36954

$`var infl factors from HH:vif`
      pubs       cits degree_yrs 
  1.763010   1.181128   1.820156 

[[4]]
NULL
```

```r
Anova(fit6, type=3)
```

```
Anova Table (Type III tests)
```

```
Response: salary
                Sum Sq Df  F value    Pr(>F)    
(Intercept) 1.3093e+10  1 264.8822 < 2.2e-16 ***
pubs        5.9458e+07  1   1.2029 0.2772728    
cits        6.3412e+08  1  12.8291 0.0006989 ***
degree_yrs  4.6897e+08  1   9.4878 0.0031599 ** 
Residuals   2.8669e+09 58                       
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(fit3,fit6)
```

```
Analysis of Variance Table

Model 1: salary ~ pubs + cits
Model 2: salary ~ pubs + cits + degree_yrs
  Res.Df        RSS Df Sum of Sq      F  Pr(>F)   
1     59 3335822387                               
2     58 2866853951  1 468968436 9.4878 0.00316 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

It is also useful to examine the Variance Inflation Factors in multiple regression models. Unlike the two-IV model, each IV now has a different VIF since each has a different tolerance. Note that the IV with the smallest tolerance has the largest VIF.

```
vif(fit6)
```

```
      pubs       cits degree_yrs 
  1.763010   1.181128   1.820156 
```

Use the ‘tidy’ function from the ‘broom’ package to produce an alternative to the ‘summary’ table, and format it with ‘kable’:

```
fit6.tidy <- tidy(fit6) # tidy produces a nicer table than summary
kable(fit6.tidy) # kable converts the table to markdown format that is nicer than the defaul
```

term	estimate	std.error	statistic	p.value
(Intercept)	38967.84674	2394.30824	16.275201	0.0000000
pubs	93.60794	85.34837	1.096775	0.2772728
cits	204.06019	56.97179	3.581776	0.0006989
degree_yrs	874.46140	283.89492	3.080229	0.0031599

I actually prefer using the `gt` function for tables whenever I can. But sometimes `gt` doesn't play nice with rmarkdown when rendering to a pdf document. For example, using it in the code chunk here, with a tidy object, creates a problem. I show the code commented so that you can see how to use it in other formats. The same analysis is tabled here using `gt`.

```
#gt(fit6.tidy) # `gt` converts the table to markdown format that is nicer than the default
```

Use the `glance` function from the 'broom' package to obtain some additional information and then use 'kable' to format the output table. Note that I've relabeled the components of the `glance` object to fit better with our typically used notation schemes. I also broke the table into two components to better fit on the page. This latter strategy produced a printing of the pvalue that lost the exponential notation and I've not yet figured out how to prevent that. It appears to be an issue with `kable`, so I have printed the two tables without `kable` formatting.

```
#library(broom)
glancefit6a <- glance(fit6) # additional info on the model
glancefit6at1 <- glancefit6a[1:5]
#glancefit6at1
glancefit6at2 <- glancefit6a[6:11]
#glancefit6at2
kable(glancefit6at1,
      col.names=c("R.squared", "adj.R.squared", "RMSE","F value", "p.value"))
```

R.squared	adj.R.squared	RMSE	F value	p.value
0.5011234	0.4753195	7030.542	19.42041	0

```
kable(glancefit6at2,
      col.names=c("df regression","logLik", "AIC", "BIC", "deviance", "df residual"),digits=3
```

df regression	logLik	AIC	BIC	deviance	df residual
3	-635.104	1280.208	1290.844	2866853951	58

Notes USING GLANCE:

1. Users of the **broom** package should be aware that what `glance` previously reported as "df" would not have been the expected regression df that would equal the number of IVs. Instead it reported one more than that, including the df for the intercept. In the model above with 3 IVs, that apparently erroneous df was reported as 4, which is actually the rank of the design matrix. But the F test was always the standard MSreg/MSresidual, so it was confusing. That has been corrected in the recent versions as used here (ver 1.0.9) so that the 3 df seen in the table here is the correct DFregression for a three-IV model. Users who installed the package prior to the correction of this "feature" (ver 0.0.7) should update their installation.

This document is created with R version 4.5.2

2. The standard glance table is too wide - it has too many columns/values. So, I extracted two different subsets of its components and produced two different tables in the commented out code lines using `kable`.
3. I also relabeled the labels (in those commented out `kable` code lines) for the columns of the first table to be more in line with our standard terminology (RMSE is root mean square error, or the square root of MS residual)

Next, we use 'tidy' and 'kable' to obtain a nicely formatted anova summary table:

```r
tidyanova6 <- tidy(anova(fit6))
kable(tidyanova6,
      col.names=c("Source", "df", "SS", "MS", "F value", "p value"))
```

Source	df	SS	MS	F value	p value
pubs	1	1472195326	1472195326	29.784332	0.0000010
cits	1	938602110	938602110	18.989081	0.0000544
degree_yrs	1	468968436	468968436	9.487811	0.0031599
Residuals	58	2866853951	49428516	NA	NA

Or use the 'Anova' function:

```r
tidyAnova6 <- tidy(Anova(fit6, type=3))
kable(tidyAnova6,
      col.names=c("Source", "SS", "df", "F value", "p value"))
```

Source	SS	df	F value	p value
(Intercept)	13092731852	1	264.882153	0.0000000
pubs	59458298	1	1.202915	0.2772728

Source	SS	df	F value	p value
cits	634124345	1	12.829119	0.0006989
degree_yrs	468968436	1	9.487811	0.0031599
Residuals	2866853951	58	NA	NA

Now add information from the 'mrinfo' function that was discussed above, extracting the additional useful information that we need for a fuller analysis than summary/anova/Anova gives us.

```
mrinfo(fit6, minimal=TRUE)
```

```
[[1]]
NULL

$`supplemental information`
           beta wt structure r partial r semipartial r tolerances  unique
pubs       0.13506     0.71500   0.14254       0.10172    0.56721 0.01035
cits       0.36102     0.77662   0.42559       0.33219    0.84665 0.11035
degree_yrs 0.38541     0.85873   0.37495       0.28567    0.54940 0.08161
            common   total
pubs       0.24584 0.25618
cits       0.19190 0.30224
degree_yrs 0.28793 0.36954

$`var infl factors from HH:vif`
     pubs       cits degree_yrs
  1.763010   1.181128   1.820156

[[4]]
NULL
```

13.1.1 Conclusions from the three-IV model

Several interesting outcomes emerged with the addition of degree_yrs to the model that already included pubs and cits. The multiple R-squared did increase to about .50 so degree_yrs did appear to improve predictability. But an interesting thing happened to the effect of pubs in this three-IV model. The regression coefficient was about 251 in the two-IV model, but it dropped to about 94 in this three-IV model. Pubs was not a significant predictor in the three-IV model and it's unique fraction of variance dropped to about 1% of the DV variation. Most of the overlap of pubs with salary was shared by the other IV's (common fraction of variance was about .25, or nearly all of its zero order overlap with salary). This is because degree_yrs must have been strongly correlated with pubs. Lets examine that correlation.

```
with(cohen1, cor(pubs,degree_yrs))
```

```
[1] 0.6505472
```

A better model might be one that simply leaves pubs out since degree_yrs absorbed most of its shared variance.

```
fit6b <- lm(salary~cits+degree_yrs,data=cohen1)
fit6b.tidy <- tidy(fit6b) # tidy produces a nicer table than summary
kable(fit6b.tidy)
```

term	estimate	std.error	statistic	p.value
(Intercept)	39073.6747	2396.47360	16.304655	0.0000000
cits	212.1116	56.59392	3.747958	0.0004078
degree_yrs	1061.7642	227.17563	4.673759	0.0000176

```
mrinfo(fit6b, minimal=TRUE)
```

```
[[1]]
NULL

$`supplemental information`
           beta wt structure r partial r semipartial r tolerances  unique
cits       0.37526     0.78476   0.43852        0.3482    0.86094 0.12124
degree_yrs 0.46796     0.86773   0.51981        0.4342    0.86094 0.18853
           common   total
cits        0.181 0.30224
degree_yrs  0.181 0.36954

$`var infl factors from HH:vif`
      cits degree_yrs
  1.161517   1.161517

[[4]]
NULL
```

```
glancefit6b <- glance(fit6b)
glancefit6bt1 <- glancefit6b[1:5]
#glancefit6bt1
glancefit6bt2 <- glancefit6b[6:11]
#glancefit6bt2
kable(glancefit6bt1,
      col.names=c("R.squared", "adj.R.squared", "RMSE","F value", "p.value"))
```

R.squared	adj.R.squared	RMSE	F value	p.value
0.4907768	0.473515	7042.621	28.43137	0

```
kable(glancefit6bt2,
      col.names=c("df regression","logLik", "AIC", "BIC", "deviance", "df residual"),digits=
```

df regression	logLik	AIC	BIC	deviance	df residual
2	-635.74	1279.481	1287.989	2926312249	59

The revised two-IV model with degree_yrs and cits as IVs had an R-squared nearly as large as the three-IV model, and the AIC/BIC indices were slightly smaller indicating a better model. Both degree_yrs and cits were significant IVs as tested by the t-tests. Thus the most parsimonious model might be this revised two-IV model. Perhaps pubs was only a good IV because it was correlated so highly with degree_yrs and it is the latter that is the more important predictor.

13.2 Categorical IVs

We can do a quick preview of using categorical IVs in linear models. The cohen data set also had a categorical variable, gender, that could potentially serve as an IV. However, it is a factor and its values in the data set are the string values of “female” and “male”. How could that be used as an IV in regression? The answer is that factors can be recoded to have numeric values with specially constructed coded values. For a factor such as gender with only two levels the code is simply a one and a zero. This is already built in to the characteristics of the factor that is stored in the data frame. These numeric codes are called contrasts in R:

```
contrasts(cohen1$gender)
```

```
       male
female    0
male      1
```

With that knowledge, we now see that `lm` can handle such categorical variables by using the numeric contrast code. First I will illustrate with a simple regression using only gender as the IV.

```
fit7 <- lm(salary~gender, data=cohen1)
summary(fit7)
```

```

Call:
lm(formula = salary ~ gender, data = cohen1)

Residuals:
     Min       1Q   Median       3Q      Max 
-18660.3  -5819.8     -3.7   4769.2  26903.7 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)    52650       1810  29.080   <2e-16 ***
gendermale      3949       2445   1.615    0.111    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9580 on 60 degrees of freedom
Multiple R-squared:  0.04168,	Adjusted R-squared:  0.0257 
F-statistic: 2.609 on 1 and 60 DF,  p-value: 0.1115
```

It appears that gender is not a strong predictor of salary (R-squared is small, and t-test is NS). But lets focus on two additional points. One is notation. Observe that the name of the gender variable in the coefficients table is not simply gender, the IV name. Rather it is found there as "gendermale". That notation reminds us that gender is a factor and the reason that it is called gendermale rather than genderfemale is that male was coded with a 1 as seen above. The zero and one assignments are arbitrary and could be reversed. We will soon extend this conversation with more complex categorical IVs that have more than two categories.

The second point of emphasis is the thinking that surrounds the question of what does it mean to think of gender as "correlated" with salary, or that it can "predict" salary? Perhaps a better phrasing might be that we have assessed whether gender is "associated" with salary. We have concluded that it is not, but what if it were? That could only mean that average salaries might differ between females and males. And that sounds just like a two-independent-samples t-test situation. So let's do that test.

```
t.test(salary~gender, var.equal=T, data=cohen1)
```

```

	Two Sample t-test

data:  salary by gender
t = -1.6153, df = 60, p-value = 0.1115
alternative hypothesis: true difference in means between group female and group male is not e
95 percent confidence interval:
 -8839.888   941.241
sample estimates:
mean in group female   mean in group male 
            52650.00             56599.32 
```

Note that the male mean is about $4000 higher than the female mean. But the difference was not significant. A bit of a closer examination of the output reveals interesting coincidences. Note that the df for the t, the t-value itself, and the p value match what was reported for the regression coefficient in the above `lm` fit. This is not coincidence because the "association" question approached with regression is tantamount to the mean difference question posed by the independent samples t-test. Soon enough, we will progress through the technical reasons that these are the same inference.

For one final illustration in this chapter, we now build a regression model that adds gender to the earlier model that had degree_yrs and citations as IVs.

```
fit7b <- lm(salary~degree_yrs + cits + gender, data=cohen1)
summary(fit7b)
```

```

Call:
lm(formula = salary ~ degree_yrs + cits + gender, data = cohen1)

Residuals:
     Min       1Q   Median       3Q      Max 
-14214.6  -4590.3   -312.3   4014.9  21917.9 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
```

```
(Intercept) 38787.21    2478.78  15.648  < 2e-16 ***
degree_yrs   1040.41     232.58   4.473 3.64e-05 ***
cits          210.14      57.09   3.681 0.000512 ***
gendermale    931.06    1859.70   0.501 0.618513
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 7088 on 58 degrees of freedom
Multiple R-squared:  0.493, Adjusted R-squared:  0.4667
F-statistic:  18.8 on 3 and 58 DF,  p-value: 1.227e-08
```

In this model gender is also not a significant IV and the regression coefficient is smaller than it was in simple regression indicating that any variance that it shared with salary was at least partially confounded with degree_yrs and cits.

One final comment about this type of modeling with a variable such as gender. It might be an interesting research question whether the predictability that IVs such as degree_yrs and cits have on salary differs in the two levels of a categorical variable such as gender. In other words, does the influence of degree_yrs and cits depend on whether we examine males or females? This is what we will come to call an interaction question. It can be modeled easily with `lm`. I will show the code/results for the three-IV model with all interactions as well as the results, but will save comment until we cover the interaction topic more explicitly.

```
fit7c <- lm(salary~degree_yrs*cits*gender, data=cohen1)
summary(fit7c)
```

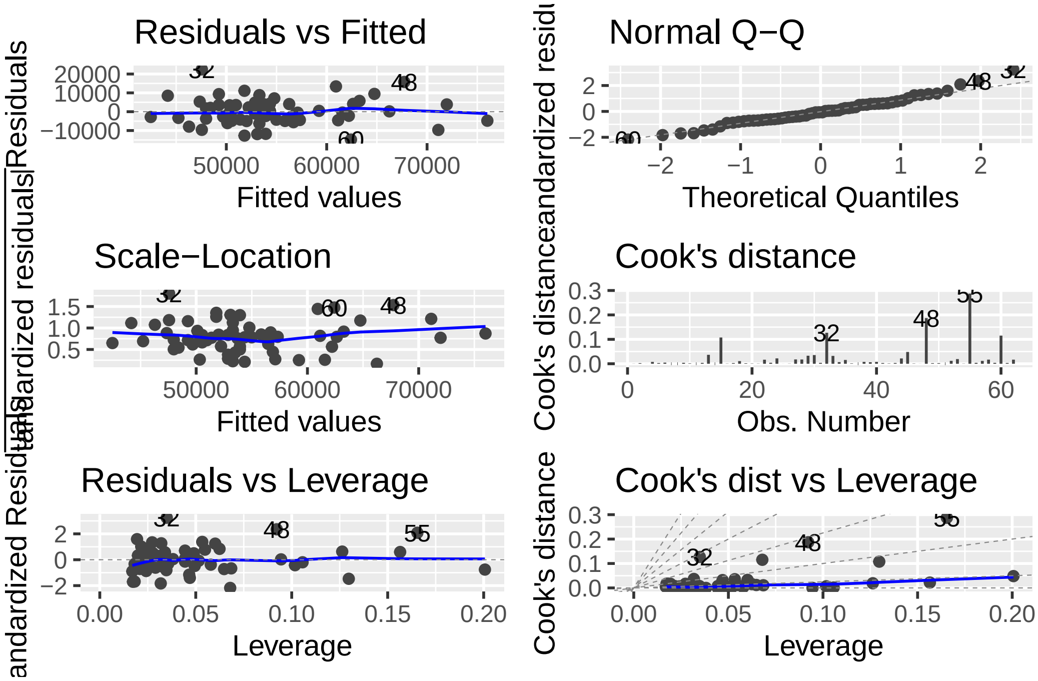
```

Call:
lm(formula = salary ~ degree_yrs * cits * gender, data = cohen1)

Residuals:
     Min       1Q   Median       3Q      Max 
-13966.3  -4404.6   -773.2   3657.9  22513.1 

Coefficients:
                           Estimate Std. Error t value Pr(>|t|)   
(Intercept)                26877.05    8716.17   3.084  0.00322 **
degree_yrs                  3715.49    1591.34   2.335  0.02330 * 
cits                         632.12     243.37   2.597  0.01208 * 
gendermale                 11287.69   10484.27   1.077  0.28643   
degree_yrs:cits              -89.18      45.11  -1.977  0.05315 . 
degree_yrs:gendermale      -2442.09    1793.28  -1.362  0.17892   
```

```
cits:gendermale                -403.30     275.46  -1.464  0.14897
degree_yrs:cits:gendermale       86.42      47.53   1.818  0.07457 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 7051 on 54 degrees of freedom
Multiple R-squared:  0.5328,    Adjusted R-squared:  0.4722
F-statistic: 8.798 on 7 and 54 DF,  p-value: 3.563e-07
```

13.3 An additional tool for diagnostic and influence visualization.

In addition to the several graphical tools outlined above, the `autoplot` function may be a helpful and efficient way of evaluating a model.

```
ggplot2::autoplot(fit6b, which=1:6,ncol=2,label.size=3)
```



14 Efficiency in OLS analysis using the olsrr package

The **olsrr** package has a collection of functions that streamline many of the piecemeal approaches outlined in earlier chapters. It makes it simple to obtain extensive/detailed information at once, with user-friendly functions. Not all of the topics covered in earlier chapters are included, but extensive capabilities are also included for model criticism.

14.1 A basic analysis

The initial use of the `ols_regress` function can replace the individual uses of `summary`, `anova`, and `confint` functions. The model fit here is the three-IV model examined in the "Extensions" chapter.

```r
fit6 <- lm(salary~pubs+cits+degree_yrs, data=cohen1)
ols_regress(fit6)
```

```
                             Model Summary
------------------------------------------------------------------------
R                          0.708       RMSE                    6799.969
R-Squared                  0.501       MSE                 46239579.851
Adj. R-Squared             0.475       Coef. Var                 12.826
Pred R-Squared             0.420       AIC                     1280.208
MAE                     5317.619       SBC                     1290.844
------------------------------------------------------------------------
 RMSE: Root Mean Square Error
 MSE: Mean Square Error
 MAE: Mean Absolute Error
 AIC: Akaike Information Criteria
 SBC: Schwarz Bayesian Criteria

                                   ANOVA
-----------------------------------------------------------------------------
                      Sum of
```

```
                     Squares        DF      Mean Square       F         Sig.
----------------------------------------------------------------------------
Regression    2879765872.603         3    959921957.534     19.42    0.0000
Residual      2866853950.768        58     49428516.393
Total         5746619823.371        61
----------------------------------------------------------------------------

                                       Parameter Estimates
--------------------------------------------------------------------------------------------
      model           Beta    Std. Error    Std. Beta         t        Sig         lower        up
--------------------------------------------------------------------------------------------
(Intercept)      38967.847      2394.308                   16.275     0.000     34175.118     43760
       pubs         93.608        85.348        0.135       1.097     0.277         -
77.235      264.451
       cits        204.060        56.972        0.361       3.582     0.001        90.019       318
 degree_yrs        874.461       283.895        0.385       3.080     0.003       306.184      1442
--------------------------------------------------------------------------------------------
```

14.2 Collinearity diagnostics

The `ols_coll_diag` function provides collinearity diagnostics:

```
ols_coll_diag(fit6)
```

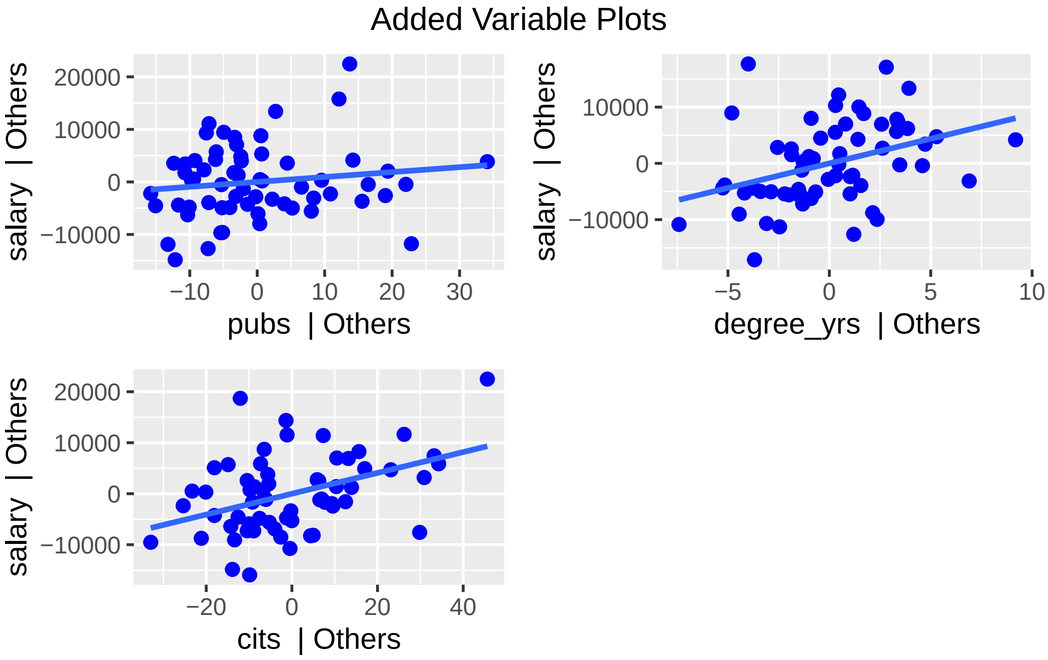
```
Tolerance and Variance Inflation Factor
---------------------------------------
   Variables Tolerance      VIF
1       pubs 0.5672118 1.763010
2       cits 0.8466483 1.181128
3 degree_yrs 0.5494035 1.820156


Eigenvalue and Condition Index
------------------------------
  Eigenvalue Condition Index   intercept        pubs        cits   degree_yrs
1 3.55809347        1.000000 0.009810827 0.014093645 0.009215181 0.0110334455
2 0.25588579        3.728942 0.146119575 0.358711061 0.097741038 0.0590237875
3 0.10745243        5.754407 0.015671115 0.619788632 0.026076066 0.9299270069
4 0.07856832        6.729533 0.828398483 0.007406662 0.866967715 0.0000157601
```

14.3 Added-variable plots, partial and semi-partial correlations

We can also obtain added-variable plots. These depict the partial correlations of each IV with the DV, both adjusted for other IVs. Here, it can be seen that pubs has a weaker partial relationship and this reinforces the fact that it's test in the 3-IV model was non-significant. Recall that the partial correlation could be obtained with the `mrinfo` function. Partial and semi-partial correlations are provided with the `ols_correlations` function here.

```
ols_plot_added_variable(fit6)
```

```
`geom_smooth()` using formula = 'y ~ x'
`geom_smooth()` using formula = 'y ~ x'
`geom_smooth()` using formula = 'y ~ x'
```



```
ols_correlations(fit6)
```

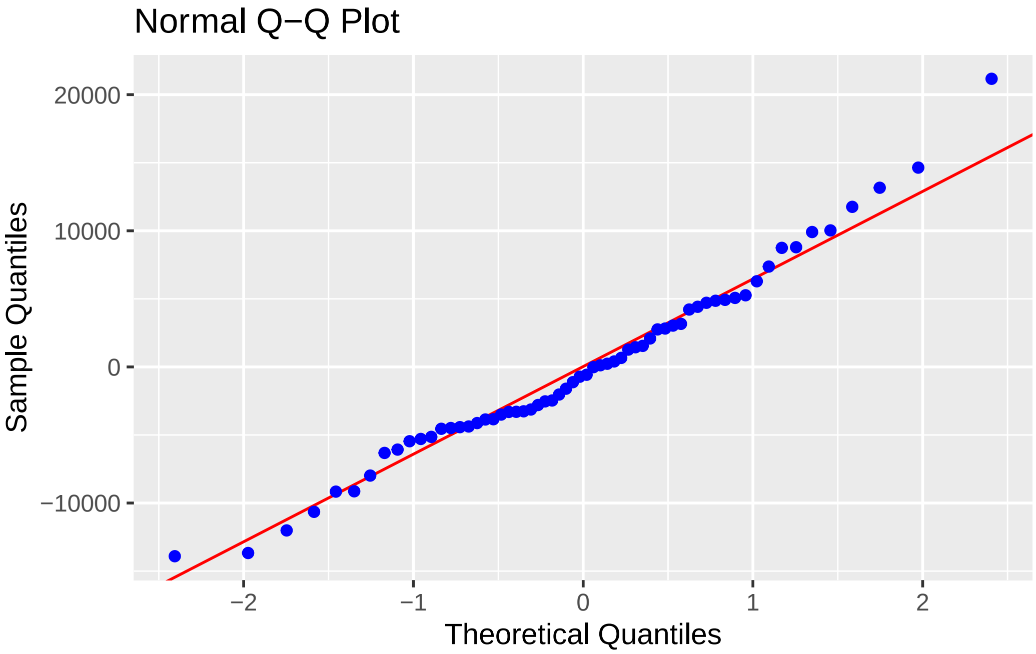
```
                    Correlations
---------------------------------------------------
Variable        Zero Order      Partial     Part
---------------------------------------------------
pubs                 0.506        0.143    0.102
```

```
cits                   0.550       0.426     0.332
degree_yrs             0.608       0.375     0.286
--------------------------------------------
```

14.4 Residual Assumptions

A normal QQ plot of the residuals is available.

```
ols_plot_resid_qq(fit6)
```



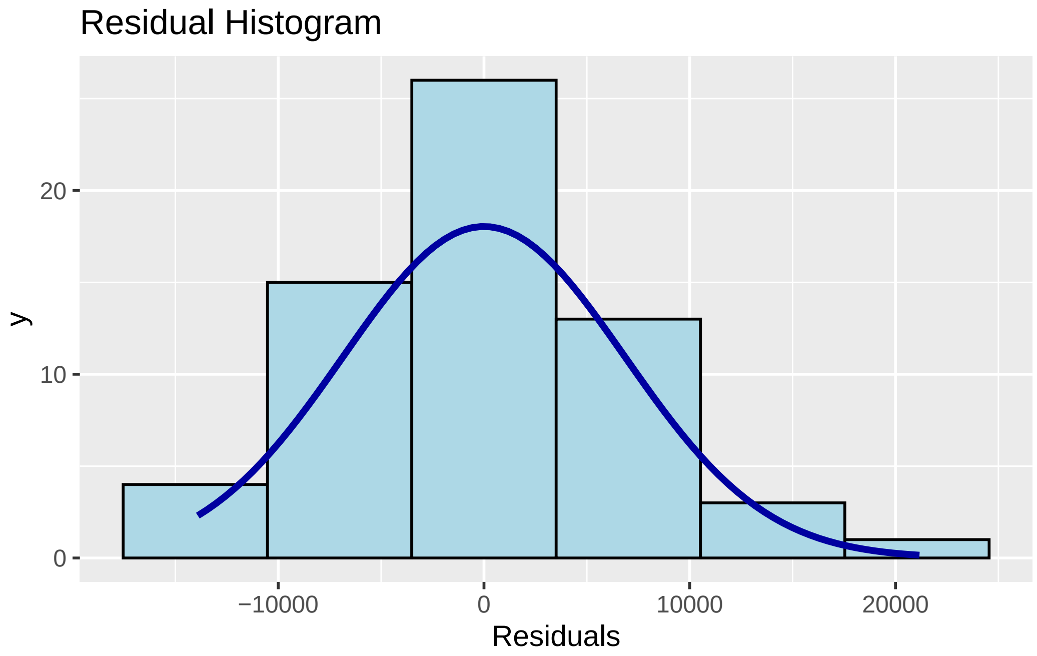
Tests of the residual normality assumption are easily obtained.

```
ols_test_normality(fit6)
```

```
-----------------------------------------------
       Test             Statistic       pvalue
-----------------------------------------------
Shapiro-Wilk              0.9782         0.3374
Kolmogorov-Smirnov        0.0759         0.8404
Cramer-von Mises          5.2312         0.0000
Anderson-Darling          0.4327         0.2946
-----------------------------------------------
```

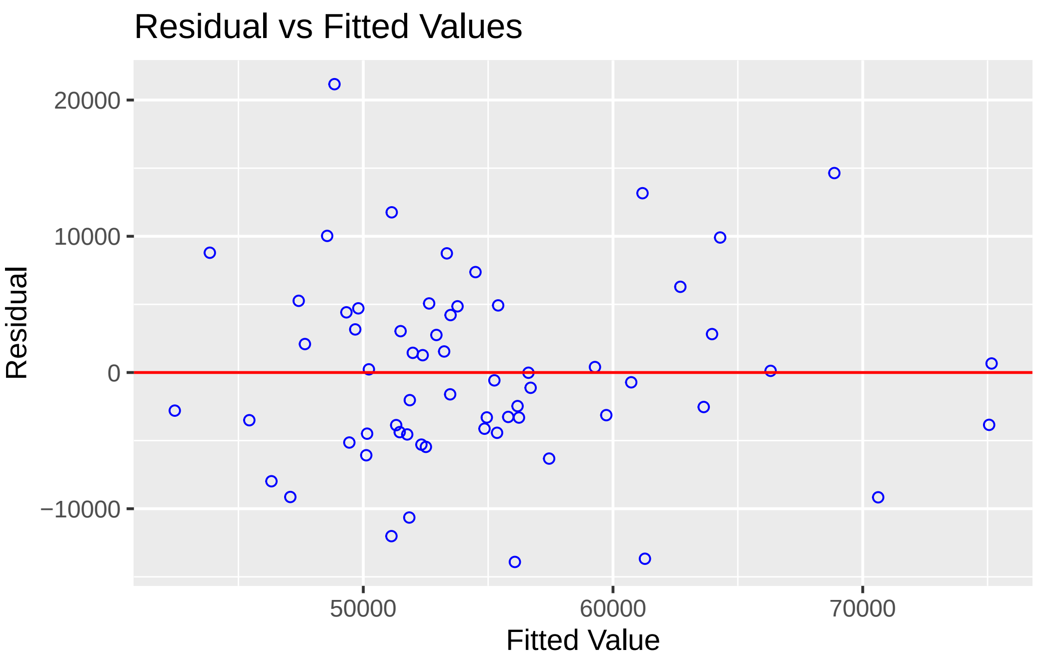
A histogram of the residuals with a normal curve overlaid for comparison is also available.

```
ols_plot_resid_hist(fit6)
```



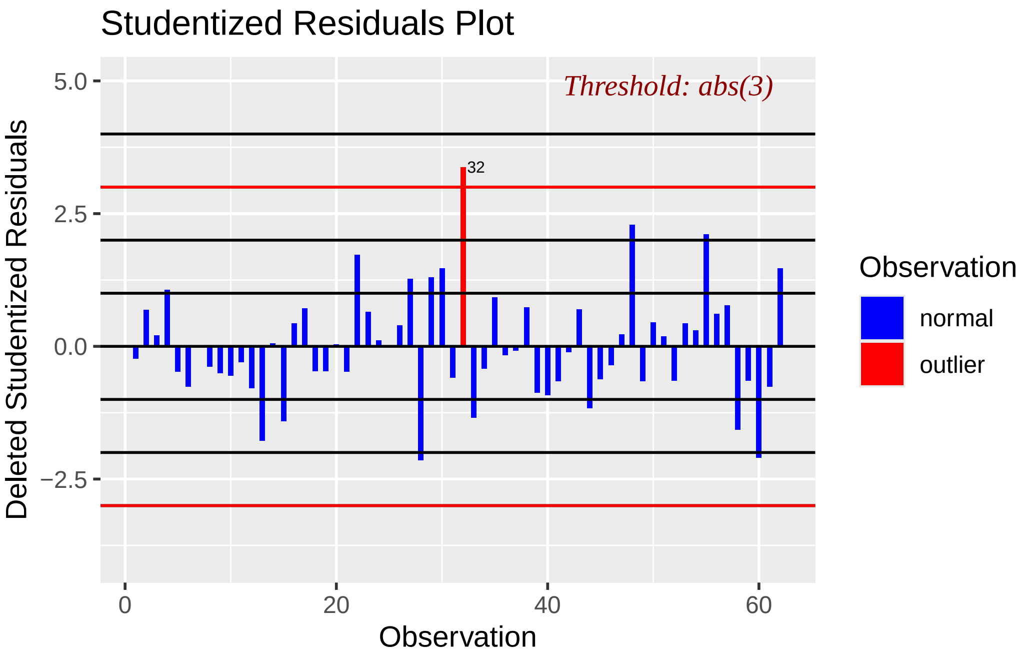
And the standard plot of residuals against yhats is also available.

```
ols_plot_resid_fit(fit6)
```

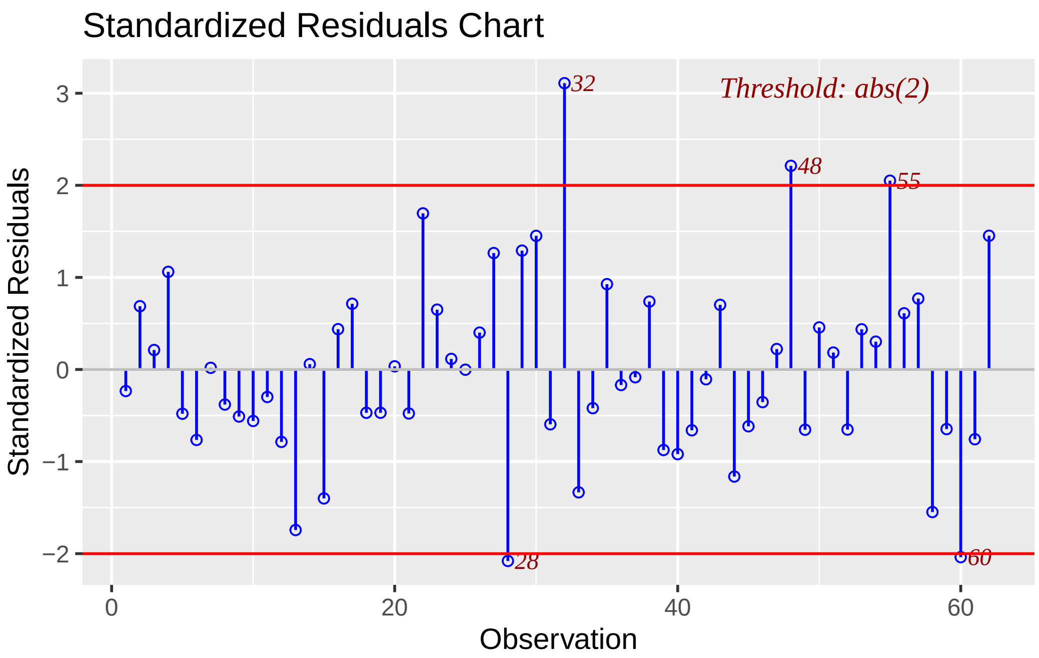


Studentized and Standardized residuals can be examined to look for sequential ordering effects in the data set by plotting them against the case number.

```
ols_plot_resid_stud(fit6)
```



```
ols_plot_resid_stand(fit6)
```



Several of the above plots plus other model diagnostic plots can be obtained more rapidly with the `ols_plot_diagnostics` function for which the code is shown here. The plots are not returned in order to save space.

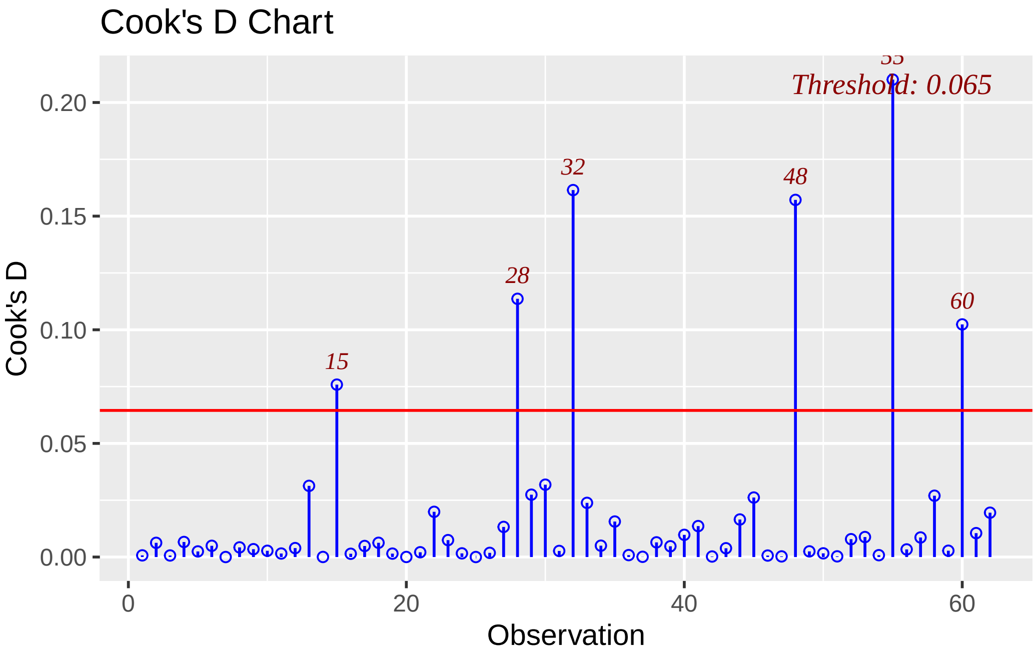
```
ols_plot_diagnostics(fit6)
```

14.5 Plots for examination of Influence

Several plots are available for visualizing the influence statistics for a model.

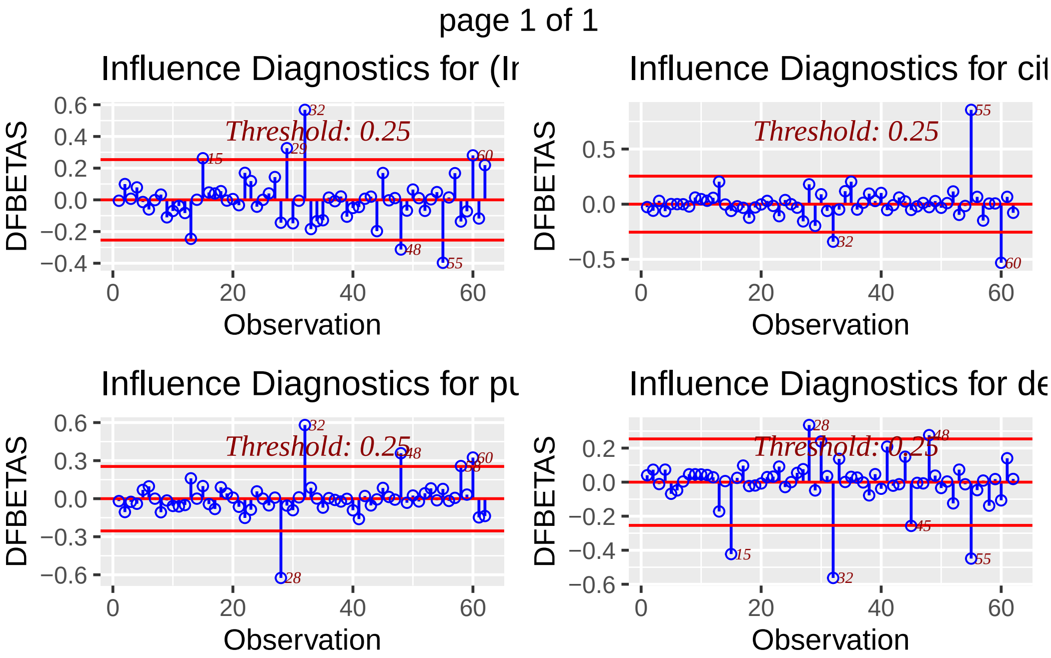
First is examination of Cook's D values. It provides a visual indicators of which cases exceed a threshold for large influence and those cases are numerically labeled. I have not yet sorted out how this threshold is determined for this function and the following ones.

```
ols_plot_cooksd_chart(fit6)
```



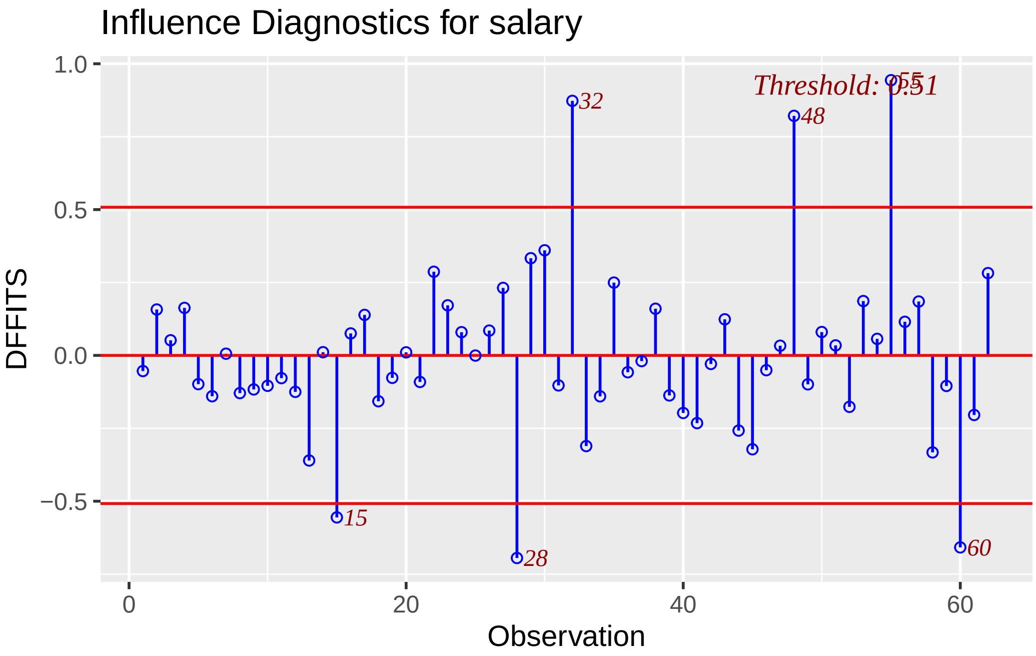
The DFBeta index is visualized with a panel of graphs, one for each IV and one for the intercept, permitting identification of influential cases for each IV separately.

```
ols_plot_dfbetas(fit6)
```



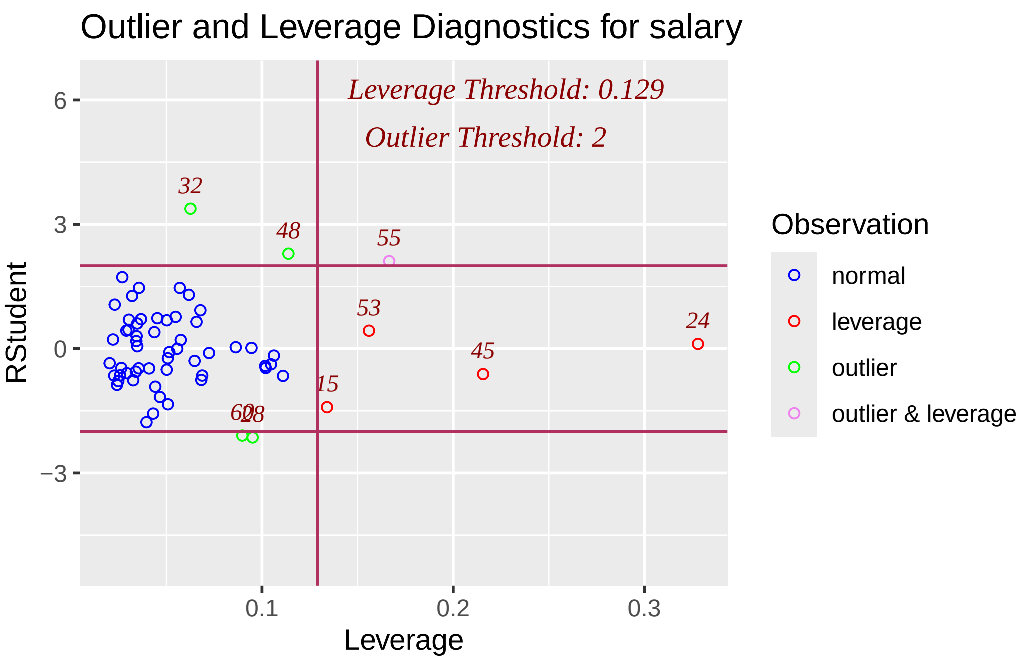
And a comparable plot for DFfits is also available.

```
ols_plot_dffits(fit6)
```

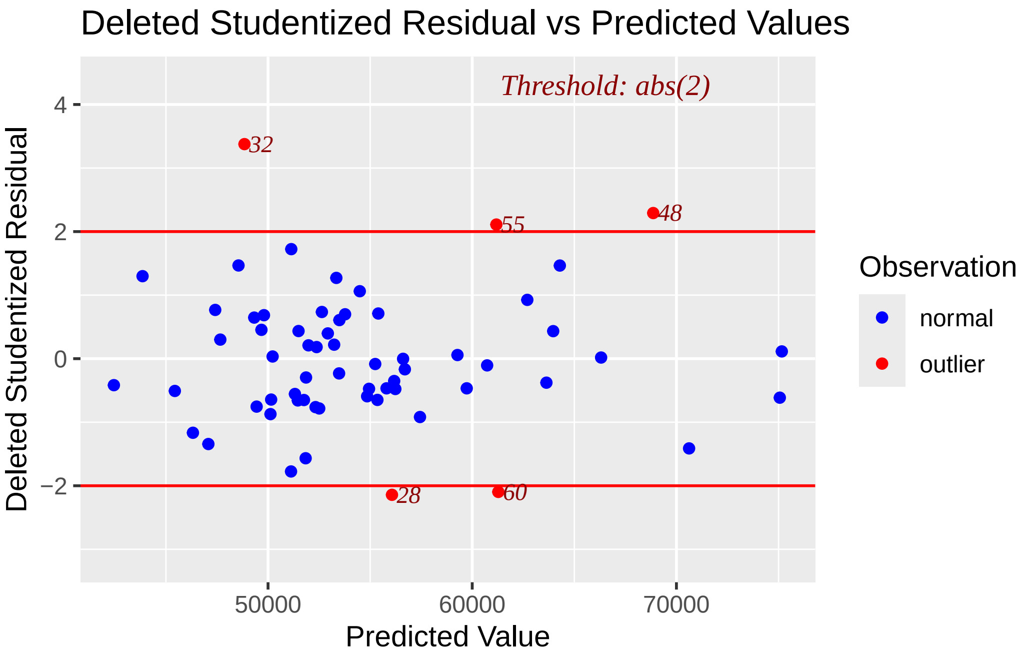


Finally, two additional plots are common in model diagnostics. They examine studentized residuals and deleted studentized residuals against leverage and yhats, respectively.

```
ols_plot_resid_lev(fit6)
```



```
ols_plot_resid_stud_fit(fit6)
```



The **olsrr* package has numerous other tools and is worth exploring. The reference manual and vignettes on the CRAN site are very helpful.

olsrr on CRAN

15 Variable Selection and Model Building

Model building with OLS methods requires approaching the question "what is the best model". This often translates to exactly which set of independent variables are included in a final model. The issue is that "best" can be defined several ways. Traditional textbooks on multiple regression and common culture it its application has for many years employed a style of variable selection that is algorithmic/automated. Common among these methods have been p-value based approaches and we have covered those (e.g., forward selection, stepwise, backward selection, etc). These methods have received much criticism in recent years, in part due to their a-theoretical nature. Modern texts have taken this critical perspective Harrell (2015). An online version of the Harrell text is available.

Newer approaches to model building involve information criteria based techniques and some of that was discussed in the chapter above on Model Comparisons.

This chapter will not effort to lay out all of the possible methods, traditional and new that have been employed and implemented in R. Rather, an list of resources is provided. If there comes a time when I feel that I have something to add to those extensive resources, I will revise this chapter.

Two review papers can provide background and context:

1. Heinze, et al, 2017
2. Ullmann, et al 2024

Multiple online resources outline how to implement the various methods in R.

The **olsrr** package has an extensive suite of functions and strong documentation:

olsrr Variable Selection Methods

And a wide selection of other possibilities:

Donatello & Roualdes Sanderson Porter Ruczinski Vandormael Kecojevic Sestelo, et al Imran, et al

16 Bayes Factor approach to multiple regression

Recalling that this tutorial document for linear modeling is intended for an introductory statistics level, the reader should understand that only a superficial treatment of Bayesian Inference is done here. The type bayesian analysis demonstrated here is also a narrow treatment of the topic that itself has received some criticism.

There are many approaches and flavors of Bayesian inference. A major set of tools is found in the Stan libraries. In R, these are implemented in the `rstan` package. There are extensive ways of doing inference for linear models (and many other models) including using interfaces to standalone software - see the task view on CRAN:

Cran Task View: Bayesian Inference

These approaches are often laborious, and require far more background in Bayesian methods than most readers of this document will have at this point in time. Alternatives involve using STAN or BUGS algorithms. They are implemented in apps such as WinBugs for the Windows platform and OpenBugs. An R package written by Andrew Gelman is an interface to WinBugs or OpenBugs and is called `R2WinBUGS`. A variety of Stan interfaces are used frequently. These packages are not illustrated here, but links to useful sites are:

https://mc-stan.org/users/interfaces/rstan

https://cran.r-project.org/web/packages/R2WinBUGS/index.html

Other high-level modeling interfaces that I have begun to see used with some frequency are implemented in the **brms** and **rstanarm** packages which have an emphasis on regression models:

brms rstanarm

In recent years, another kind of bayesian inference has gained traction is social, behavioral, and life sciences. This approach involves a focus on Bayes Factors as an index of support for or against hypotheses. It is not the purpose of this document to provide a full background on the Bayes Factor logic or method. Nonetheless, it is fairly simple to implement a Bayes Factor analysis of the rudimentary multiple regression methods covered in this document. An extensive collection of articles/books on the topic of Bayes Factor usage is found in the toolkit bibliography for the 510/511 course.

16.1 Basic approaches with Bayes Factors

The **BayesFactor** package, developed by Richard Morey (Morey & Rouder, 2018), is well documented and extensive tutorials are available.

For example:

https://richarddmorey.github.io/BayesFactor/

This section provides a brief overview of rudimentary analyses that create Bayes Factor evaluations of the linear model. I will use the two-IV model that we settled on as a reasonable one, the one with degrees_yr and cits as IVs and salary as DV. First, the ordinary least squares model is repeated so that we can compare results.

```
olsfit <- lm(salary~degree_yrs+cits,data=cohen1)
kable(tidy(olsfit))
```

term	estimate	std.error	statistic	p.value
(Intercept)	39073.6747	2396.47360	16.304655	0.0000000
degree_yrs	1061.7642	227.17563	4.673759	0.0000176
cits	212.1116	56.59392	3.747958	0.0004078

The `regressionBF` function permits simultaneous comparison of several models. Evaluation of only one model can use the `lmBF` function. The goal is the calculation of Bayes Factor indices permitting comparison of models and yielding a metric that is interpreted as relative evidence for one hypothesis over another. Initially here, the `regressionBF` function is used to assess three different models against an "intercept-only" model. And then we will compare the three models against each other. Why three models? With two IVs there are three possible models of interest - one each with a single IV and a third with both IVs.

With the large sample size and the sizable correlations of degree_yrs and cits with salary, it is not surprising that each model yields a Bayes Factor that is a very large value. The BF is interpreted as a ratio. For example, support for the degree_yrs - only model is found to be over 7000 times more likely than the intercept-only model. The two-IV model has extremely an extremely large BF. It is important to understand that there is substantial theory underlying the BF computation and assumptions have been made to create default shape and scaling of choices for the prior. The reader is expected to become skilled in these choices and the theory before use of these methods.

```
bf1 <- regressionBF(salary~degree_yrs+cits, data=cohen1)
bf1
```

```
Bayes factor analysis
--------------
[1] degree_yrs        : 71125.54 ±0.01%
[2] cits              : 4126.233 ±0%
[3] degree_yrs + cits : 4963640  ±0%

Against denominator:
  Intercept only
---
Bayes factor type: BFlinearModel, JZS
```

At this point we have seen that both the OLS and BF models offer strong support for the idea that the degree_yrs plus cits model is a strong one. But it might be informative to look at a model that has much weaker evidence.

We can reconsider the model that used gender as an IV. It was not a significant predictor in the OLS model, which is repeated here.

```
olsfit2 <- lm(salary~gender,data=cohen1)
kable(tidy(olsfit2))
```

term	estimate	std.error	statistic	p.value
(Intercept)	52650.000	1810.539	29.079734	0.0000000
gendermale	3949.324	2444.918	1.615319	0.1114884

In a BayesFactor analysis we find a very different sized BF than for the degree_yrs+cits model. A BF value of 1.0 would indicate equivalent support for the null hypothesis (β=0) and the alternative ($\beta \neq 0$). The BF value found here (.773) indicates equivocal support for each with only very slight evidence favoring the null. We can take the reciprocal of the BF to find the degree of relative evidence for the null. That value is 1/.773 or 1.29. So we would say that the null is 1.29 times more likely than the null

```
bf1b <- lmBF(salary~gender, data=cohen1)
bf1b
```

```
Bayes factor analysis
--------------
[1] gender : 0.7726611 ±0.01%

Against denominator:
  Intercept only
---
Bayes factor type: BFlinearModel, JZS
```

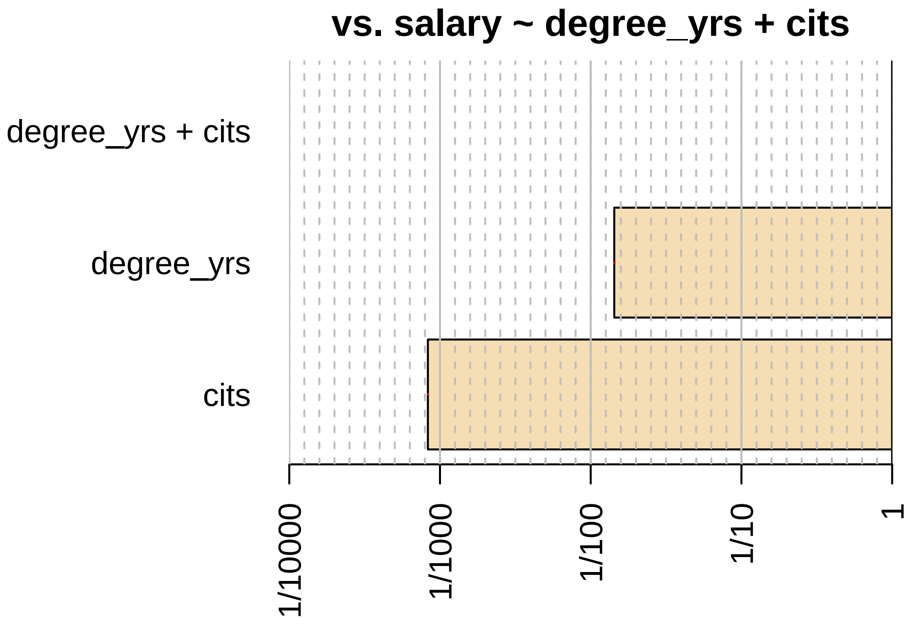
Returning to the original two-IV model, the **BayesFactor** package permits some interesting and useful approaches to model comparison. Above, we only compared each of the three possible models to an intercept-only model. Now, we will compare them to each other. The relative evidence for one particular model can be found by taking the ratio of the BF to those models. Here, I compare the "best" model (using `max`) to each of the others. First, see that comparing the best model (the two-IV model) to itself yields a ratio of 1, and this makes sense. Against the two-IV model,degree__yrs-only has less than 2% the strength of evidence times the strength of evidence. Or take the reciprocal and find that the two-IV model is nearly 70 times more likely Against the cits-only model the two-IV model fares even better with the cits-only model garnering less than .1% support.

```r
bf2 <- bf1/max(bf1)
bf2
```

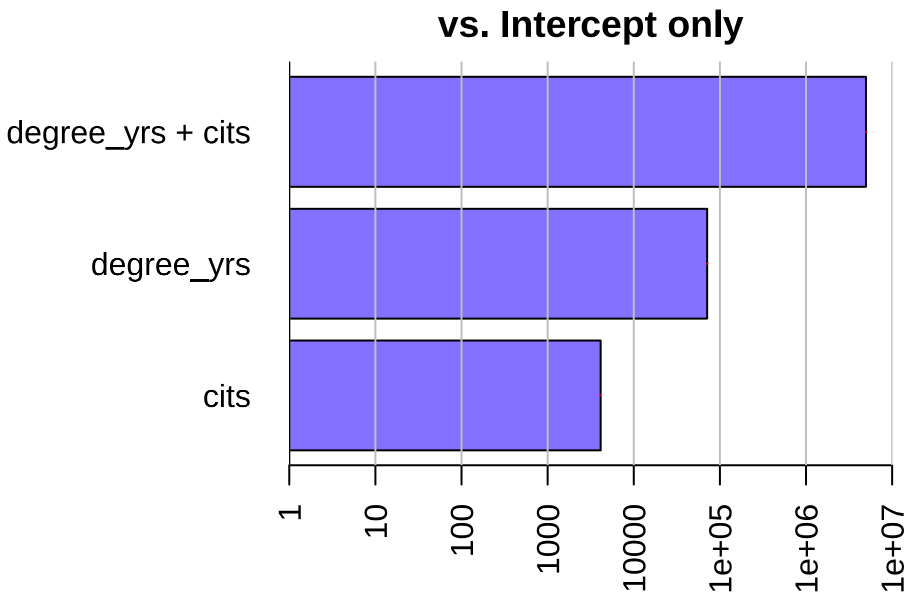
```
Bayes factor analysis
--------------
[1] degree_yrs        : 0.01432931   ±0.01%
[2] cits              : 0.0008312918 ±0%
[3] degree_yrs + cits : 1            ±0%

Against denominator:
  salary ~ degree_yrs + cits
---
Bayes factor type: BFlinearModel, JZS
```

It is possible to draw some useful plots of competing models. In this simplistic system, these plots are redundant, but with more IVs and more possible models, the visualizations can be a rapid way of comparing models.

```r
plot(bf2) # the ratio models
```



```
plot(bf1) # the comparison to intercept-only model
```



This brief exposition is a superficial treatment of BF methods, but may give an indication of the relative ease with which they can be performed in R. There are well-done tutorials on the Morey's web page cited above, and a blog provides well written background theory:

[http://bayesfactor.blogspot.com/2014/02/the-bayesfactor-package-this-blog-is.html]

16.2 An admonition and suggestion regarding Bayesian Inference

It is important to understand that there is substantial theory underlying the BF computation and assumptions have been made to create default shape and scaling of choices for the prior. The reader is expected to become skilled in these choices and the theory before use of these methods. A voluminous literature is available for the logic, theory, and implementation of Bayesian methods. A substantial amount of that literature is available via the stattoolkit bibliography shared with the readers.

For simple multiple regression models, the reader might wish to examine the capabilities of the JASP software, an app that is built on an R foundation. It has simple menu-driven capabilities for Bayesfactor applications to a wide variety of inferential situations: JASP

17 Documentation for Reproducibility

R software products such as this markdown document should be simple to reproduce, if the code file or code are available. But it is also important to document the exact versions of the R installation, the OS, and the R packages in place when the document is created.

```r
sessionInfo()
```

```
R version 4.5.2 (2025-10-31 ucrt)
Platform: x86_64-w64-mingw32/x64
Running under: Windows 11 x64 (build 26200)

Matrix products: default
  LAPACK version 3.12.1

locale:
[1] LC_COLLATE=English_United States.utf8
[2] LC_CTYPE=English_United States.utf8
[3] LC_MONETARY=English_United States.utf8
[4] LC_NUMERIC=C
[5] LC_TIME=English_United States.utf8

time zone: America/New_York
tzcode source: internal

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

loaded via a namespace (and not attached):
 [1] compiler_4.5.2    fastmap_1.2.0     cli_3.6.5         tools_4.5.2
 [5] htmltools_0.5.8.1 rstudioapi_0.17.1 rmarkdown_2.29    knitr_1.50
 [9] jsonlite_2.0.0    xfun_0.53         digest_0.6.37     rlang_1.1.6
[13] evaluate_1.0.4
```

17.1 Revision History

Ver1.4 February 7, 2026

Converted to Quarto, updated code and text, corrected typos, added sections…

Ver1.3 January 29, 2026

Revised Chapter 11 on diagnostics/influence

Ver1.2 April 7, 2020

Converted the document to bookdown

Added many sections and updated many code chunks

Ver 1.1 Jan. 29, 2018

ver 1.0 Feb. 2, 2017

References

Allaire, J., Xie, Y., McPherson, J., Luraschi, J., Ushey, K., Atkins, A., … Iannone, R. (2020). *Rmarkdown: Dynamic documents for r*. Retrieved from https://CRAN.R-project.org/package=rmarkdown

Arnold, J. B. (2019). *Ggthemes: Extra themes, scales and geoms for 'ggplot2'*. Retrieved from https://CRAN.R-project.org/package=ggthemes

Attali, D., & Baker, C. (2019). *ggExtra: Add marginal histograms to 'ggplot2', and more 'ggplot2' enhancements*. Retrieved from https://CRAN.R-project.org/package=ggExtra

Auguie, B. (2017). *gridExtra: Miscellaneous functions for "grid" graphics*. Retrieved from https://CRAN.R-project.org/package=gridExtra

Canty, A., & Ripley, B. (2019). *Boot: Bootstrap functions (originally by angelo canty for s)*. Retrieved from https://CRAN.R-project.org/package=boot

Cohen, J., Cohen, P., West, S. G., & Aiken, L. S. (2003). *Applied multiple regression/correlation analysis for the behavioral sciences* (3rd ed., pp. xxviii, 703 p.). Book, Mahwah, NJ: L. Erlbaum Associates.

Cook, R. D., & Weisberg, S. (1999). *Applied regression including computing and graphics* (pp. xxvi, 593 p.). Book, New York: Wiley.

Darlington, R. B. (1990). *Regression and linear models* (pp. xxxii, 542 p.). Book, New York: McGraw-Hill.

Dudek, B. (2026). *Bcdstats: A collection of functions to support b. Dudek's APSY510/511 classes*. Retrieved from https://https://github.com/bcdudek/bcdstats

file., S. A. (2024). *Paletteer: Comprehensive collection of color palettes*. Retrieved from https://github.com/EmilHvitfeldt/paletteer

Fox, J. (2016). *Applied regression analysis and generalized linear models* (Third Edition, pp. xxiv, 791 pages). Book, Los Angeles: SAGE.

Fox, J., Weisberg, S., & Fox, J. (2011). *An r companion to applied regression* (2nd ed., pp. xxii, 449 p.). Book, Thousand Oaks, Calif.: SAGE Publications.

Fox, J., Weisberg, S., & Price, B. (2020). *Car: Companion to applied regression*. Retrieved from https://CRAN.R-project.org/package=car

Gross, J., & Ligges, U. (2015). *Nortest: Tests for normality*. Retrieved from https://CRAN.R-project.org/package=nortest

Hamner, B., & Frasco, M. (2018). *Metrics: Evaluation metrics for machine learning*. Retrieved from https://github.com/mfrasco/Metrics

Harrell, J. F. E. (2015). *Regression modeling strategies : With applications to linear models, logistic and ordinal regression, and survival analysis* (pp. 1 online resource (XXV, 582 pages 157 illustrations, 53 illustrations in color). Electronic Book, Springer International Publishing : Imprint: Springer,.

Hebbali, A. (2020). *Olsrr: Tools for building OLS regression models*. Retrieved from https://CRAN.R-project.org/package=olsrr

Heiberger, R. M. (2020). *HH: Statistical analysis and data display: Heiberger and holland*. Retrieved from https://CRAN.R-project.org/package=HH

Horikoshi, M., & Tang, Y. (2018). *Ggfortify: Data visualization tools for statistical analysis results*. Retrieved from https://CRAN.R-project.org/package=ggfortify

Hothorn, T., Zeileis, A., Farebrother, R. W., & Cummins, C. (2019). *Lmtest: Testing linear regression models*. Retrieved from https://CRAN.R-project.org/package=lmtest

Howell, D. C. (2013). *Statistical methods for psychology* (8th ed., pp. xix, 770 p.). Book, Belmont, CA: Wadsworth Cengage Learning.

Iannone, R., Cheng, J., & Schloerke, B. (2019). *Gt: Easily create presentation-ready display tables*. Retrieved from https://github.com/rstudio/gt

Komsta, L., & Novomestky, F. (2015). *Moments: Moments, cumulants, skewness, kurtosis and related tests*. Retrieved from https://CRAN.R-project.org/package=moments

Mangiafico, S. (2020). *Rcompanion: Functions to support extension education program evaluation*. Retrieved from https://CRAN.R-project.org/package=rcompanion

Morey, R. D., & Rouder, J. N. (2018). *BayesFactor: Computation of bayes factors for common designs*. Retrieved from https://CRAN.R-project.org/package=BayesFactor

Nimon, K., Oswald, F., & Roberts., J. K. (2013). *Yhat: Interpreting regression effects*. Retrieved from https://CRAN.R-project.org/package=yhat

Pena, E. A., & Slate, E. H. (2006). Global validation of linear model assumptions. *J Am Stat Assoc*, *101*, 341. Journal Article.

Pena, Edsel A., & Slate, E. H. (2019). *Gvlma: Global validation of linear models assumptions*. Retrieved from https://CRAN.R-project.org/package=gvlma

Revelle, W. (2020). *Psych: Procedures for psychological, psychometric, and personality research*. Retrieved from https://CRAN.R-project.org/package=psych

Ripley, B. (2019). *MASS: Support functions and datasets for venables and ripley's MASS*. Retrieved from https://CRAN.R-project.org/package=MASS

Robinson, D., & Hayes, A. (2020). *Broom: Convert statistical analysis objects into tidy tibbles*. Retrieved from https://CRAN.R-project.org/package=broom

RStudio Team. (2015). *RStudio: Integrated development environment for r*. Boston, MA: RStudio, Inc. Retrieved from http://www.rstudio.com/

Sarkar, D. (2020). *Lattice: Trellis graphics for r*. Retrieved from https://CRAN.R-project.org/package=lattice

Schloerke, B., Crowley, J., Cook, D., Briatte, F., Marbach, M., Thoen, E., … Larmarange, J. (2020). *GGally: Extension to 'ggplot2'*. Retrieved from https://CRAN.R-project.org/package=GGally

Soetaert, K. (2016). *plot3Drgl: Plotting multi-dimensional data - using 'rgl'*. Retrieved from https://CRAN.R-project.org/package=plot3Drgl

Soetaert, K. (2019). *plot3D: Plotting multi-dimensional data*. Retrieved from https://CRAN.R-project.org/package=plot3D

Trapletti, A., & Hornik, K. (2019). *Tseries: Time series analysis and computational finance.* Retrieved from https://CRAN.R-project.org/package=tseries

Weisberg, S. (2014). *Applied linear regression* (Fourth edition., pp. xvii, 340 pages). Book, Hoboken, NJ: Wiley.

Wickham, H. (2020). *Plyr: Tools for splitting, applying and combining data.* Retrieved from https://CRAN.R-project.org/package=plyr

Wickham, H., Chang, W., Henry, L., Pedersen, T. L., Takahashi, K., Wilke, C., … Dunnington, D. (2020). *ggplot2: Create elegant data visualisations using the grammar of graphics.* Retrieved from https://CRAN.R-project.org/package=ggplot2

Wright, D. B., & London, K. (2009). *Modern regression techniques using r : A practical guide for students and researchers* (pp. viii, 204 p.). Book, Los Angeles ; London: SAGE.

Xie, Y. (2015). *Dynamic documents with R and knitr* (2nd ed.). Boca Raton, Florida: Chapman; Hall/CRC. Retrieved from http://yihui.name/knitr/

Xie, Y. (2020). *Knitr: A general-purpose package for dynamic report generation in r.* Retrieved from https://CRAN.R-project.org/package=knitr

Yan, Y. (2024). *MLmetrics: Machine learning evaluation metrics.* Retrieved from https://github.com/yanyachen/MLmetrics

Zeileis, A., & Lumley, T. (2019). *Sandwich: Robust covariance matrix estimators.* Retrieved from https://CRAN.R-project.org/package=sandwich