# 34

# Using R for Data Analysis

## *A Best Practice for Research*

Ken Kelley, Keke Lai, and Po-Ju Wu

R is an extremely flexible statistics programming language and environment that is Open Source and freely available for all mainstream operating systems. R has recently experienced an "explosive growth in use and in user contributed software" (Tierney, 2005, p. 7). The "user-contributed software" is one of the most unique and beneficial aspects of R, as a large number of users have contributed code for implementing some of the most up-to-date statistical methods, in addition to R implementing essentially all standard statistical analyses. Because of R's Open Source structure and a community of users dedicated to making R of the highest quality, the computer code on which the methods are based is openly critiqued and improved.[1] The flexibility of R is arguably unmatched by any other statistics program, as its object-oriented programming language allows for the creation of functions that perform customized procedures and/or the automation of tasks that are commonly performed. This flexibility, however, has also kept some researchers away from R. There seems to be a misperception that learning to use R is a daunting challenge. The goals of this chapter include the following: (a) convey that the time spent learning R, which in many situations is a relatively small amount, is a worthwhile investment; (b) illustrate that many commonly performed analyses are straightforward to implement; and (c) show that important methods not available elsewhere can be implemented in R (easily in many cases). In addition to these goals, we will show that an often unrealized benefit of R is that it helps to create "reproducible research," in the sense that a record will exist of the exact analyses performed (e.g., algorithm used, options specified, subsample selected, etc.) so that the results of analyses can be recovered at a later date by the original researcher or by others if necessary (and thus "How was this result obtained?" is never an issue).

Currently, R is maintained by the R Core Development Team. R consists of a base system with optional add-on packages for a wide variety of techniques that are contributed by users from around the world (currently, there are more than 1,100 packages available on the Comprehensive R Archival Network, http://cran.r-project.org/). An R package is a collection of functions and corresponding documentation that work seamlessly with R.R has been called the lingua franca of statistics by the editor of the *Journal of Statistical Software* (de Leeuw, 2005, p. 2).[2]

One of R's most significant advantages over other statistical software is its philosophy. In R, statistical analyses are normally done as a series of steps, with intermediate results being stored in

objects, where the objects are later "interrogated" for the information of interest (R Development Core Team, 2007b). This is in contrast to other widely used programs (e.g., SAS and SPSS), which print a large amount of output to the screen. Storing the results in objects so that information can be retrieved at later times allows for easily using the results of one analysis as input for another analysis. Furthermore, because the objects contain all pertinent model information, model modification can be easily performed by manipulation of the objects, a valuable benefit in many cases. R packages for new innovations in statistical computing also tend to become available more quickly than do such developments in other statistical software packages.

As Wilcox (Chapter 18, this volume) notes, a practical problem with modern methods is their implementation. Without accessible tools (i.e., software) to implement new methods, the odds of them being implemented is slim. Because R is cutting edge, many modern methods are available in R.

The need for implementing methods has led to much interest in R over the past few years in the behavioral, educational, and social sciences (BESS), and this trend will likely continue. For example, Doran and Lockwood (2006) provide a tutorial on using R to fit value-added longitudinal models for behavioral and educational data using the *nonlinear mixed effects* (nlme) package (Pinheiro, Bates, DebRoy, & Sarkar, 2007). There is also a special issue in the *Journal of Statistical Software,* with 10 articles on psychometrics in R, and statistical texts used in the applied BESS are beginning to incorporate R (e.g., Fox, 2002; Everitt, 2005). Further evidence comes from Wilcox (Chapter 18, this volume), who provides R functions that implement the methods he has developed for robust methods in R (and S-Plus, a related program).[3] Methods for the Behavioral, Educational, and Social Sciences (MBESS; Kelley, 2007a, 2007b, in press) is an R package that implements methods that are especially helpful for the idiosyncratic needs of the BESS researchers. For example, a set of functions within MBESS is for confidence interval formation for noncentral parameters from *t*, *F*, and chi-square distributions, which lead to functions for confidence interval formation for various effect sizes that require noncentral distributions (as discussed in Thompson, Chapter 17, this volume). In addition to confidence interval formation, MBESS contains functions for sample size planning from the power-analytic and accuracy in parameter estimation approaches for a variety of effects commonly of interest in the BESS.

Perhaps R's biggest hindrance is also its biggest asset, and that is its general and flexible approach to statistical inference. With R, if you know what you want, you can almost always get it . . . but you have to ask for it. Using R requires a more thoughtful approach to data analysis than does using some other programs, but that dates back to the idea of the S language being one where the user interacts with the data, as opposed to a "shotgun" approach, where the computer program provides everything thought to be relevant to the particular problem (Becker, 1994, p. 1). For those who want to stay on the cutting edge of statistical developments, using R is a must. This chapter begins with arithmetic operations and illustration of simple functions. Commonly used methods (e.g., multiple regression, *t* tests, analysis of variance, longitudinal methods) and advanced techniques within these methods (e.g., confidence intervals for standardized effect sizes, visualization techniques, sample size planning) are then illustrated. We hope this chapter will convey that using R is indeed a *best practice* and can be a valuable tool in research.

## Basic R Commands

As mentioned, R is an object-oriented language and environment where objects, whether they be a single number, data set, or model output, are stored within an R session/workspace. These objects can then be used within functions, used to create other objects, or removed as appropriate. In fact, a function itself is an object. The expression <– is the assignment operator (assign what is on the right to the object on the left), as is –> (assign what is on the left to the object on the right). Expressions are entered directly into an R session at the prompt, which is generally denoted >. In this chapter, we use `R>` as the prompt to emphasize that the R code that follows is directly executable.

Suppose a data set, `my.data`, exists within an R session (we discuss loading data files in the next section). Typing `my.data` and then pressing enter/return will display the values contained in the `my.data` data set:

```
R> my.data
R>     x     y
1      1     2
2      3     4
3      3     8
4      4     9
5      5    10
```

As can be seen, my.data is a 5-by-2 matrix with the first column labeled x and the second labeled y.

The square brackets, "[ ]," can be used to extract information from a data set (or matrix), by specifying the specific values to extract. For example, consider the following commands:

```
R> x <- my.data[,1]
R> y <- ma.data[,2]
R> x
[1] 1 3 3 4 5
R> y
[1] 2 4 8 9 10
```

The first command extracts the first column of my.data, the vector x, and the second command extracts the second column, the vector y. Notice the comma that separates rows and columns. Since no rows were specified, all were selected. We can obtain various pieces of information from the objects by using functions. For example, applying the following functions returns the sum, length, mean, and the variance of the vector x, respectively:

```
R> sum(x) #the summation of x
[1] 16
R> length(x) #the number of components of x
[1] 5
R> mean(x) #the mean of x
[1] 3.2
R> var(x) #the variance of x
[1] 2.2
```

Notice the use of the number sign (#) for comments; anything that follows a number sign on a line is ignored. R uses vectorized arithmetic, which implies that most equations are implemented in R as they are written, both for scalar and matrix algebra (in general). Many computing languages have their own idiosyncratic language for scalar and especially matrix algebra.

To obtain the summary statistics for a matrix instead of a vector, functions can be used in a similar fashion. Using the data set my.data, consider the following commands, which are analogous to the commands applied to the vector:

```
R> sum(my.data)
[1] 49
R> length(my.data)
[1] 2
R> mean(my.data)
   x    y
 3.2  6.6
R> var(my.data)
     x    y
x  2.2  4.6
y  4.6 11.8
```

In fact, the same functions were used, but R is intelligent enough to apply them differently depending on the type of data specified (e.g., a vector, matrix, data frame, etc.). Notice that the use of length() with x returned 5, the number of elements in the vector, whereas the use of length() with my.data returned 2, indicating there are two variables (i.e., columns), x and y, in the matrix. To obtain the dimensions of matrix or data frame, the dim() function can be used.

```
R> dim(my.data)
[1] 5 2
```

Thus, my.data is a 5 (number of rows) by 2 (number of columns) matrix.

Help files for R functions are available with the help() function. For example, if one were interested in the additional arguments that can be used in the mean, help(mean) could be used. Sometimes one might be interested in a function but not know the name of the function. One possibility is to use the search function, where the term(s) to search are given as a text string in quotes. For example, suppose one were interested in a function to obtain the median but

unsure of the function name. The help.search() function could be used as

```
R> help.search("median")
```

which returns information on functions that have "median" in their documentation. Another resource is the R Web site, where search facilities allow for searching across help files and mailing list archives: http://r-project.org.

## Loading Data

R can be used in conjunction with other commonly used statistical (and mathematical) programs, such as Excel, SPSS, and SAS.

### Files in the Format of .txt and .dat

The function to load a data set in the form of .txt or .dat file is read.table(). This function has a rich array of arguments, but the most common specification is of the form

```
read.table(file, header=FALSE, sep=" ")
```

where file is the argument that identifies the file to be loaded into R, header is a logical argument of whether the file's first line contains the names of the variables, and sep denotes the character used to separate the fields (e.g., "*", " ", ", "&", etc.).

For example, consider the following command:

```
R> data1 <- read.table(file="data1.dat",
header=TRUE, sep=" ").
```

This command loads the data file "data1.dat" from the current working directory into R (since a specific file location is not specified) and stores the data into the R object data1. R's working directory is the folder where R reads and stores files; the default position is where R is installed. If the data file to be loaded is not in the current directory, the user also needs to define the file's position, such as

```
R> data2 <- read.table(file="c:/My
Documents/data2.txt", header=FALSE, sep=",").
```

Notice in the first example that the sep argument used a space, whereas a comma was used in the second. This is the case because data1.dat and data2.txt have fields separated with spaces and commas, respectively. Furthermore, R requires the use of "/" or "\\" to signal directory changes, whereas the notation commonly used for folder separation in Microsoft Windows (i.e., "\") is not appropriate in R; this is the case because R has its origins in Unix, which uses the forward slash. Note that the extension name (e.g., .dat, .txt, .R, etc.) of the file should always be specified. Using setwd(), one can set the current working directory to a desired folder, so that one does not need to specify the file's position in the future. To load "data2.txt" in the previous example, instead of defining the file's position, one can use the following commands.

```
R> setwd("C:/My Documents")
R> data2 <- read.table(file="data2.txt",
    header=FALSE, sep=",")
```

It is important to note if the working directory is modified, R, however, sets the default working directory back to where the program is installed whenever R is closed. Because most mainstream statistical and mathematical programs are able to convert data files of their own format into either .dat or .txt format ASCII files, such a conversion and use of the procedures described is always one approach to load data files into R.

### Loading Excel Files

We will use the data set in file salary.xls to illustrate the methods in this section. This data set, which contains the salaries and other information of 62 professors, comes from Cohen, Cohen, West, and Aiken (2003, pp. 81–82). In future sections, we will also use this data set to illustrate graphical techniques and regression analysis in R. To import Excel files into R requires the RODBC (Lapsley & Ripley, 2007) package. RODBC stands for R Open DataBase Connectivity; ODBC is a standard database access method for connecting database applications. By default, this package is not loaded into an R session. To see which packages are currently loaded, the search() function is used, which shows the basic packages that are loaded by default when R is opened:

```
R> search()
[1]  ".GlobalEnv"        "package:stats"     "package:graphics"
[4]  "package:grDevices" "package:utils"     "package:datasets"
[7]  "package:methods"   "Autoloads"         "package:base"
```

When using R in Microsoft Windows, loading a package can be done by selecting the "Packages" tab on the tool bar and then selecting "Load package." A window opens that lists the packages that are installed on the system, which can be selected and loaded. If RODBC is not on the list, it will need to be installed. In Microsoft Windows, select the "Packages" tab on the toolbar and then "Install package(s)," select a server/mirror (generally the closest location), and then choose RODBC. An alternative way to load installed packages is with the library() function, which is illustrated with the RODBC package:

```
R> library(RODBC) .
```

Note that running the library() function without any arguments in the parentheses lists all available packages.

After RODBC is loaded, use odbcConnect Excel() to open an ODBC connection to the Excel database:

```
R>connect <– odbcConnectExcel
("salary.xls").
```

Then function sqlTables() lists the sheets in the Excel file:

```
R> sqlTables(connect)
  TABLE_CAT TABLE_SCHEM TABLE_ NAME
TABLE_TYPE
1 G:\\Program Files\\R\\R–2.4.1\\salary
<NA> salary$ SYSTEM TABLE
```

Notice that the first sheet, whose name is "salary" (in the column TABLE_NAME), is the sheet that contains the data of interest. Therefore, we use sqlFetch() to obtain the data of interest as follows:

```
R> prof.salary <– sqlFetch(connect, "salary")
```

The data set is then loaded into R and stored in the object called prof.salary.

### Loading SPSS Files

The function read.spss(), which is in the *foreign* package (R Development Core Team, 2007a), is used to load an SPSS file into R. For example, after loading the *foreign* package, the following command:

```
R> prof.salary2 <– read.spss(file="salary
.sav")
```

loads salary.sav into R and stores the data set in the object prof.salary2. The file salary.sav contains the same data set as the one in "salary.xls."

### Creating and Loading .R Files

After data are created in R, or data of other formats are imported into R, many times it is desirable to save the data in R format, denoted with a .R file extension, so that loading data can be easily done in a future session. This can be achieved by using the dump() function:

```
R> dump("prof.salary", file="prof.salary.R")
```

which creates a file called "prof.salary.R" that contains the object prof.salary in the current working directory. Alternatively, as before when the data were loaded into R, a particular file location can be specified where the data should be "dumped" (i.e., exported/stored).

Loading data from a .R data file can be done in several ways, the easiest of which is to source the data set into R with the source() command, which runs a selected file. When a file consists of a .R data set, that file is then loaded into R and made available for use. For example, to load the data in file "prof.salary.R," consider the following command:

```
R> source("prof.salary.R")
```

## Graphical Procedures

We will use the professor salary data from Cohen et al. (2003), `prof.salary`, to illustrate some of R's graphical functions.[4] After loading the data, use `names()` to determine the names of the variables.

```
R> names(prof.salary)
[1] "id" "time" "pub" "sex" "citation" "salary"
```

Here, (a) `id` represents the identification number; (b) `time` refers to the time since getting the Ph.D. degree, (c) `pub` refers to the number of publications, (d) `sex` represents gender (1 for female and 0 for male), (e) `citation` represents the citation count, and (f) `salary` is the professor's current salary. To reference a column of the data set (e.g., `pub`), one needs to use the dollar sign "$":

```
R> prof.salary$pub
 18 3 2 17 11 6 38 48 9 22 30 21 10 27 37 8
13 6 12 29 29 7 6 69 11
 ...
```

A more convenient way is to attach the data set to R's search path. Then the user can reference `pub` in `prof.salary` simply with `pub`.

```
R> attach(prof.salary)
R> pub
 18 3 2 17 11 6 38 48 9 22 30 21 10 27 37 8
13 6 12 29 29 7 6 69 11
 ...
```

An attached data set is one where the column names have been "attached," which implies the columns can be directly called upon. At any time, only one data set can be attached. To attach a new data set, one must detach the data set that is currently attached to R. R automatically detaches the data set whenever the user exits the program:

```
R> detach(prof.salary)
R> attach(newdata).
```

### Scatterplot

The function `plot()` can be used to plot data. Although it has a diverse array of arguments, the most common specifications is of the form

```
plot(x, y, type, col, xlim, ylim, xlab, ylab, main),
```

where `x` is the data to be represented on the abscissa (*x*-axis) of the plot; `y` is the data to be represented on the ordinate (*y*-axis; note that the ordering of the values in `x` and `y` must be consistent, meaning that the first element in `y` is linked to the first element in `x`, etc.); `type` is the type of plot (e.g., `p` for points, `l` for lines, `n` for no plotting but setting up the structure of the plot so that points and/or lines are added later); `col` is the color of the points and lines; `xlim` and `ylim` are the ranges of *x*-axis and *y*-axis, respectively; `xlab` and `ylab` are the labels of *x*-axis and *y*-axis, respectively; and `main` is the title of the plot. All of the above arguments, except `x` and `y`, are optional, as R automatically chooses the appropriate settings (usually). For example, to plot the relationship between `salary` and `pub` (without the regression line), the following can be used:

```
R> plot(x=pub, y=salary, xlim=c(0, 80),
xlab="Number of Publications",
ylab="Professor's Salary")
```

Note in the application of the `plot()` function that the range of the *x*-axis is defined by `c()`, which is a function to generate vectors by combining the terms (usually numbers). A regression line or a smoothed regression line can be added to the scatterplot if desired. The smoothed regression line fits a regression model to a set of points in a certain "neighborhood," or locally. Such a technique is called *lowess* (or *loess*; see Cleveland, 1979, 1981, for a detailed discussion of smoothed locally weighted regression lines). Adding such a smoothed regression line to a plot can be done as follows using the `lines()` function combined with the `lowess()` function:

```
R> lines(lowess(x=pub, y=salary, f=.8)).
```

The `lines()` function is used to draw lines or line segments, whose basic specification is of the form `lines(x, y)`, where the arguments are the same as

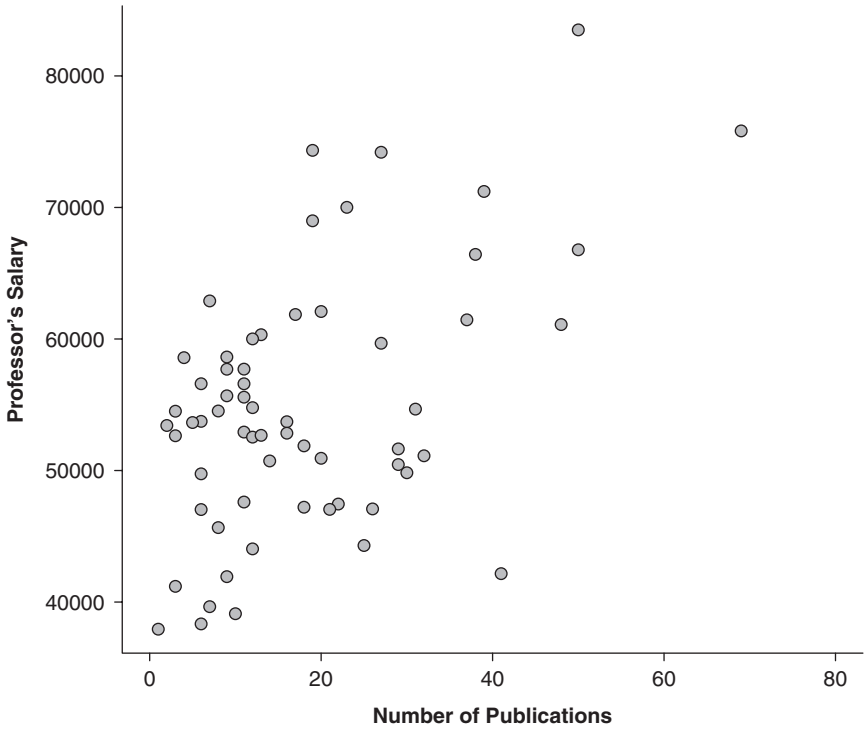


**Figure 34.1** Scatterplot for professor's salary as a function of the number of publications.

those in plot(). The lowess() function has f as a smoothing span that defines the width of the neighborhood in which points are treated locally, with a larger f representing a larger "neighborhood," which then gives more smoothness.

The function locator() helps to determine which point in a plot corresponds with which individual in the data. It can be used, for example, to identify outliers and miscoded data. After a scatterplot has been plotted,

```
R> locator()
```

turns the mouse pointer into a cross for point identification by selecting a specific point.

### Matrix Plot

The function pairs(), whose arguments are all the same as those of plot(), can be used to produce scatterplot matrices. For example,

```
R> pairs(prof.salary[-1])
```

plots all the variables except the first one (i.e., id) in prof.salary. When the user is interested in only a few variables in a data set, one possibility is to create a new object with only those variables. For example, suppose one is interested in only pub, citation, and salary and does not want all five variables in the matrix plot.

```
R> pub.cit.sal <- data.frame(pub, citation,
    salary)
R> pairs(pub.cit.sal)
```

### Histogram

The function to plot histograms is hist(). The basic specification is of the form

```
hist(x, breaks, freq)
```

where x is the data to be plotted, breaks defines the way to determine the location and/or quantity of bins, and freq is a logical statement of whether the histogram represents frequencies

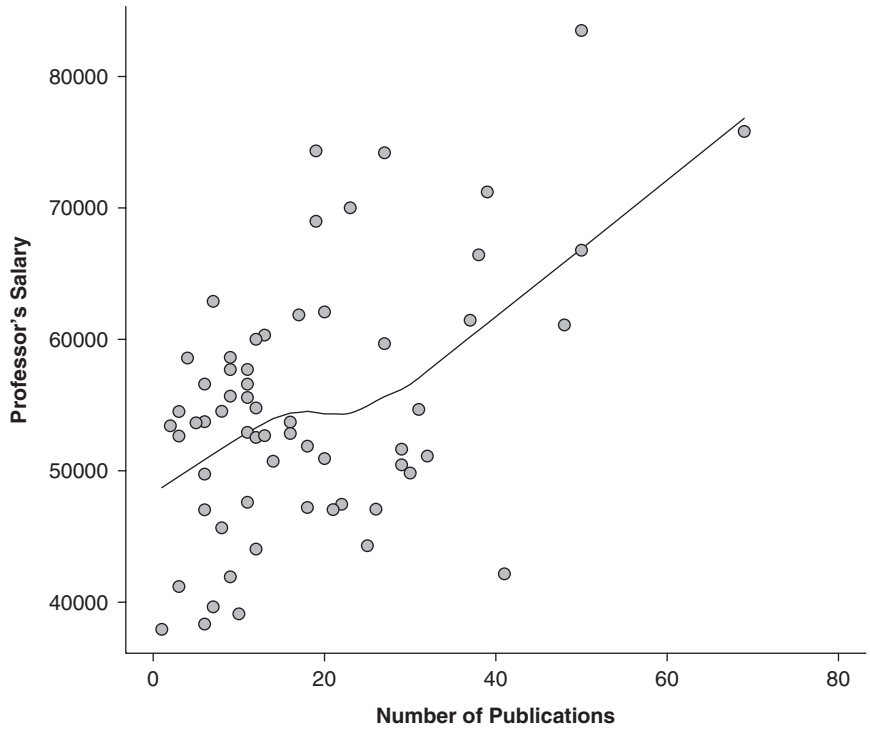


**Figure 34.2** A smoothed regression line is added on the scatterplot for professor's salary as a function of the number of publications.

(freq=TRUE) or probability densities (freq=FALSE). For example, to plot histograms of pub, consider the following commands:

```
R> par(mfrow=c(2,2))
R> hist(pub, main="1st")
R> hist(pub, freq=FALSE, main="2nd")
R> hist(pub, freq=FALSE, breaks=10,
    main= "3rd")
R> hist(pub, freq=FALSE, breaks=seq
    (from=0, to=75, by=6), main="4th")
R> lines(density(pub, bw=3))
```

The function par() is used to modify graphical parameters, and it has a rich array of arguments to control line styles, colors, figure arrangement, titles and legends, and much more. With the function par(), the user can customize nearly every aspect of the graphical display. Moreover, all of the arguments in par() can be included in, and thus control, other graphical functions, such as hist() and lines(); put another way, a uniform set of parameters controls all graphical functions and all aspects of figure presentation. The argument mfrow in par() is used to arrange multiple figures on the same page. If mfrow is defined as mfrow=c(m,n), then figures will be arranged into an m-row-by-n-column array.

If breaks in hist() is defined by a single number, then the number is considered by R as the number of bins. In order to define the range of a single bin, the user needs to use a vector giving the breakpoints between the cells. The function to generate such vectors is seq(), whose basic specification is of the form

```
seq(from, to, by)
```

where from and to are the starting and end values of the sequence, respectively, and by is the increment of the sequence. Thus, the fourth histogram bins the data every 6 units. When included as an argument in lines(), the function density() can be used to add a smoothed density line to the histogram. In density(), bw is the smoothing bandwidth to be used, analogous to the span in lowess()—the larger the bw, the smoother the density line. Only when the vertical axis represents the probability can the probability density curve be drawn on the histogram.

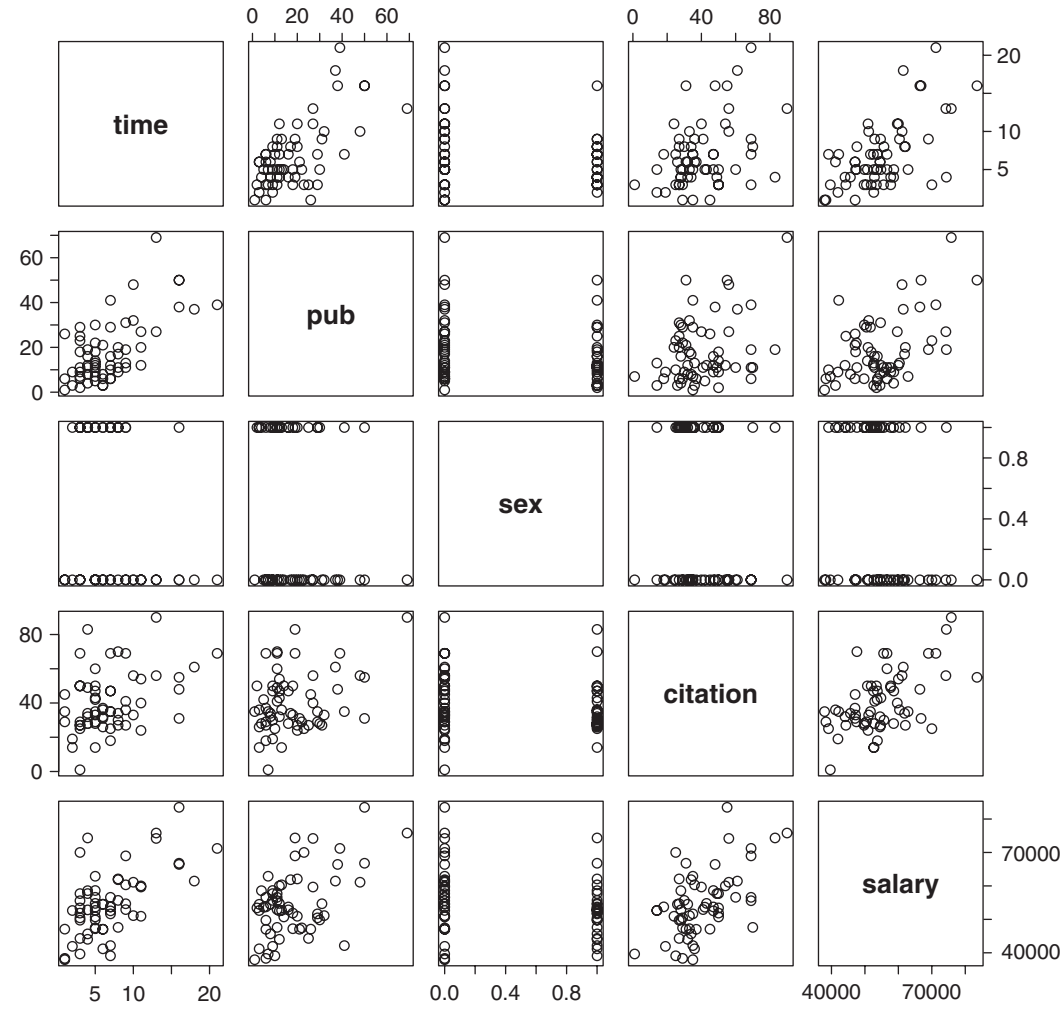


**Figure 34.3** Scatterplot matrix for the time since getting the Ph.D. degree, the number of publications, gender, the number of citations, and the current salary.

### QQ Plot

The graphical functions to visually inspect for normality are `qqnorm()`, `qqplot()`, and `qqline()`. The function `qqnorm()` plots the sample quantiles against the theoretical quantiles from a normal distribution. The function `qqline()` adds a line to the current QQ plot, indicating where the observed values are expected given a normal distribution. The function `qqplot()` is used to examine the relationship between two variables. Their basic specifications are

```
qqnorm(y)
qqline(y)
qqplot(x, y)
```

where `x` and `y` are data to be represented on the *x*-axis and *y*-axis, respectively. Suppose we want to examine the normality of `pub` and the relationship between `pub` and `salary`. Because standardized scores are generally preferred in QQ plots, we first standardize `pub` and `salary`. The function `mean()` calculates the mean of a set of data and `sd()` the standard deviation.

```
R> std.pub <- (pub - mean(pub)) / sd(pub)
R> std.salary <- (salary - mean(salary)) /
    sd(salary)
R> qqnorm(std.pub)
R> qqline(std.pub)
R> qqplot(std.pub, std.salary,
    xlab="Standardized Publications",
```

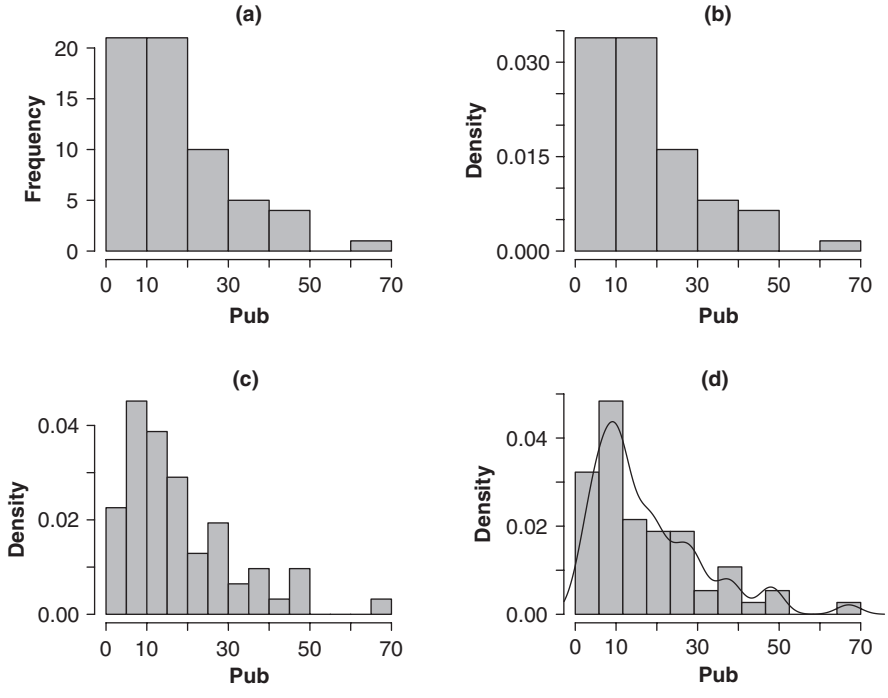


**Figure 34.4** Histograms with different specifications for the number of publications.

```
ylab="Standardized Salary")
```

Another way to standardize, which is much simpler, is to use `scale()`, whose common specification is of the form

```
scale(x)
```

where `x` is the data to be standardized. Therefore, instead of using `mean()` and `sd()`, the following commands produce the same QQ plot as Figure 34.5.

```
R> scale.pub <- scale(pub)
R> qqnorm(scale.pub)
R> qqline(scale.pub)
```

## Multiple Regression

In this section, we will continue to use the professor salary data set from Cohen et al. (2003), `prof.salary`, to illustrate how to conduct multiple regression analysis with R.

### Fitting Regression Models

R uses ~, +, and −, along with a response variable(s) and *K* predictor variables, to define a particular model's equation. The generic formula is of the form

```
response variable ~ predictor1 + (or −)
predictor2 . . . + (or −) predictorK
```

where + signals inclusion of the predictor, and – signals exclusions of the predictor. The minus sign may seem meaningless when defining a new formula, but it is useful in removing predictors from a currently existing model in the context of model modifications.

The function for fitting a linear model is the linear model function, `lm()`, whose basic specification is in the form of

```
lm(formula, data)
```

where `formula` is a symbolic description of the model to be fitted (just discussed), and `data` identifies the particular data set of interest. For example, to study the regression of professors' salaries ($Y$) conditional on publications ($X_1$) and citations ($X_2$), we use the following syntax to fit the model and obtain the model summary:

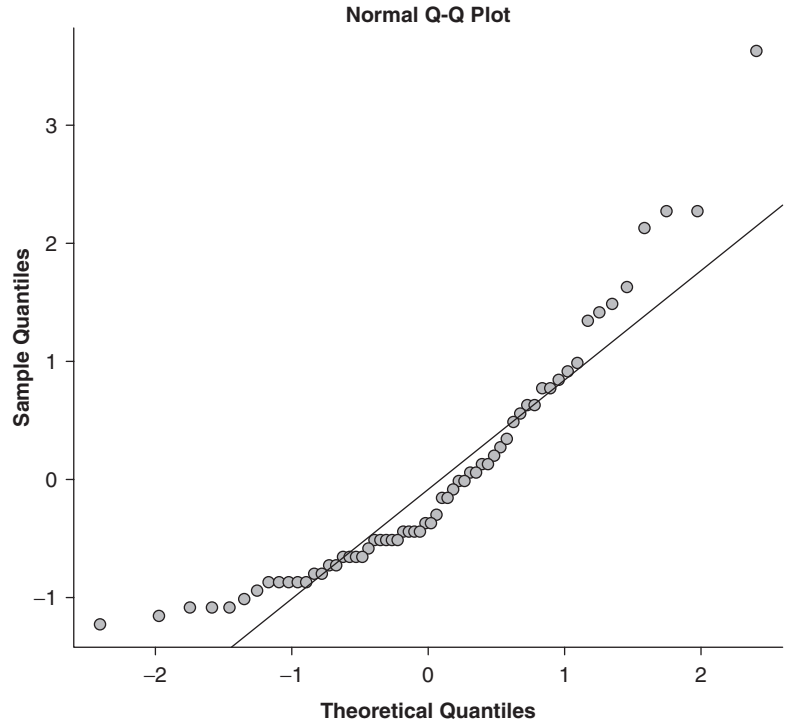


**Figure 34.5** QQ plot for the number of publications with the equiangular line indicating the expected publications given the normal distribution as a reference.

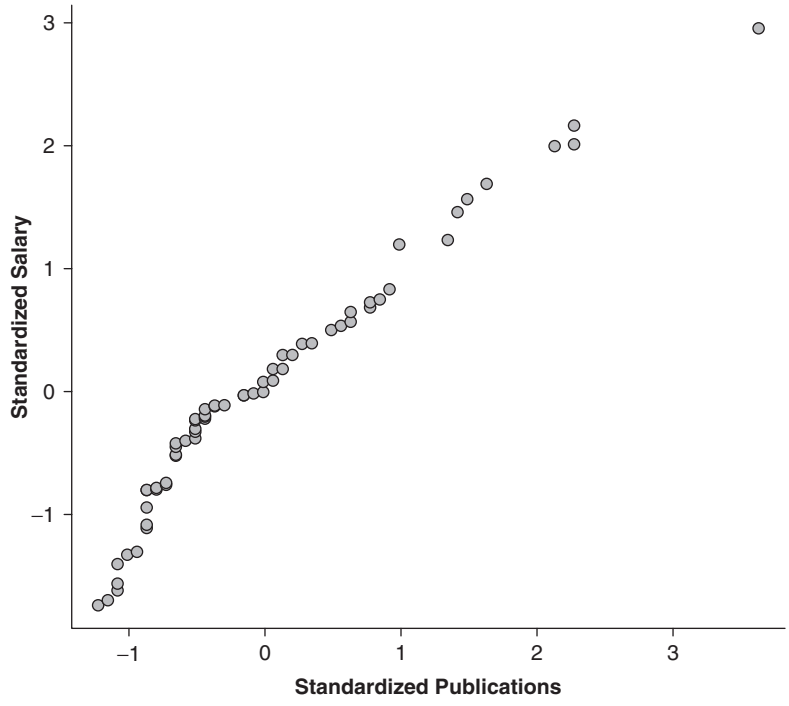


**Figure 34.6** QQ plot for standardized professor's current salary and standardized number of publications.

```
R> model1 <— lm(salary ~ pub + citation, data=prof.salary)
R> summary(model1)
Call:
lm(formula=salary ~ pub + citation, data=prof.salary)
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 *** |
| pub | 251.75 | 72.92 | 3.452 | 0.001034 ** |
| citation | 242.30 | 59.47 | 4.074 | 0.000140 *** |

—

```
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
```

The fitted regression model is thus

$$\hat{Y} = 40493 + 251.8X_1 + 242.3X_2, \qquad (1)$$

with the model's squared multiple correlation coefficient being 0.4195. In later sections, we will discuss forming confidence intervals for the population regression coefficients and for the population squared multiple correlation coefficient.

After fitting the model, the residuals can be plotted for visual inspection of the quality of fit:

```
R> par(mfcol=c(2,2))
R> plot(model1)
```

The anova() function can be used to obtain a table of the sums of squares (i.e., an ANOVA table) for the fitted model:

```
R> anova(model1)
Analysis of Variance Table
Response: salary
```

| | Df | Sum Sq | Mean Sq | F value | Pr(>F) |
|---|---|---|---|---|---|
| pub | 1 | 1472195326 | 1472195326 | 26.038 | 3.743e-06 *** |
| citation | 1 | 938602110 | 938602110 | 16.601 | 0.0001396 *** |
| Residuals | 59 | 3335822387 | 56539362 | | |

—

```
Signif. codes: 0'***' 0.001'**' 0.01'*' 0.05'.' 0.1' ' 1
```

The object model1 contains rich information about the fitted regression model. To see the available objects, the function names() can be used:

```
R> names(model1)
[1]  "coefficients"  "residuals"  "effects"  "rank"
[5]  "fitted.values"  "assign"  "qr"  "df.residual"
[9]  "xlevels"  "call"  "terms"  "model"
```

For example, suppose one wants to check whether there is systematic relationship between the residuals and the predictors. The following commands should be considered (recall how we extracted pub from prof.salary with the sign"$"):

```
R>  par(mfrow=c(2,2))
R>  plot(pub, model1$residuals,
      main="Residuals vs Predictor 1")
R>  plot(citation, model1$residuals,
      main="Residuals vs Predictor 2")
```

### Model Comparison

The function update() is a convenient function for situations where the user needs to fit a model that only differs from a previously fitted model in a nested form. Its basic form is

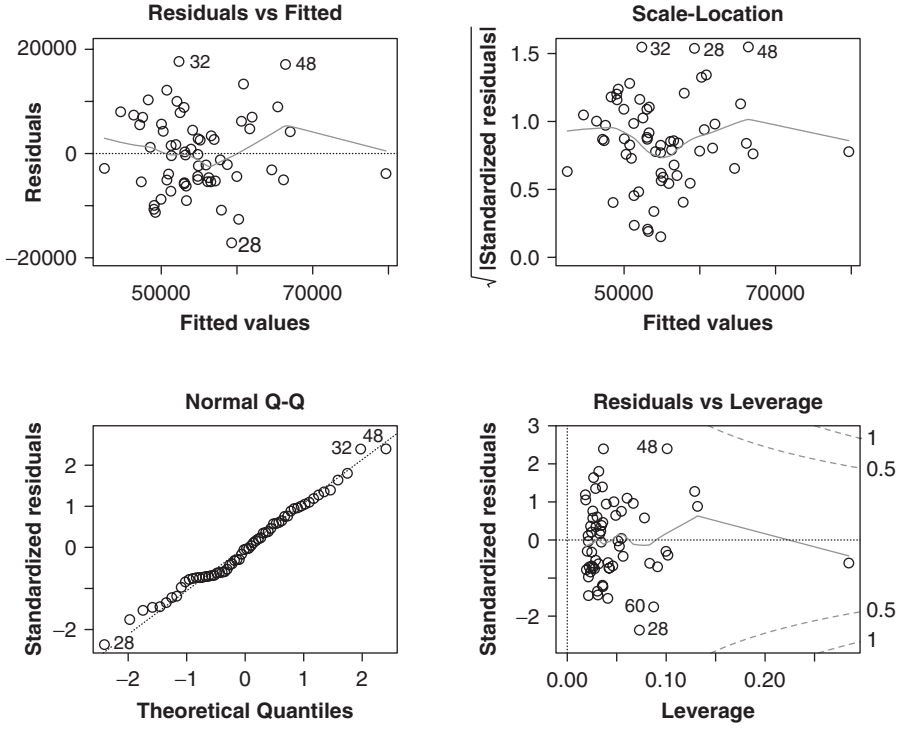


**Figure 34.7** Scatterplot for residuals as a function of fitted values with a smoothed regression line (top left), scatterplot for standardized residuals as a function of fitted values (top right), QQ plot for standardized residuals (bottom left), and scatterplot for standardized residuals as a function of leverage (bottom right), all of which are produced by plotting the fitted linear regression model object.

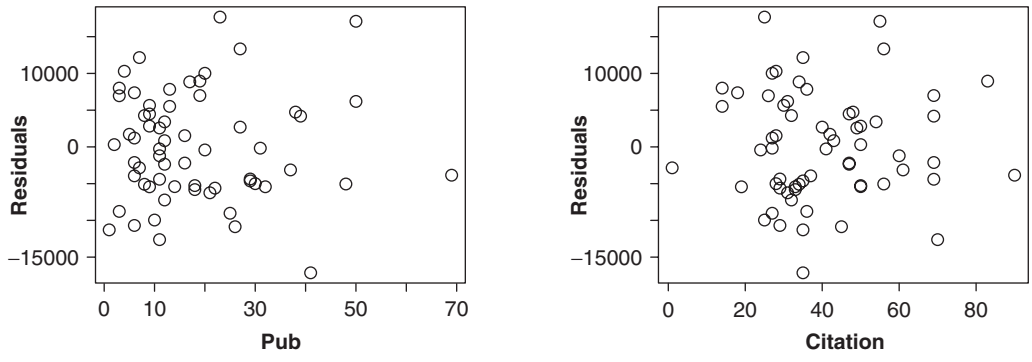


**Figure 34.8** Scatterplots for residuals as a function of the number of publications (on the left) and for residuals as a function of the number of citations (on the right).

```
update(object, formula)
```

where object is the originally fitted model, and formula is the new model to be calculated. Also, in defining the new formula, the period (i.e., ".") denotes the corresponding part of the old model formula.

For example, suppose there was interest in adding the predictor time ($X_3$) and sex ($X_4$) to the previous model:

```
R> model2 <— update(model1, . ~ . + time + sex)
R> summary(model2)
Call:
lm(formula=salary ~ pub + citation + time + sex, data=prof.salary)
Residuals:
```

| Min | 1Q | Median | 3Q | Max |
|---|---|---|---|---|
| -13376.8 | -4482.5 | -989.7 | 4316.2 | 20671.2 |

Coefficients:

| | Estimate | Std. Error | t value | Pr(>\|t\|) |
|---|---|---|---|---|
| (Intercept) | 39587.35 | 2717.48 | 14.568 | < 2e-16 *** |
| pub | 92.75 | 85.93 | 1.079 | 0.28498 |
| citation | 201.93 | 57.51 | 3.511 | 0.00088 *** |
| time | 857.01 | 287.95 | 2.976 | 0.00428 ** |
| sex | -917.77 | 1859.94 | -0.493 | 0.62360 |

—

```
Signif. codes: 0'***' 0.001'**' 0.01'*' 0.05'.' 0.1' ' 1
Residual standard error: 7077 on 57 degrees of freedom
Multiple R-Squared: 0.5032, Adjusted R-squared: 0.4684
F-statistic: 14.44 on 4 and 57 DF, p-value: 3.357e-08
```

Given the specifications above, the new model obtained is

$$\hat{Y} = 39587.35 + 92.75X_1 + 201.93X_2 + 857.01X_3 - 917.77X_4, \quad (2)$$

with the squared multiple correlation coefficient increasing from 0.4195 in the previous model to 0.5032 in the present model.

To compare the full model (i.e., the one with four predictors) with the reduced one (i.e., the one with two predictors), anova() can be used, which evaluates if the sum of squares accounted for by the additional two predictors in the full model leads to a significant decrease in the proportion of variance in *Y* that was previously unaccounted for in the reduced model. Interested readers may refer to Cohen et al. (2003) or Maxwell and Delaney (2004) for a discussion of model comparisons:

```
R> anova(model1, model2)
Analysis of Variance Table
Model 1: salary ~ pub + citation
Model 2: salary ~ pub + citation + time + sex
  Res.Df       RSS  Df  Sum of Sq      F    Pr(>F)
1     59 3335822387
2     57 2854659884   2  481162503 4.8038 0.01180 *
—
Signif. codes: 0'***' 0.001'**' 0.01'*' 0.05'.' 0.1' ' 1
```

Notice that with the additional two variables, a significant reduction ($p < .05$) in the unaccounted for variance was achieved.

### Interaction Plots

A general expression for a regression equation containing a two-way interaction is

$$\hat{Y} = \beta_0 + \beta_1 X + \beta_2 Z + \beta_3 X Z, \quad (3)$$

where $\beta_0$ is the intercept; $\beta_1$ and $\beta_2$ are the regression coefficients of the main effects of *X* and *Z*, respectively; and $\beta_3$ is the regression coefficient for the interaction between *X* and *Z*. Many theories in the social sciences hypothesize that variables interact (there are moderators), and thus the idea of testing interactions is fundamental in many areas of the BESS (Cohen et al., 2003; Aiken & West, 1991). The MBESS R package contains functions to plot two- and three-dimensional interaction plots.

The function intr.plot() in MBESS plots a three-dimensional representation of a multiple regression surface containing one two-way interaction. The most common specification of this function is in the form

```
intr.plot(b.0, b.x, b.z, b.xz, x.min, x.max,
z.min, z.max, hor.angle, vert.angle)
```

where b.0, b.x, b.z, b.xz are the estimates of $\beta_0$, $\beta_1$, $\beta_2$, $\beta_3$ in Equation 3, respectively; x.min, x.max, z.min, and z.max define the minimum and maximum values of *X* and *Z* of interest, respectively; hor.angle is the horizontal viewing angle; and vert.angle is the vertical viewing angle.

Cohen et al. (2003, pp. 257–263) provide an example for a regression containing one two-way interaction, whose model equation is

$$\hat{Y} = 2 + 0.2X + 0.6Z + 0.4XZ, \quad (4)$$

with **X** being [0, 2, 4, 6, 8, 10] and **Z** being [0, 2, 4, 6, 8, 10]. To replicate this example, intr.plot() can be defined as follows:

```
R> par(mfrow=c(2,2))
R> intr.plot(b.0=2, b.x=.2, b.z=.6, b.xz=.4, x.min=0, x.max=10, z.min=0, z.max=10)
R> intr.plot(b.0=2, b.x=.2, b.z=.6, b.xz=.4, x.min=0, x.max=10, z.min=0, z.max=10, hor.angle=-65,
   vert.angle=15)
R> intr.plot(b.0=2, b.x=.2, b.z=.6, b.xz=.4, x.min=0, x.max=10, z.min=0, z.max=10, hor.angle=-65,
   vert.angle=5)
R> intr.plot(b.0=2, b.x=.2, b.z=.6, b.xz=.4, x.min=0, x.max=10, z.min=0, z.max=10, hor.angle=45)
```

The function intr.plot.2d() in MBESS is used to plot regression lines for one two-way interaction, holding one of the predictors (in this function, *Z*) at values –2, –1, 0, 1, and 2 standard deviations above the mean. The most common specification of intr.plot.2d() is of the form

```
intr.plot.2d(b.0, b.x, b.z, b.xz, x.min, x.max, mean.z, sd.z)
```

where b.0, b.x, b.z, b.xz, x.min, x.max have the same meaning as those in intr.plot(), and mean.z and sd.z are the mean and standard deviation of *Z*, respectively.

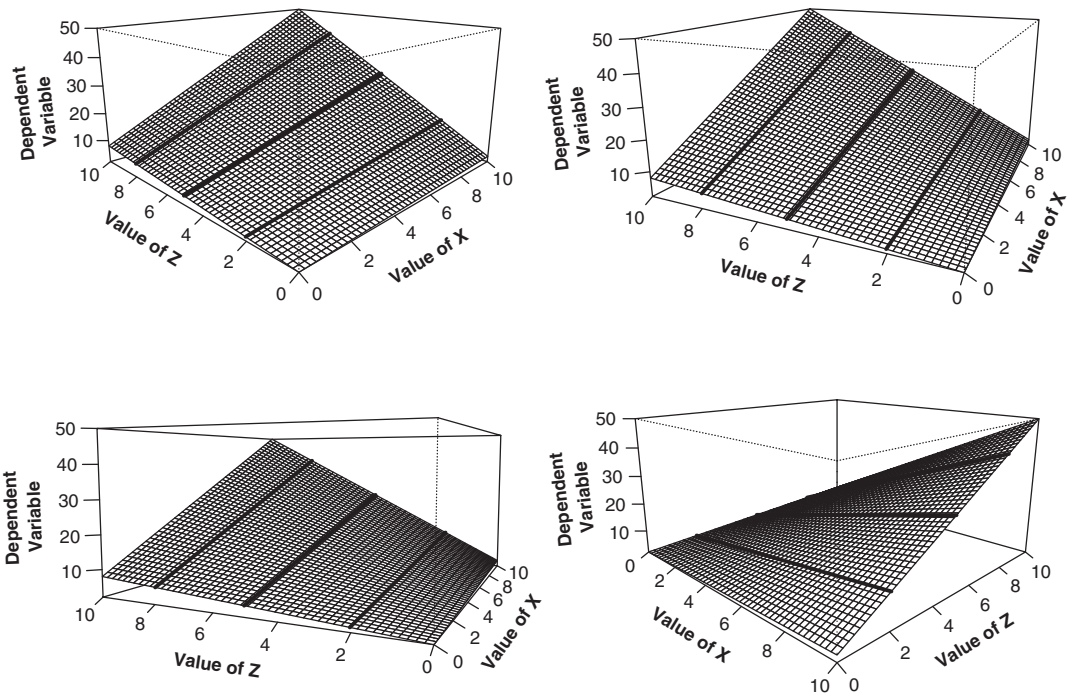


**Figure 34.9** Regression surface for $\hat{Y} = 2 + 0.2X + 0.6Z, + 0.4XZ$ displayed from –45° horizontal angle and 15° vertical angle (upper left), –65° horizontal angle and 15° vertical angle (upper right), –65° horizontal angle and 5° vertical angle (lower left), and 45° horizontal angle and 15° vertical angle (lower right). The three bold lines on the regression surface are regression lines holding Z constant at –1, 0, and 1 standard deviations from *Z*'s mean.

Cohen et al. (2003, pp. 263–268) give an example for the regression lines of

$$\hat{Y} = 16 + 2.2X + 2.6Z + 0.4XZ, \qquad (5)$$

holding $Z$ constant at values –1, 0, and 1 standard deviations above the mean, when $X \in [0, 50]$, the mean of $Z$ is 0, and the standard deviation of $Z$ is 1. We can replicate and extend this example by specifying `intr.plot.2d()` as follows.

```
R> intr.plot.2d(b.0=16, b.x=2.2, b.z=2.6,
b.xz=.4, x.min=0, x.max=50, mean.z=0, sd.z=1)
```

### Confidence Intervals for Regression Parameters

Forming confidence intervals for standardized effect sizes is quite involved because such intervals require the use of noncentral distributions (Kelley, 2007a; Smithson, 2003; Steiger, 2004; Steiger & Fouladi, 1997). Linking the confidence intervals for a statistic of interest and noncentral distributions is achieved with the *confidence interval transformation principle* and the *inversion confidence interval principle,* as discussed in Steiger and Fouladi (1997) and Steiger (2004). Although methods to construct confidence intervals for the population squared multiple correlation coefficient (e.g., Algina & Olejnik, 2000; Smithson, 2003) and for the population standardized regression coefficients (e.g., Kelley, 2007a; Kelley & Maxwell, 2003, in press) have been developed, no mainstream statistical packages besides R with MBESS can perform such tasks without using special programming scripts. Although such confidence intervals are difficult to obtain, they are important nonetheless. The benefits of confidence intervals for standardized effect sizes are the focus of Thompson (Chapter 18, this volume). MBESS has a powerful set of functions that

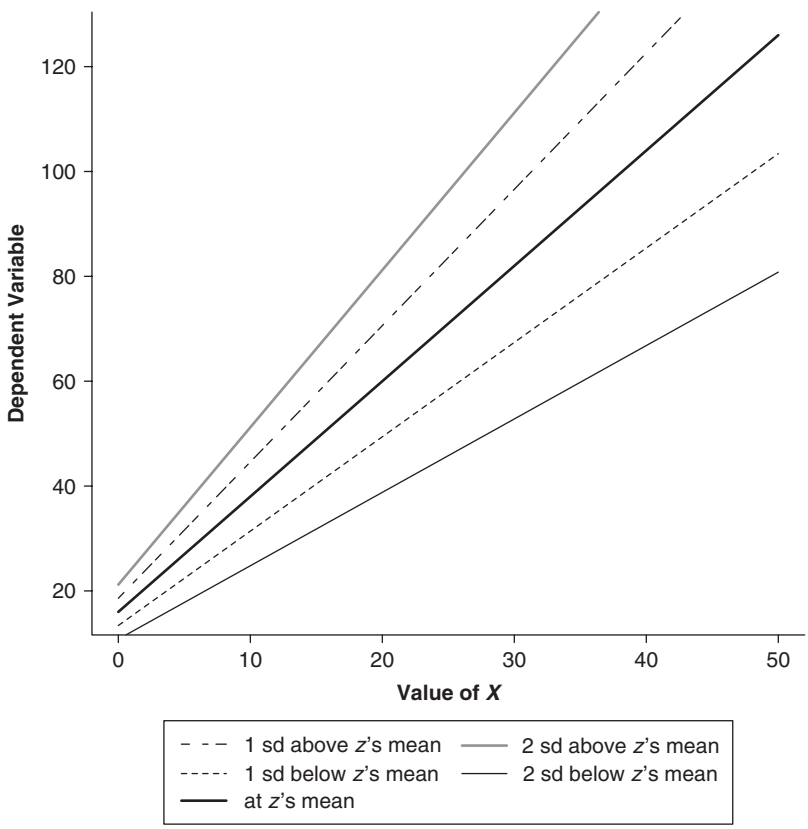


**Figure 34.10** Regression lines for $\hat{Y} = 16 + 2.2X + 2.6Z, + 0.4XZ,$ holding $Z,$ whose mean is 0 and standard deviation is 1, constant at values –2, –1, 0, 1, and 2 standard deviations from the mean.

implements confidence intervals for noncentral *t*, *F*, and chi-square parameters. These functions (conf.limits.nct(), conf.limts.ncf(), and conf.limits.nc.chi.square()) return the confidence interval for noncentrality parameters, which are then used in other MBESS functions to implement confidence intervals for specific effect sizes that are commonly used in the BESS.

### Confidence Intervals for Omnibus Effects

The sample squared multiple correlation coefficient, denoted $R^2$, often termed the *coefficient of multiple determination,* is defined as

$$R^2 = \frac{SS_{\text{regression}}}{SS_{\text{total}}}. \quad (6)$$

MBESS includes the function ci.R2() to form the exact confidence intervals for the population squared multiple correlation coefficient in the context of fixed (e.g., Smithson, 2003; Steiger, 2004) or random regressors (Algina & Olejnik, 2000; Ding, 1996; Lee, 1971; Steiger & Fouladi, 1992). In almost all applications of multiple regression in the behavioral, educational, and social sciences, regressors are random. When the predictors are random, a basic specification of the function is of the form

```
ci.R2(R2, N, K, conf.level=.95)
```

where R2 is the observed (i.e., sample) squared multiple correlation coefficient, conf.level is the desired confidence interval coverage, N is the sample size, and K is the number of predictors. In the case of fixed regressors, the statement

```
Random.Regressors=FALSE
```

should be included in the ci.R2 function.

For example, to form the 95% exact confidence interval for $P^2$, the population squared multiple correlation coefficient, of model2, where the predictors are regarded as random, ci.R2() is specified as follows:

```
R> ci.R2(R2=0.5032, N=62, K=4)
$Lower.Conf.Limit.R2
[1] 0.2730107
$Prob.Less.Lower
[1] 0.025
$Upper.Conf.Limit.R2
[1] 0.6420285
$Prob.Greater.Upper
[1] 0.025
```

Recall that the observed squared multiple correlation coefficient can be obtained from the function summary(). Therefore, the 95% confidence interval for the population squared multiple correlation coefficient of model2, when the predictors are considered random, is

$$\text{CI}_{.95} = [0.273 \leq P^2 \leq 0.642],$$

where $\text{CI}_{.95}$ represents a 95% confidence interval.

The function ci.R() is used to obtain the confidence interval for the population multiple correlation coefficient (i.e., P). The most common specification of this function is of the form

```
ci.R(R, N, K, conf.level=.95)
```

where R is the observed multiple correlation coefficient, and other arguments are the same as those in ci.R2(). This function also by default considers the predictors random; when the predictors are fixed, the user can include Random.Regressors=FALSE in the argument.

Because ci.R() and ci.R2() require only the observed multiple correlation coefficient or its square, respectively, and the degrees of freedom, these functions can also be used to form confidence intervals for effects reported in published articles or multiple regression models that were fitted in other programs that do not have the capabilities available in R to implement the confidence intervals.

### Confidence Intervals for Targeted Effects

The function confint() is used to construct confidence intervals for unstandardized regression coefficients from a fitted linear model. Its basic specification is of the form

```
confint(object, parm, level=.95)
```

where object is the fitted linear model, parm is the parameter whose confidence intervals are to be formed, and level is the desired confidence level; if parm is not specified, confidence intervals for all

regression coefficients will be computed. To obtain a 90% confidence interval for pub in model2, the function confint() is specified as follows:

```
R> confint(model2, "pub", level=.90)
        5%          95%
pub  −50.92814   236.4208
```

Therefore, the 90% confidence interval for the population unstandardized regression coefficient of pub in model2 (i.e., $B_1$) is $CI_{.90} = [-50.93 \leq B_1 \leq 236.42]$, where $CI_{.90}$ represents a confidence interval at the subscripted level.

However, confint() can only be used to form confidence intervals for unstandardized regression coefficients and always returns a two-tailed confidence interval. Another function to obtain confidence intervals for targeted regression coefficients is ci.rc(), which is contained in the MBESS package. The basic specification of ci.rc() is of the form

```
ci.rc(b.k, s.Y, s.X, N, K, R2.Y_X, R2.k_X.without.k, conf.level=.95)
```

where b.k is the value of the regression coefficient for the *k*th regressor of interest (i.e., $X_k$), s.Y is the standard deviation of the response variable *Y*, s.X is the standard deviation of $X_k$, N is sample size, K is the total number of regressors, R2.Y_X is the squared multiple correlation coefficient predicting *Y* from all the predictors, R2.k_X.without.k is the squared multiple correlation coefficient predicting $X_k$ from the remaining $K - 1$ predictors, and conf.level is the desired confidence interval coverage.[5] Unlike confint(), which is based on R's fitted linear model objects, ci.rc() requires only summary statistics, and thus the output from other statistical programs and published articles can be used as input.

For example, to obtain the 90% confidence interval for the population regression coefficient of pub in model2, consider the following steps:

```
R>model.pub <− update(model2, pub ~ . − pub)
R> summary(model.pub)
Call:
. . .
Residuals:
. . .
Coefficients:
. . .
Multiple R-Squared: 0.433, Adjusted R-squared: 0.4037
. . .
R> ci.rc(b.k=92.75, s.Y=sd(salary), s.X=sd(pub), N=62, K=4, R2.Y_X=0.5032, R2.k_X.without.k=0.433, conf.level=.90)
$Lower.Limit.for.beta.k
[1] −50.92821
$Prob.Less.Lower
[1] 0.05
$Upper.Limit.for.beta.k
[1] 236.4282
$Prob.Greater.Upper
[1] 0.05
```

Therefore, the 90% confidence interval for the population unstandardized regression coefficient of pub in model2 (i.e., $B_1$), computed by ci.rc(), is

$$CI_{.90} = [-50.93 \leq B_1 \leq 236.42],$$

which is the same as what confint() returned previously. Moreover, with some additional arguments (namely, alpha.lower and alpha .upper), ci.rc() is also able to form one-tailed confidence intervals or other nonsymmetric confidence intervals, which is not possible with confint(). Note that R2.Y_X is obtained from model2's summary table (the model with all predictors), and R2.k_X.without.k is obtained from model.pub's summary table (the model without the predictor of interest).

The function ci.src() in MBESS is used to form confidence intervals for the population standardized regression coefficient. A basic specification of this function is of the form

```
ci.src(beta.k, SE.beta.k, N, K, conf.level=0.95)
```

where beta.k is the standardized regression coefficient of the *k*th predictor (i.e., the one of interest), SE.beta.k is the standard error of the *k*th regression coefficient, and N, K, and conf.level are the same as those in ci.rc().

For example, suppose we want to obtain the 95% confidence interval for the population standardized regression coefficient of pub in model2. Because the data used in model2 are unstandardized and beta.k in ci.src() requires standardized ones, we first need to standardize the fitted model's regression coefficients. When the user inputs standardized data to fit the linear model, the regression coefficients returned are already standardized.

```
R> std.pub <— scale(pub)
R> std.time <— scale(time)
R> std.citation <— scale(citation)
R> std.sex <— scale(sex)
R> std.model2 <— lm(std.salary ~ std.pub + std.time + std.citation + std.sex)
R> summary(std.model2)
Call:
. . .
Residuals:
. . .
Coefficients:
                    Estimate          Std. Error          t value          Pr(>|t|)
. . .
std.pub             1.338e-01         1.240e-01           1.079            0.28498
. . .
R> ci.src(beta.k=0.1338, SE.beta.k=0.124, N=62, K=4)
$Lower.Limit.for.beta.k
[1] -0.1110479
$Prob.Less.Lower
[1] 0.025
$Upper.Limit.for.beta.k
[1] 0.3774881
$Prob.Greater.Upper
[1] 0.025
```

Thus, the 95% confidence interval for the population standardized regression coefficient of pub in model2 (i.e., $\beta_1$) is $CI_{.95} = [-0.111 \leq \beta_1 \leq 0.377]$.

### Sample Size Planning in Multiple Regression

#### *Sample Size Planning for the Omnibus Effect: Power Analysis*

Cohen (1988) discussed methods of sample size planning for the test of the null hypothesis that $P^2 = 0$, where $P^2$ is the population squared multiple correlation coefficient. Cohen provided an extensive set of tables for sample size determination for a large but limited set of conditions. Those methods, and related ones, have been implemented in MBESS so that researchers can plan sample size for a desired power for the omnibus effect in multiple regression.

The function ss.power.R2() in MBESS can be used to plan sample size so that the test of the squared multiple correlation coefficient has sufficient power. Its basic specification is of the form

```
ss.power.R2(Population.R2, alpha.level=0.05, desired.power=0.85, K)
```

where Population.R2 is the population squared multiple correlation coefficient, alpha.level is the Type I error rate, desired .power is the desired power, and K is the number of predictors.

For example, to obtain the necessary sample size when the population multiple correlation coefficient is believed to be .25, Type I error rate is set to .05, the desired power is .85, and the regression model includes four predictors, ss.power.R2() would be specified as

```
R> ss.power.R2(Population.R2=.25, alpha.level=0.05, desired.power=0.85, K=4)
$Necessary.Sample.Size
[1] 46
$Actual.Power
[1] 0.8569869
$Noncentral.F.Parm
[1] 15.33333
```

```
$Effect.Size
[1] 0.3333333
```

Thus, the necessary sample size is 46.

*Sample Size Planning for the Omnibus Effect: Accuracy in Parameter Estimation (AIPE)*

The sample size for multiple regression can be planned in such manner that the confidence interval for the population squared multiple correlation coefficient is sufficiently narrow; "sufficiently narrow" is something defined by researchers depending on the particular situation, much like the desired level of power. This approach to sample size planning is termed *accuracy in parameter estimation* (AIPE; Kelley & Maxwell, 2003; Kelley, Maxwell, & Rausch, 2003; Kelley & Rausch, 2006) because the goal of such an approach is to obtain an accurate parameter estimates. Interested readers may refer to Kelley and Maxwell (2003, in press) and Kelley (2007b, 2007c) for a discussion of AIPE for omnibus effects in multiple regression.

The function ss.aipe.R2() in MBESS can be used to determine necessary sample size for the multiple correlation coefficient so that the confidence interval for the population multiple correlation coefficient is sufficiently narrow. Its basic specification is of the form

```
ss.aipe.R2(Population.R2, conf.level=.95,
width, Random.Regressors, K, verify.ss=FALSE)
```

where width is the width of the confidence interval, Random.Regressors is a logical statement of whether the predictors are random (TRUE) or fixed (FALSE), conf.level is the confidence interval coverage, Population.R2 and K are the same as those arguments in ss.power.R2(), and verify.ss is a logical statement of whether the user requires the exact sample size (verify.ss=TRUE ), which involves a somewhat time-consuming set of intense calculations (specifically an a priori Monte Carlo simulation), or a close approximation (verify.ss=FALSE).

For example, suppose the population squared multiple correlation coefficient is believed to be .5, the confidence level is set to .95, and the regression model includes five random predictors. If one wishes to obtain the exact necessary sample size so that the expected full confidence interval width is .25, ss.aipe.R2() would be specified as

```
R> ss.aipe.R2(Population.R2=.5, width=.25,
K=5, conf.level=.95, verify.ss=TRUE)
   $Required.Sample.Size
   [1] 125
```

An additional specification in ss.aipe.R2() allows for a probabilistic component that the confidence interval obtained in a study will be sufficiently narrow with some desired degree of probability (i.e., assurance), which is accomplished with the additional argument assurance. For example, suppose one wishes to have 99% assurance that the 95% confidence interval will be no wider than .25 units. The ss.aipe.R2() function would be specified as

```
R> ss.aipe.R2(Population.R2=.5, width=.25,
K=5, conf.level=.95, assurance=.99,
verify.ss=TRUE)
   $Required.Sample.Size
   [1] 145
```

*Sample Size Planning for Targeted Effects: Power for Regression Coefficients*

Cohen (1988) and Maxwell (2000) develop methods to plan the necessary sample size so that the hypothesis test of a targeted regressor has a sufficient degree of statistical power to reject the null hypothesis that the regressor is zero in the population. Those methods have been implemented in MBESS with the ss.power.rc() function, which returns the necessary sample size from the power approach for a targeted regression coefficient. Its basic specification is of the form

```
ss.power.rc(Rho2.Y_X, Rho2.Y_X.without.k,
K, desired.power=0.85, alpha.level=0.05)
```

where Rho2.Y_X is the population squared multiple correlation coefficient, Rho2.Y_X.without.k is the population squared multiple correlation coefficient predicting the response predictor variable from the remaining $K - 1$ predictors, K is the total number of predictors, desired.power is the desired power level, and alpha.level is the Type I error rate. Maxwell (2000) describes an example in which the population squared multiple correlation coefficient is .131 and reduces to .068 when the predictor of interest is removed in a situation where there are five regressors.[6] This example can be implemented with ss.power.rc() as follows:

```
R> ss.power.rc(Rho2.Y_X=0.131,
Rho2.Y_X.without.k= 0.068, K=5,
alpha.level=.05, desired.power=.80)
  $Necessary.Sample.Size
  [1] 111
  $Actual.Power
  [1] 0.8025474
  $Noncentral.t.Parm
  [1] 2.836755
  $Effect.Size.NC.t
  [1] 0.2692529
```

Thus, in the situation described, necessary sample size in order to have a power of .80 is 111.[7]

*Sample Size Planning for Targeted Effects: AIPE for a Regression Coefficient*

Kelley and Maxwell (2003, in press) develop methods for sample size planning for unstandardized and standardized regression coefficients from the AIPE perspective. These methods have been implemented in functions ss.aipe.rc() and ss.aipe.src() from within MBESS so that the necessary sample size can be obtained. The basic specification of the ss.aipe.rc() function is of the form

```
ss.aipe.rc(Rho2.Y_X, Rho2.k_X.without.k,
K, b.k, width, sigma.Y, sigma.X.k, conf.level=.95)
```

where Rho2.Y_X is the population squared multiple correlation coefficient, K is the number of predictors, b.k is the regression coefficient for the *k*th predictor variable (i.e., the predictor of interest), Rho2.k_X.without.k is the population squared multiple correlation coefficient predicting the *k*th predictor from the remaining $K - 1$ predictors, sigma.Y is the population standard deviation of the response variable, sigma.X.k is the population standard deviation of the *k*th predictor variable, and width and conf.level are the same as those in ss.aipe.R2() function. From the example for the power of an individual regression coefficient, suppose that the standard deviation of the dependent variable is 25 and the standard deviation of the predictor of interest is 100. The regression coefficient of interest in such a situation is 1.18. Supposing a desired confidence interval width of .10, necessary sample size can be planned as follows:

```
R> ss.aipe.rc(Rho2.Y_X=0.131,
Rho2.k_X.without.k=0.068,
  K=5, b.k=1.18, width=.10,
which.width="Full", sigma.Y=25,
  sigma.X.k=100)
  [1] 99
```

Thus, necessary sample size in the situation described is 99.

The function ss.aipe.src() in MBESS can be used to determine the necessary sample size for the AIPE approach for a standardized regression coefficient of interest. The most common specification of this function is of the same form as given in ss.aipe.src(), except the standardized regression coefficient is specified. Supposing the desired width is .30 for the standardized regression coefficient of .294, necessary sample size can be planned as follows:

```
R> ss.aipe.src(Rho2.Y_X=0.131,
Rho2.k_X.without.k=0.068,
  K=5, beta.k=.294, width=.30,
which.width="Full")
  [1] 173
```

where beta.k is used instead of b.k to emphasize that the regression coefficient is standardized. Thus, the necessary sample size in such situation is 173.

## Student's *t* Test in R

Student's *t* test is used for testing hypotheses about means when the population variance is unknown. There are three types of *t* tests: (a) one-sample *t* test, which compares the sample mean to a specified population mean; (b) paired-samples *t* test, which compares the means of two paired samples; and (c) the two-group *t* test, which compares the means of two independent samples. We can use the function t.test() to do all three types of *t* tests.

### One-Sample *t* Test

When a one-sample *t* test is desired, the basic specification of t.test() is of the form

```
t.test(x, mu, conf.level=.95)
```

where x is the particular data of interest, mu is the specified population value of the mean, and conf.level is desired confidence interval coverage.

To illustrate the one-sample *t* test, we employ the data reported in Hand, Daly, Lunn,

McConway, and Ostrowski (1994) on the estimation of room length. Shortly after metric units were officially introduced in Australia, a group of 44 students was asked to estimate in meters the width of the lecture hall in which they were sitting. The true width of the hall was 13.1 meters. To test the hypothesis that students' estimation of the width of the hall in metric units was equal to the true value, the following code is used to load the data and then to test the hypothesis:

```
R> meter <- c(8,9,10,10,10,10,10,10,11,11,
11,11,12,12,13,13,13,14,14,14,15,15,15,15,15,
15,15,15,16,16,16,17,17,17,17,18,18,20,22,25,
27,35,38,40)
R> t.test(meter, mu=13.1)
One Sample t-test
data: meter
t = 2.7135, df = 43, p-value = 0.009539
alternative hypothesis: true mean is not
equal to 13.1
95 percent confidence interval:
13.85056 18.19490
sample estimates:
mean of x
16.02273
```

The summary table shows that the $t$ statistic, with 43 degrees of freedom, is 2.714 with a corresponding (two-sided) $p$ value less than .01. The 95% confidence interval for the population mean is

$$CI_{.95} = [13.851 \leq \mu \leq 18.195],$$

where $\mu$ is the population mean. Thus, the students' estimate of the room length in meters differed significantly from its actual value. Both $p$ value and confidence interval reveal that it is unlikely to observe data such as those in this study if the students were able to make a correct guess in meters.[8]

### Paired-Sample *t* Test

When a paired-sample $t$ test is desired, `t.test()` is specified as

```
t.test(x, y, mu, paired=TRUE, conf.level=.95)
```

where `x` and `y` are the paired groups of data of interest, `paired=TRUE` signals that the procedure for the paired $t$ test is to be used, and other arguments are the same as those in the one-sample $t$ test context.

We will demonstrate a paired-samples $t$ test using the data from Cushny and Peebles (1905), which was used by Gosset ("Student") to demonstrate the theoretical developments of the $t$ distribution (Student, 1908). The data are the average number of hours of sleep gained by 10 patients on two different drugs, Dextro-hyoscyamine hydrobromide and Laevo-hyoscyamine hydrobromide. Gosset used the paired-sample $t$ test to test the hypothesis that the average sleep gain by two different drugs was the same. We define two vectors, Dextro and Laevo with the scores from Dextro-hyoscyamine hydrobromide and Laevo-hyoscyamine hydrobromide, respectively, and then implement a paired-samples $t$ test:

```
R> Dextro <- c(.7, -1.6, -.2, -1.2, -.1, 3.4,
3.7, .8, 0, 2)
R> Laevo <- c(1.9, .8, 1.1, .1, -.1, 4.4, 5.5,
1.6, 4.6, 3.4)
R> t.test(Dextro, Laevo, paired=TRUE)
Paired t-test
data: Dextro and Laevo
t = -4.0621, df = 9, p-value = 0.002833
alternative hypothesis: true difference in
means is not equal to 0
95 percent confidence interval:
-2.4598858 -0.7001142
sample estimates:
mean of the differences
-1.58
```

Thus, the observed $t$ statistic, with 9 degrees of freedom, is –4.0621 with a corresponding (two-sided) $p$ value of .0028. The mean of the differences is –1.58 in the sample, and the 95% confidence interval for the population mean difference is

$$CI_{.95} = [-2.460 \leq \mu_D - \mu_L \leq -0.700],$$

where $\mu_D$ and $\mu_L$ are the population mean of the hours of sleep for Dextro and Laevo, respectively.

Another way to conduct a paired-sample $t$ test is to calculate the difference between each pair first and then conduct a one-sample $t$ test on the differences. Therefore, an equivalent way to test the hypothesis that the two drugs have equivalent effects on drugs is to calculate the differences and use the `t.test()` function in the same manner as was done previously in the one-sample context:

```
R> D <- Dextro—Laevo
R> t.test(D, mu=0)
One Sample t-test
data: D
t = −4.0621, df = 9, p−value = 0.002833
alternative hypothesis: true mean is not
equal to 0
95 percent confidence interval:
−2.4598858 −0.7001142
sample estimates:
mean of x
−1.58
```

Notice that the results of the analyses are the same.[9]

### Two Independent Group *t* Test

To perform a two independent group *t* test, the most common specification of t.test() is of the form

```
t.test(y ~ x, var.equal=TRUE)
```

where x and y are the particular groups of data of interest, and var.equal is a logical statement of whether the variances of the two groups of data are assumed equal in the population. By default, R assumes that the variances are unequal and uses a degrees-of-freedom correction based on the degree of observed heterogeneity of variance. Because most other statistical programs assume homogeneity of variance, in that they use the standard two-group *t* test, we have specified var.equal=TRUE in our example for comparison purposes.

We use the data reported from Thompson (Chapter 17, this volume), denoted LibQUAL+™, which is a random sample of perceived quality of academic library services from a larger data set (see also Thompson, Cook, & Kyrillidou, 2005, 2006). The data have been added as a data set in the MBESS package and can be loaded with the following data function:

```
R> data(LibQUAL).
```

The grouping variables are (a) Role (undergraduate student, graduate student, and faculty) and (b) Sex. The outcome variables are (a) LIBQ_tot, (b) ServAffe, (c) InfoCont, (d) LibPlace, (e) Outcome, and (f) Satisfac. Thompson (Chapter 17, this volume) uses a *t* test to compare the sex differences on the outcome variable LIBQ_tot and conducts a two-way ANOVA to compare the effects of role and sex on LIBQ_tot. In the following section, we will demonstrate how to use t.test() and other functions to reproduce the methods discussed in Chapter 17 and related methods with R and MBESS. More specifically, we will discuss how to compute standardized effect sizes, confidence intervals for standardized mean differences, and sample size planning in the *t* test context in R.

After loading the data set LibQUAL, the function class() is used to determine the attribute of the object LibQUAL.

```
R> class(LibQUAL)
[1] "data.frame"
```

Thus, LibQUAL is a data frame, which is a special type of data structure where numeric and categorical variables can be stored. We also need to verify whether Sex has the attribute of a factor because only when a variable is defined as factor can that variable be used as a grouping variable:

```
R> class(LibQUAL$Sex)
[1] "integer"
```

Because the vector Sex is specified as an integer, it needs to be converted to a factor for analysis. Notice that the dollar sign ($) is used to extract a named column from the data frame. We use the function as.factor() to convert the attribute into a factor and then redefine Sex in the data frame as a factor (notice that the dollar sign is used on both sides of the assignment operator) so that we can easily perform the two-group *t* test:

```
R> LibQUAL$Sex <- as.factor(LibQUAL$Sex)
R> class(LibQUAL$Sex)
[1] "factor"
R> t.test(LIBQ_tot ~ Sex, data=LibQUAL,
var.equal=TRUE)
Two Sample t-test
data: LIBQ_tot by Sex
t = −0.2381, df = 64, p−value = 0.8125
alternative hypothesis: true difference in
means is not equal to 0
95 percent confidence interval:
−0.7682029 0.6045665
sample estimates:
mean in group 0 mean in group 1
6.893636 6.975455
```

Notice that the use of the function t.test() on two independent samples is a bit different from

the other two kinds of *t* test. A formula much like that discussed in the lm() function is used where the dependent variable, LIBQ_tot, is predicted by the grouping variable, Sex. This implies that the outcome variable LIBQ_tot is modeled by the grouped variable Sex, and the model formula form in t.test()is the general format used in R for model specification.

From the output, the *t* statistic is –.2381 with 64 degrees of freedom with a corresponding (two-sided) *p* value of .8125. The mean of the first group (labeled 0) is 6.894, and the second group (labeled 1) is 6.975 in the sample. The 95% confidence interval for the population mean difference is

$$CI_{.95} = [-0.768 \leq \mu_0 - \mu_1 \leq 0.605].$$

Alternatively, we can perform the two-group *t* test in a similar fashion as was done with the one-sample *t* test and the paired-samples *t* test. That is, we can specify the form

```
t.test(x, y, mu, var.equal=TRUE,
conf.level=.95)
```

where x and y are the two independent groups of data of interest. Notice that compared to paired-samples *t* tests, we exclude the command paired=TRUE but add the command var.equal=TRUE.

### Confidence Intervals for Effect Sizes Related to the Group Means and Group Mean Differences

Although the confidence interval from the *t* test output provides helpful information, at times what is of interest is the standardized mean difference and its corresponding confidence interval. A commonly used effect size in the *t* test is the standardized mean difference (e.g., Cohen, 1988), *d,* which is defined as

$$d = \frac{M_1 - M_2}{s} \qquad (7)$$

in the sample, where $M_1$ is the mean of Group 1, $M_2$ is the mean of Group 2, and *s* is the square root of the pooled variance, assumed equal across groups in the population.

The function smd() in MBESS can be used to calculate the standardized mean difference. It is most commonly specified in the form

```
smd(Group.1, Group.2)
```

where Group.1 and Group.2 are the particular data of interest from Group 1 and Group 2, respectively. Hence, specifying smd() as follows returns the standardized mean difference between the scores on the LibQUAL+™ total scale of the female group (Sex=0) and of the male group (Sex=1):[10]

```
R> smd(Group.1=LibQUAL[1:33,4],
Group.2=LibQUAL[34:66,4])
[1] –0.05862421
```

To obtain the unbiased estimate of the population standardized mean difference (*d* is slightly biased), the option Unbiased=TRUE can be used:

```
R> smd(Group.1=LibQUAL[1:33,4],
Group.2=LibQUAL[34:66,4], Unbiased=TRUE)
[1] –0.05793406
```

The correction used in the function smd() yields an exactly unbiased statistic (based in part on the gamma function), whereas that used in Thompson (Chapter 17, this volume) yields an approximately unbiased statistic (see Hedges & Olkin, 1985, for derivations and discussion of the exact and approximately unbiased statistics).

We can obtain the confidence interval for the standardized mean difference with the function ci.smd() in MBESS. Its basic specification is of the form

```
ci.smd(smd, n.1, n.2, conf.level)
```

where smd is the observed standardized mean difference; n.1 and n.2 are sample sizes of Group 1 and Group 2, respectively; and conf.level is the desired confidence interval coverage. Therefore, to construct the 95% confidence interval for the standardized mean difference in the scores on the LibQUAL+™ total scale of the female group and the male group, ci.smd() could be specified as follows:

```
R> ci.smd(smd=–0.05862421, n.1=33,
n.2=33, conf.level=.95)
$Lower.Conf.Limit.smd
[1] –0.541012
$smd
[1] –0.05862421
$Upper.Conf.Limit.smd
[1] 0.4242205
```

### Power Analysis for *t* Test

It is important to consider statistical power when designing a study, as the power of a statistical test is the probability that it will yield statistically significant results (e.g., see Cohen, 1988, for a review). Since power is a function of Type I error rate, standardized effect size, and sample size, after specifying the Type I error rate and standardized effect size, the necessary sample size given a specified power value or power given a specified sample size can be determined. The function power.t.test() in R is used to plan necessary sample size. It is usually specified as

```
power.t.test(power, delta, sd, type)
```

where power is the desired power level, delta is the (unstandardized) mean difference, sd is the population standard deviation, and type is the type of the *t* test ("one.sample" for one sample, "two.sample" for two independent sample, and "paired" for paired sample). Note that when sd=1 (which is the case by default), delta can be regarded as the standardized mean difference. For example, we can get the necessary sample size for each group to achieve a power = 0.8 for a two-sample *t* test when the standardized mean difference is 0.5:

```
R> power.t.test(power=.8, delta=.5,
type="two.sample"),
   Two-sample t test power calculation
   n = 63.76576
   delta = 0.5
   sd = 1
   sig.level = 0.05
   power = 0.8
   alternative = two.sided
   NOTE: n is number in *each* group
```

Thus, after rounding to the next larger integer, it can be seen that a per group sample size of 64 is necessary to achieve a power = 0.8 (128 is thus the total sample size). Alternatively, power can be determined when n is specified (instead of power) in the power.t.test() function.

### AIPE for Mean Differences and Standardized Mean Differences

The AIPE approach to sample size planning can be used in a similar manner, where what is of interest is the necessary sample size for the expected confidence interval width to be sufficiently narrow, optionally with some assurance that the confidence interval will be sufficiently narrow (Kelley & Rausch, 2006). For example, suppose the population standardized mean difference is .50, and it is desired that the total 95% confidence interval width be .50. The ss.aipe.sm() function in MBESS could be specified as follows:

```
   R> ss.aipe.smd(delta=.5, conf.level=.95,
width=.50)
   [1] 127
```

Because the standard procedure is for the expected width, which implies that roughly 50% of the time, the confidence interval will be wider than desired, a desired degree of assurance can be incorporated into the sample size procedure to specify the probability that a confidence interval will not be wider than desired. For example, suppose one would like to be 99% assurance that the computed confidence interval will be sufficiently narrow. The ss.aipe.smd() function could be specified as follows:

```
   R> ss.aipe.smd(delta=.5, conf.level=.95,
width=.50, assurance=.99)
   [1] 133
```

## Analysis of Variance

Analysis of variance (ANOVA) is a method to compare the means of two or more groups. The function for fitting an ANOVA model in R is aov(), whose usage is very similar to that of the lm() function illustrated for the multiple regression examples. The difference between aov() and lm() is that, when summary() is used to present the model summary, aov() objects return an ANOVA table, whereas lm() objects return a regression table. There is a function anova(), not to be confused with aov(), that is used to return the ANOVA table of an existing object or a comparison between nested models.

The basic specification of the function for ANOVA is aov() and is used in the following manner:

```
aov(formula, data)
```

where both arguments are the same as those in lm(). Note that the grouping variable in the formula must have one or more factors (i.e., groups) identified. For example, to perform ANOVA on the LibQUAL+™ data, where the hypothesis is that no mean differences exist in the

effects of Role on the outcome variable, LIBQ_tot, consider the following specification of aov():

```
R> LibQUAL.aov.Role <- aov(LIBQ_tot ~ Role, data=LibQUAL).
```

The object LibQUAL.aov.Role contains the necessary information to report results, but it does so in only a limited way. The results of interest (i.e., the ANOVA table) are obtained by using either summary() or anova() on the object fitted by aov():

```
R> summary(LibQUAL.aov.Role)
```

| | Df | Sum Sq | Mean Sq | F value | Pr(>F) |
|---|---|---|---|---|---|
| Role | 2 | 2.698 | 1.349 | 0.6961 | 0.5023 |
| Residuals | 63 | 122.072 | 1.938 | | |

```
R> anova(LibQUAL.aov.Role)
Analysis of Variance Table
Response: LIBQ_tot
```

| | Df | Sum Sq | Mean Sq | F value | Pr(>F) |
|---|---|---|---|---|---|
| Role | 2 | 2.698 | 1.349 | 0.6961 | 0.5023 |
| Residuals | 63 | 122.072 | 1.938 | | |

We can also use lm() to conduct an ANOVA on the previous example (notice the *F*-statistic at the end of the output):

```
R> LibQUAL.lm.Role <- lm(LIBQ_tot ~ Role, data = LibQUAL)
```

as both ANOVA and regression are special cases of the general linear model. However, the summary() function for an lm object does not return an ANOVA table; instead, it returns a regression table:

```
R> summary(LibQUAL.lm.Role)
Call:
lm(formula = LIBQ_tot ~ Role, data = LibQUAL)
Residuals:
```

| Min | 1Q | Median | 3Q | Max |
|---|---|---|---|---|
| -5.1127 | -0.4535 | 0.2084 | 0.7965 | 1.9795 |

```
Coefficients:
```

| | Estimate | Std. Error | t value | Pr(>\|t\|) |
|---|---|---|---|---|
| (Intercept) | 6.792727 | 0.296775 | 22.888 | <2e-16 *** |
| Role2 | -0.002273 | 0.419703 | -0.005 | 0.996 |
| Role3 | 0.427727 | 0.419703 | 1.019 | 0.312 |

```
—
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 1.392 on 63 degrees of freedom
Multiple R-Squared: 0.02162, Adjusted R-squared: -0.009439
F-statistic: 0.6961 on 2 and 63 DF, p-value: 0.5023 . . .
```

Because the lm() function was used, the sample multiple correlation coefficient, Multiple R-Squared, and the adjusted multiple correlation coefficient, Adjusted R-squared, are made available with the summary() function.

### Confidence Intervals for Standardized Effect Sizes for Omnibus Effects

The effect size for the *p*th factor of the ANOVA model is defined as

$$\eta_p^2 = \frac{\sigma_p^2}{\sigma_T^2} \tag{8}$$

or

$$\phi_p^2 = \frac{\sigma_p^2}{\sigma_E^2}, \tag{9}$$

where $\sigma_p^2$ is the variance due to factor *p*, $\sigma_T^2$ is the total variance of the dependent variable, and $\sigma_E^2$ is the within-group variance of the dependent variable (Fleishman, 1980; Kelley, 2007a; Steiger, 2004).[11] Due to the structure of the effect size, $\eta_p^2$ is the proportion of variance in the dependent variable accounted for by the grouping factor, and $\phi_p^2$ is the signal-to-noise ratio. The MBESS package contains functions to calculate confidence intervals for these quantities.

The function ci.pvaf() in the MBESS package can be used to calculate the confidence limits for the proportion of variance in the dependent variable accounted for by the grouping variable (i.e., $\eta_p^2$). Its basic specification is of the form

```
ci.pvaf(F.value, df.1, df.2, N, conf.level=0.95)
```

where F.value is the observed *F* value from fixed effects ANOVA for the particular factor, df.1 is the numerator degrees of freedom for the *F* test, df.2 is the denominator degrees of freedom, N is the sample size (which need not be specified for single-factor designs), and conf.level is the confidence interval coverage. To obtain the 95% confidence interval for $\eta_p^2$ in the example we used when discussing the function aov(), ci.pvaf() should be specified as follows.

```
R> ci.pvaf(F.value=0.6961, df.1=2, df.2=63, N=66)
$Lower.Limit.Proportion.of.Variance.Accounted.for
[1] 0
$Upper.Limit.Proportion.of.Variance.Accounted.for
[1] 0.1107225
```

The function ci.snr() in MBESS can be used to obtain the confidence limits for the signal-to-noise ratio (i.e., $\phi_p^2$). Its basic specification is of the form

```
ci.snr(F.value, df.1, df.2, N, conf.level=0.95),
```

where all the arguments are the same as those of ci.pvaf(). To obtain the 95% confidence interval for $\phi_p^2$ from the example we used when discussing aov(), ci.snr() should be specified as follows:

```
R> ci.snr(F.value=0.6961, df.1=2, df.2=63, N=66)
$Lower.Limit.Signal.to.Noise.Ratio
[1] 0
$Upper.Limit.Signal.to.Noise.Ratio
[1] 0.1245084
```

### Confidence Intervals for Targeted Effects

Although the omnibus *F* test often addresses an important research question, it is many times desirable to perform follow-up comparisons in an effort to examine specific targeted effects. The ci.c() function from MBESS can be used to form confidence intervals for the population contrasts in an ANOVA setting. Its basic specification is of the form

```
ci.c(means, error.variance, c.weights, n, N, conf.level)
```

where means is a vector of the group means, error.variance is the common variance of the error (i.e., the mean square error), c.weights is a vector of contrast weights, n is a vector of sample sizes in each group, N is the total sample size (which need not be specified in a single-group design), and conf.level is the confidence interval

coverage. For example, to obtain the 95% confidence interval for the difference between the mean of students (weighted mean of undergraduate and graduate) versus the mean of faculty in the example we used when discussing aov(), ci.c would be specified as follows.

```
R>
ci.c(means=c(6.792727,6.790454,7.220454),
c.weights=c(1/2, 1/2, −1), n=c(22,22,22),
error.variance=1.859, conf.level=.95)
$Lower.Conf.Limit.Contrast
[1] −1.140312
$Contrast
[1] −0.4288635
$Upper.Conf.Limit.Contrast
[1] 0.282585
```

The function ci.sc() in MBESS can be used to form confidence intervals for the population standardized contrast in an ANOVA setting. Its basic specification is of the same form as ci.c(), except that the standardized contrast and confidence limits are returned. For example,

```
R>
ci.sc(means=c(6.792727,6.790454,7.220454),
error.variance=1.859, c.weights=c(−1, 1/2,
1/2), n=c(22,22,22), conf.level=.95)
$Lower.Conf.Limit.Standardized.Contrast
[1] −0.3570958
$Standardized.contrast
[1] 0.1560210
$Upper.Conf.Limit.Standardized.Contrast
[1] 0.6679053
```

## Longitudinal Data Analysis With R

Longitudinal research has become an important technique in the BESS because of the rich set of research questions that can be addressed with regards to intra- and interindividual change (e.g., Collins & Sayer, 2001; Curran & Bollen, 2001; Singer & Willett, 2003). Multilevel models (also called hierarchical models, mixed-effects models, random coefficient models) are a commonly used method of modeling and understanding change because these models explicitly address the nested structure of the data (e.g., observations nested within individual, individual nested within group, etc.). The *nlme* package (Pinheiro et al., 2007) provides powerful methods for analyzing both linear and nonlinear multilevel models.

We will use the data set Gardner.LD in the MBESS package for illustration purposes. The data set Gardner.LD contains the performance data of 24 individuals, who were presented with 420 presentations of four letters and were asked to identify the next letter that was to be presented. The 420 presentations were (arbitrarily it seems) grouped into 21 trials of 20 presentations. Twelve of the participants were presented the letters S, L, N, and D with probabilities .70, .10, .10, and .10, respectively, and the other 12 were presented the letter L with probability .70 and three other letters, each with a probability of .10. The analysis of longitudinal data in *nlme* requires data to be coded in *person-period* (Singer & Willett, 2003) form (also known as the "the univariate way") so that each person has a row for each of the different measurement occasions. There are four variables in the Gardner.LD data set: ID, Score, Trial, and Group. Because each participant had 21 trials, the dimension of the data matrix is 504 ($24 \times 21$) by 4. As an initial step after loading the *nlme* package and calling into the session the Gardner.LD data, it is desirable to group the data using the groupedData() function, which contains not only the data but also information on the nesting structure of the design:

```
R> data(Gardner.LD)
R> grouped.Gardner.LD <− groupedData
(Score ~ Trial|ID, data=Gardner.LD)
```

The formula Score ~ Trial|ID implies that the response variable, Score, is modeled by the primary covariate, Trial, given the grouping factor, ID. In longitudinal data, the primary covariate that is monotonically related to time (e.g., time itself, grade level, occasion of measurement, etc.) and the grouping factor indicates the variable used to denote the individual. After creating the groupedData object, we can use plot() to plot individual trajectories.

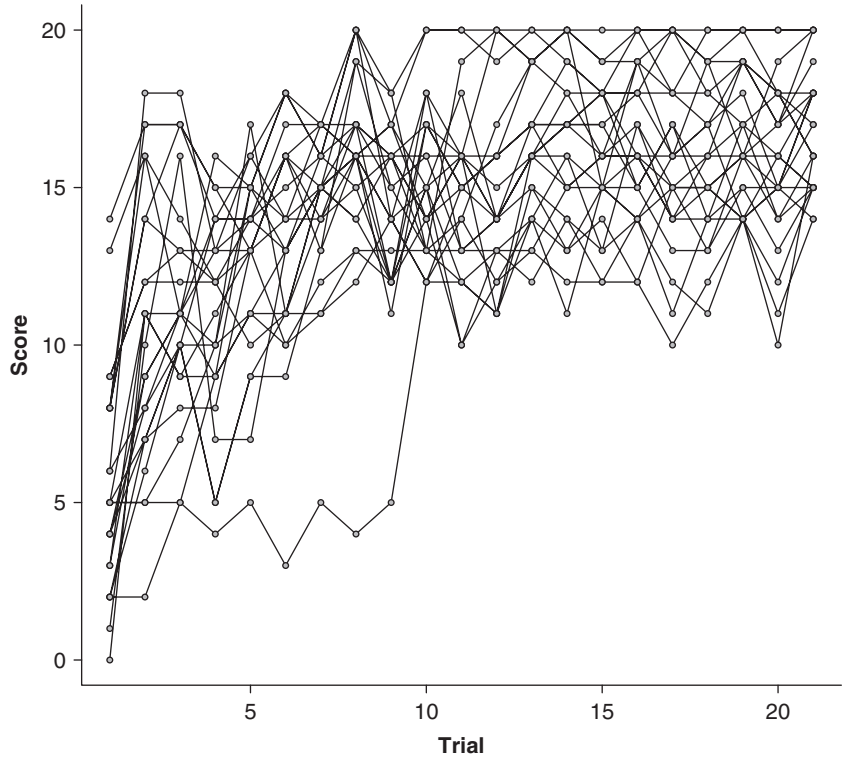


**Figure 34.11** The growth trajectories of 24 participants in the Gardner learning data.

```
R> plot(grouped.Gardner.LD)
```

Because information on the nesting structure is contained within `groupedData` object, the `plot()` function creates trajectories conditional on each individual in separate plots.

However, it is sometimes desired to plot the trajectories in a single plot to help visualize the amount of interindividual differences in change. The `vit()` function in MBESS provides the plot with all trajectories in a single figure. Consider the following application of the `vit()` function:

```
R> vit(id="ID", occasion="Trial",
score="Score", Data=Gardner.LD, xlab="Trial")
```

Similar figures with other options can be obtained with the `xyplot()` function from the *lattice* package (Sarkar, 2006). Figure 34.12 shows that the change curves for most individuals are nonlinear, starting at relatively low point and growing toward the upper value of 20.

After plotting the data, it is decided to use a logistic change curve to model the trajectories (a negative exponential model should also be considered). The model selected is defined as

$$y_{ij} = \frac{\phi_{1i}}{1 + \exp[-(t_{ij} - \phi_{2i})/\phi_{3i}]} + \varepsilon_{ij}, \quad (10)$$

where $y_{ij}$ is the score for individual $i$ at time $j$; $t_{ij}$ is the $j$th trial for individual $i$; $\phi_{1i}$, $\phi_{2i}$, and $\phi_{3i}$ are parameters for individual $i$; and $\varepsilon_{ij}$ is the error for the $i$th individual at the $j$th measurement occasion. The *nlme* package enables the user to use several self-starting functions for commonly used nonlinear regression models. We will use the `SSlogis()` function within the `nlsList()` function to obtain parameter estimates for each of the individuals. This can be done with the following commands:

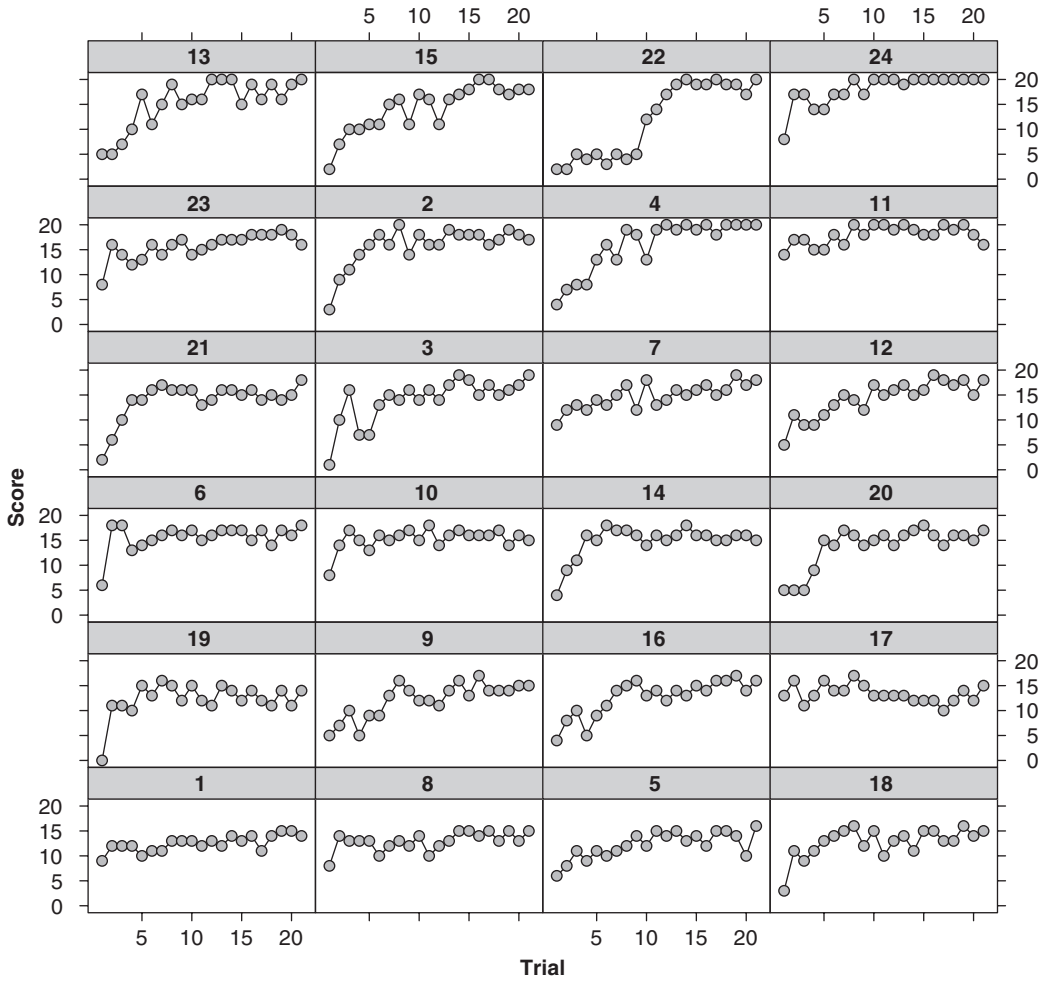


**Figure 34.12** The growth trajectory of each participant in the Gardner learning data in a grouped plot.

```
R> NLS.list.GL <— nlsList(SSlogis, grouped.Gardner.LD)
R> NLS.list.GL
Call:
Model: Score ~ SSlogis(Trial, Asym, xmid, scal) | ID
Data: grouped.Gardner.LD
Coefficients:
```

| | Asym | xmid | scal |
|---|---|---|---|
| 1 | 20.64888 | -0.39425400 | 25.9868490 |
| 8 | 25.82947 | 10.77112398 | 39.0163528 |
| 5 | 14.00626 | 1.19170594 | 3.3132692 |
| 18 | 13.76810 | 1.69643789 | 1.1114158 |
| 19 | NA | NA | NA |
| 9 | 14.67500 | 2.87314499 | 3.2330142 |

| | | | |
|---|---|---|---|
| 16 | 15.09849 | 2.95021742 | 2.7440494 |
| 17 | 43.44157 | -69.37127304 | -98.5832435 |
| 6 | NA | NA | NA |
| 10 | 15.73215 | 0.99038220 | 0.4381405 |
| 14 | 15.99992 | 1.92285958 | 0.8565851 |
| 20 | 15.93150 | 3.20708423 | 1.3398499 |
| 21 | 15.41431 | 2.42709904 | 0.7902158 |
| 3 | 16.98242 | 2.88241004 | 3.0970538 |
| 7 | 17.70834 | -2.30174516 | 7.1434532 |
| 12 | 17.43223 | 2.57507721 | 3.8061267 |
| 23 | 18.17980 | -2.16624638 | 6.0792346 |
| 2 | 17.39742 | 2.28047778 | 1.0706019 |
| 4 | 19.60291 | 3.90543041 | 2.4649680 |
| 11 | 18.94723 | -3.34623016 | 3.6815169 |
| 13 | 18.11099 | 3.42193123 | 1.9878506 |
| 15 | 18.10627 | 3.84541898 | 3.4853045 |
| 22 | 19.93583 | 9.39892560 | 2.0708870 |
| 24 | 20.12397 | -0.05064117 | 3.4446175 |

Degrees of freedom: 462 total; 396 residual
Residual standard error: 1.73849

The individuals are fitted using a separate (three-parameter) logistic model to each subject (the ID number is in the first unmarked column). The model from the SSlogic() function in the current example shows the following:

```
Score ~ SSlogis(Trial, Asym, xmid, scal) | ID
```

and thus the fitted model becomes

$$\widehat{Score}_{ij} = \frac{Asym_i}{1 + \exp[-(Trial_{ij} - xmid_i)/scal_i]}, \quad (11)$$

where *Asym* represents the asymptote, *xmid* represents the value of trial (or time, more generally) at the inflection point of the curve, and *scal* is the scale parameter. The trajectory of Individual 17 differs considerably from the others individuals and from the logistic model, which is why this individual's parameter estimates differ so considerably from the others. Note that both Individuals 6 and 19 do not have parameter estimates because the logistic model failed to converge. Examination of the plot reveals that the logistic change model does not adequately describe the trajectories of Individuals 6 and 19 (a negative exponential change model would be more appropriate). Specification of the subset.ids argument in the vit() function allows specific individual IDs to be plotted, which can be helpful for identifying misfits.

The nlsList model is useful when the goal is to model the growth trajectory of a particular fixed set of individuals. However, when interest is in estimating a multilevel model with fixed effects and covariance structure—to examine the variability within and among individuals—the function nlme() can be used.

Since we already have a fitted nlsList() object (NLS.list.GL) for each individual, we can input that object into the function nlme(), where starting values are automatically obtained. To examine the results, consider the following commands:

```
R> nlme.GL <— nlme(NLS.list.GL)
R> summary(nlme.GL)
Nonlinear mixed-effects model fit by maximum likelihood
Model: Score ~ SSlogis(Trial, Asym, xmid, scal)
Data: grouped.Gardner.LD
```

| AIC | BIC | logLik |
|---|---|---|
| 2201.701 | 2243.926 | -1090.850 |

```
Random effects:
Formula: list(Asym ~ 1, xmid ~ 1, scal ~ 1)
Level: ID
Structure: General positive-definite, Log-Cholesky parametrization
```

| | StdDev | Corr | |
|---|---|---|---|
| Asym | 2.084844 | Asym | xmid |
| xmid | 1.988619 | 0.399 | |
| scal | 1.033261 | 0.522 | 0.231 |

```
Residual 1.750237
Fixed effects: list(Asym ~ 1, xmid ~ 1, scal ~ 1)
```

| | Value | Std.Error | DF | t-value | p-value |
|---|---|---|---|---|---|
| Asym | 16.062330 | 0.4400112 | 478 | 36.50437 | 0 |
| xmid | 2.092424 | 0.4277958 | 478 | 4.89118 | 0 |
| scal | 1.619961 | 0.2422403 | 478 | 6.68742 | 0 |

```
Correlation:
```

| | Asym | xmid |
|---|---|---|
| xmid | 0.379 | |
| scal | 0.490 | 0.125 |

```
Standardized Within-Group Residuals:
```

| Min | Q1 | Med | Q3 | Max |
|---|---|---|---|---|
| -3.25058923 | -0.59251518 | 0.05021871 | 0.60113672 | 4.23964442 |

```
Number of Observations: 504
Number of Groups: 24
```

From the output, information about fixed effects, random effects, and the fit of the model is available. Using the object obtained from the nlsList() in nlme() automatically considers each of the fixed effects to be random. This can be modified by specifying fixed and random effects explicitly with the fixed and random options within nlme().

To visually examine the residuals, the following commands can be used:

```
R> plot(nlme.GL)
R> qqnorm(nlme.GL, abline=c(0,1))
```

The residual plot shows that the assumption of equal variances across time seems reasonable. This assumption can be relaxed with additional specification in nlme(). A normal QQ plot for the residual shows that the distribution of residuals seems normal.

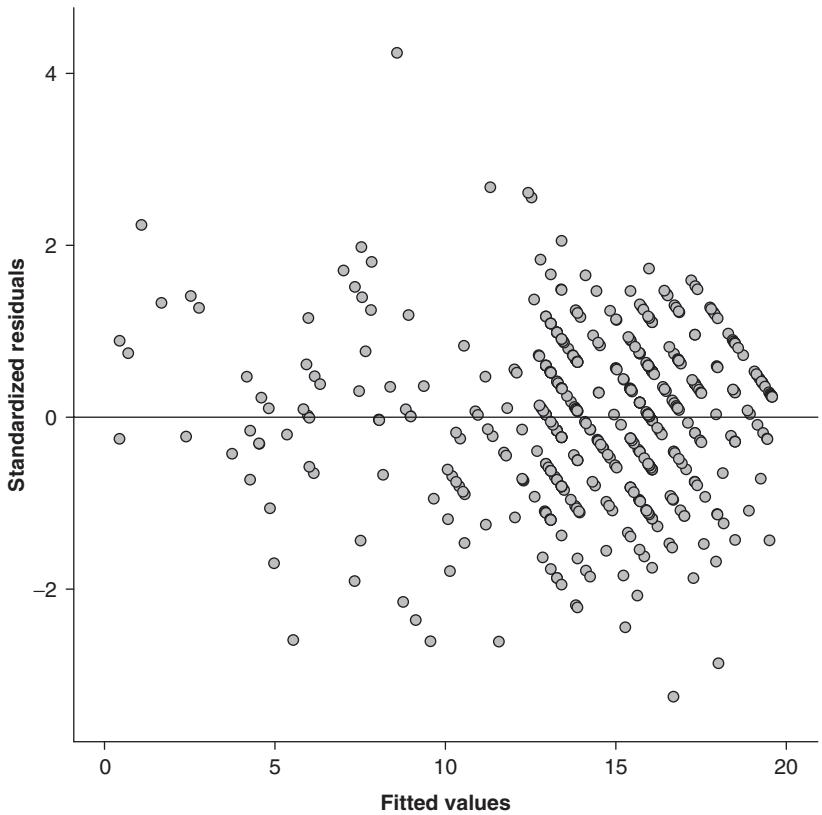


**Figure 34.13** Scatterplot for standardized residuals as a function of fitted values for the fitted logistic change model.

We can also plot the fitted trajectories for each individual by applying `plot()` to certain functions of a fitted `nlme()` object, which provides important information about potential model misfits:

```
R> plot(augPred(nlme.GL, level=0:1))
```

This section serves as only a brief introduction to the *nlme* package. Readers interested in more comprehensive discussion on analyzing mixed effects models (both linear and nonlinear) with the *nlme* package are referred to *Mixed-Effects Models in S and S-PLUS* (Pinheiro & Bates, 2000). Also helpful is Doran and Lockwood (2006), who recently provided a tutorial on linear multilevel model with the *nlme* package, with special emphasis on educational data.

## Conclusions

We hope that we have successfully illustrated R as a valuable tool for performing a wide variety of statistical analyses. Of course, R can do much more than has been presented. The use of R and the plethora of tools made available with R packages should be considered by researchers who perform statistical analyses. One thing not mentioned is an implicit benefit of R, that many times R code can be slightly modified from analysis to analysis for similar data and research questions, which greatly facilitates future analyses. For researchers who often perform the same type of analysis with different data, such a benefit can save a great deal of time.

We know from experience that many statisticians and quantitative methodologists within the BESS use R for their research. Such research is at times implemented in collaborative efforts with substantive researchers, where R is used to implement the particular method. Having R as a common language would be beneficial in such collaborative relationships. We also know from experience that R is beginning to be used in graduate courses in the BESS at well-known institutions, both in "first-year graduate statistics"

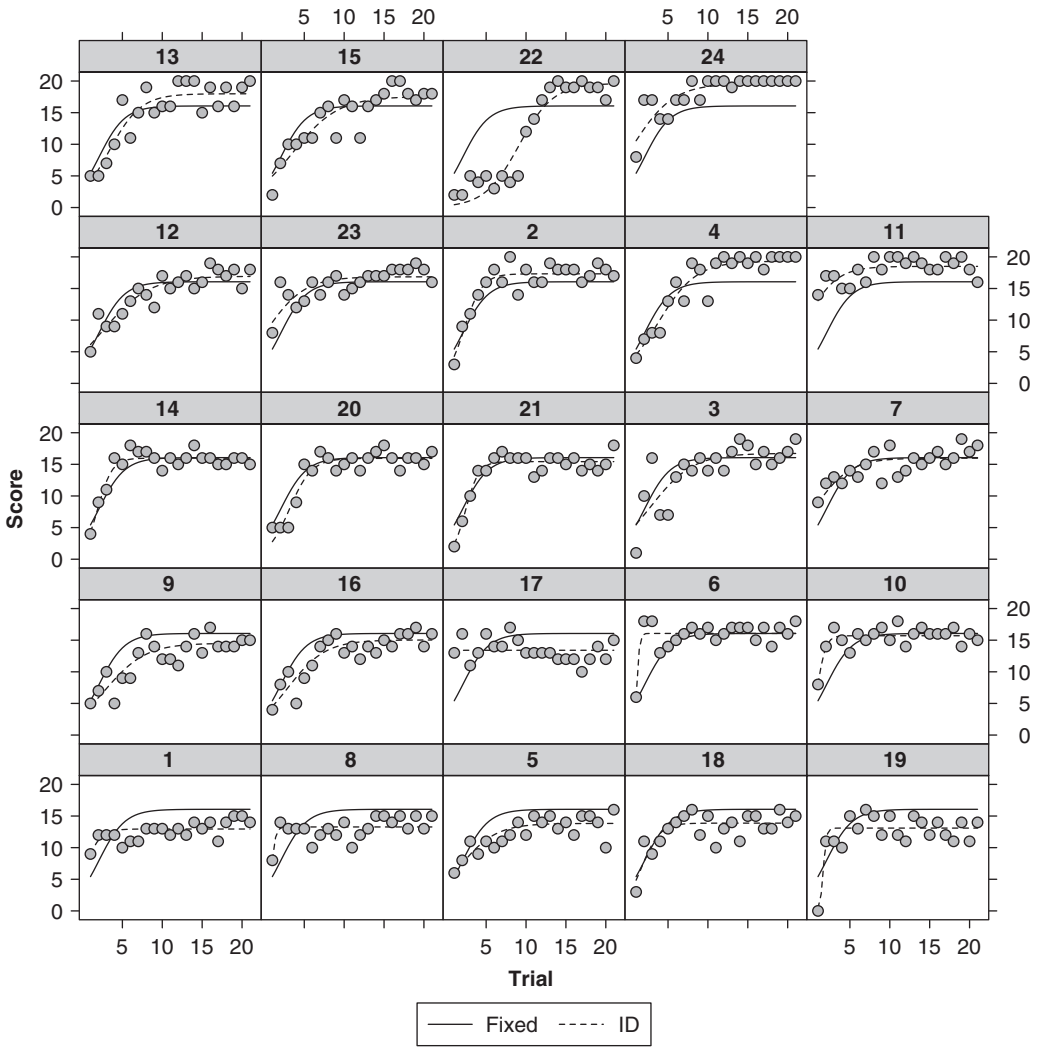


**Figure 34.14** Observed values (circles), individuals' fitted values (dashed line), and population estimates of change curve (solid line) for the logistic change model applied to the Gardner learning data.

sequences and (more commonly) for advanced statistics classes. Thus, more and more future researchers will have at least some exposure to statistical analyses implemented in R. Evidence is thus beginning to mount that R will be even more important and widely used in the coming years than it is today. Thus, there is no time like the present to begin incorporating R into one's set of statistical tools.

R cannot do everything, and we do not want to imply that it can. For some techniques, R is quite limited. A large portion of the statistical procedures implemented within R has come from mathematical statisticians and statisticians working outside the BESS. Since such individuals do not use some of the methods that are important to researchers within the BESS, some methods important to BESS researchers have not yet been implemented in R. However, as more packages become available that implement methods especially useful for the BESS, such as MBESS, the "missing methods" will continue to decrease in quantity. We realize that not everyone will immediately download, install, and start using R. However, we do hope that researchers will be open to using the program, realize that it is not as difficult as it might initially seem, and consider making it part of their repertoire of statistical tools.

## Notes

1. The "community of R users," freely available downloadable books, documentation, mailing lists, and software downloads are available at the R Web site: http://r-project.org.

2. He was once editor (1993–1997) of the *Journal of Educational* [now *and Behavioral*] *Statistics*.

3. Like R, S-Plus is a program based on the S language. S-Plus is a proprietary program owned by Insightful Corporation. R and S-Plus share a common history because of the S language, but they are ever growing further apart.

4. This data set has been made available in the MBESS R package. For those who did not create `prof.salary.R` in the previous sections, after loading MBESS, the following will load the data: `data(prof.salary)`.

5. Note that `R2.k_X.without.k` is equal to $1 - \text{Tolerance}_k$, *and* `R2.k_X.without.k` is equivalent to

$$1 - \frac{1}{r_{kk}},$$

where $r_{kk}$ is the $k$th diagonal element of the inverse of the covariance matrix of the $K$ regressors (Harris, 2001).

6. Actually, Maxwell (2000) describes a specific correlation structure. That correlation structure implicitly defines the values used in the chapter for illustrative purposes via definitional formulas.

7. Note that this result differs slightly from that reported in Maxwell (2000), which is 113. The discrepancy is because of rounding error in the calculations necessary to obtain the squared multiple correlation coefficients and because of the rounding error implicit in Cohen's (1988) tables, which were used to obtain the noncentrality parameter. Given the specified correlation matrix, number of predictors, and desired power, 111 is the exact value of necessary sample size.

8. Interestingly, 69 other students were asked to do the same task, except that the students estimated the length of the room in feet. The results were shown not to be significant, $t = .4624(68)$, $p = .65$ (with 95% confidence limits of 40.69, 46.70; the actual length was 43 feet).

9. Preece (1982) criticized the way Gosset used the Cushny and Peebles (1905) data to illustrate a paired-samples $t$ test. Although we tend to agree with Preece, we used the data because of their historical importance. Also, it should be noted that the data reported in Gosset's work contain a typographical error, in that there was a miscoded datum (but correct summary statistics). We used the correct data as reported in Cushny and Peebles (1905).

10. Note that the result is slightly different from that in Thompson's example (see Chapter 17, this volume), which is –0.064. This discrepancy comes from rounding error. If we use the means and the standard deviations reported in Thompson explicitly in `smd()`, the result will be the same:

```
R> smd(Mean.1=6.98, Mean.2=6.89, s.1=1.322,
s.2=1.472, n.1=33, n.2=33)
 [1] 0.06433112
```

11. Thompson (Chapter 17, this volume) denotes the effect sizes for one-way ANOVA as $\eta^2$ (multiple $R$-squared) and $\omega^2$ (adjusted $R$-squared).

## References


Aiken, L. S., & West, S. G. (1991). *Testing and interpreting interactions.* Newbury Park, CA: Sage.

Algina, J., & Olejnik, S. (2000). Determining sample size for accurate estimation of the squared multiple correlation coefficient. *Multivariate Behavioral Research, 35,* 119–136.

Becker, R. A. (1994). *A brief history of S.* Retrieved from http://cm.bell-labs.com/stat/doc/94.11.ps

Cleveland, W. S. (1979). Robust locally weighted regression and smoothing scatterplots. *Journal of American Statistics Association, 74,* 829–836.

Cleveland, W. S. (1981). LOWESS: A program for smoothing scatterplots by robust locally weighted regression. *The American Statistician, 35,* 54.

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed.). Hillsdale, NJ: Lawrence Erlbaum.

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

Collins, L. M., & Sayer, A. (Eds.). (2001). *New methods for the analysis of change.* Washington, DC: American Psychological Association.

Curran, P. J., & Bollen, K. A. (2001). The best of both worlds: Combining autoregressive and latent curve models. In L. M. Collins & A. G. Sayer (Eds.), *New methods for the analysis of change* (pp. 107–135). Washington, DC: American Psychological Association.

Cushny, A. R., & Peebles, A. R. (1905). The action of optical isomers: II. Hyoscines. *Journal of Physiology, 32,* 501–510.

de Leeuw, J. (2005). On abandoning XLISP-STAT. *Journal of Statistical Software, 13*(7), 1–5.

Ding, C. G. (1996). On the computation of the distribution of the square of the sample

multiple correlation coefficient. *Computational Statistics & Data Analysis, 22,* 345–350.
Doran, H. C., & Lockwood, J. R. (2006). Fitting value-added models in R. *Journal of Educational and Behavioral Statistics, 31*(2), 105–230.
Everitt, B. S. (2005). *An R and S-Plus companion to multivariate analysis.* London: Springer.
Fleishman, A. I. (1980). Confidence intervals for correlation ratios. *Educational and Psychological Measurement, 40,* 659–670.
Fox, J. (2002). *An R and S-PLUS companion to applied regression.* Thousand Oaks, CA: Sage.
Hand, D. J., Daly, F., Lunn, A. D., McConway, K., & Ostrowski, E. (Eds.). (1994). *A handbook of small data sets.* London: Chapman & Hall.
Harris, R. J. (2001). *A primer of multivariate statistics* (3rd ed.). Mahwah, NJ: Lawrence Erlbaum.
Hedges, L. V., & Olkin, I. (1985). *Statistical methods for meta-analysis.* Orlando, FL: Academic Press.
Kelley, K. (2007a). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24.
Kelley, K. (2007b). MBESS: Version 0.0.9 [Computer software and manual]. Retrieved from http://www.r-project.org/
Kelley, K. (2007c). Sample size planning for the squared multiple correlation coefficient: Accuracy in parameter estimation via narrow confidence intervals. Revised and submitted to *Multivariate Behavioral Research.*
Kelley, K. (in press). Methods for the behavioral, educational, and educational sciences: An R package. *Behavior Research Methods.*
Kelley, K., & Maxwell, S. E. (2003). Sample size for multiple regression: Obtaining regression coefficients that are accuracy, not simply significant. *Psychological Methods, 8,* 305–321.
Kelley, K., & Maxwell, S. E. (in press). Power and accuracy for omnibus and targeted effects: Issues of sample size planning with applications to Multiple Regression. In J. Brannon, P. Alasuutari, & L. Bickman (Eds.), *Handbook of social research methods.* New York: Russell Sage Foundation.
Kelley, K., Maxwell, S. E., & Rausch, J. R. (2003). Obtaining power or obtaining precision: Delineating methods of sample size planning. *Evaluation and the Health Professions, 26,* 258–287.
Kelley, K., & Rausch, J. R. (2006). Sample size planning for the standardized mean difference: Accuracy in parameter estimation via narrow confidence intervals. *Psychological Methods, 11*(4), 363–385.
Lapsley, M., & Ripley, B. D. (2007). RODBC: ODBC database access [Computer software and manual]. Retrieved from http://cran.r-project.org
Lee, Y. S. (1971). Tables of the upper percentage points of the multiple correlation. *Biometrika, 59,* 175–189.
Maxwell, S. E. (2000). Sample size and multiple regression. *Psychological Methods, 5,* 434–458.
Maxwell, S. E., & Delaney, H. D. (2004). *Designing experiments and analyzing data: A model comparison perspective* (2nd ed.). Mahwah, NJ: Lawrence Erlbaum.
Pinheiro, J. C., & Bates, D. M. (2000). *Mixed-effects models in S and S-PLUS.* New York: Springer.
Pinheiro, J. C., Bates, D., DebRoy, S., & Sarkar, D. (2007). nlme: Linear and nonlinear mixed effects models [Computer software and manual]. Retrieved from http://www.r-project.org
Preece, D. A. (1982). t is for trouble (and textbooks): A critique of some examples of the paired-samples t-test. *The Statistician, 31,* 169–195.
R Development Core Team. (2007a). foreign: Read data stored by Minitab, S, SAS, SPSS, Stata, Systat, dBase [Computer software and manual]. Retrievable from http://www.r-project.org
R Development Core Team. (2007b). R: A language and environment for statistical computing [Computer software and manual]. Vienna, Austria: R Foundation for Statistical Computing. Available from http://www.r-project.org
Sarkar, D. (2006). lattice: Lattice graphics: R package version 0.14-16 [Computer software and manual]. Retrieved from http://cran.r-project.org/
Singer, J. B., & Willett, J. B. (2003). *Applied longitudinal data analysis: Modeling change and event occurrence.* New York: Oxford University Press.
Smithson, M. (2003). *Confidence intervals.* New York: Russell Sage Foundation.
Steiger, J. H. (2004). Beyond the F test: Effect size confidence intervals and tests of close fit in the analysis of variance and contrast analysis. *Psychological Methods, 9*(2), 164–182.
Steiger, J. H., & Fouladi, R. T. (1992). R2: A computer program for interval estimation, power calculation, and hypothesis testing for the squared multiple correlation. *Behavior Research Methods, Instruments, and Computers, 4,* 581–582.
Steiger, J. H., & Fouladi, R. T. (1997). Noncentrality interval estimation and the evaluation of statistical models. In L. L. Harlow, S. A. Mulaik, & J. H. Steiger (Eds.), *What if there were no significance tests?* (pp. 221–257). Mahwah, NJ: Lawrence Erlbaum.

Student. (1908). The probable error of a mean. *Biometrika*, *6,* 1–24.

Thompson, B., Cook, C., & Kyrillidou, M. (2005). Concurrent validity of LibQUAL+™ scores: What do LibQUAL+™ scores measure? *Journal of Academic Librarianship*, *31,* 517–522.

Thompson, B., Cook, C., & Kyrillidou, M. (2006). Using localized survey items to augment standardized benchmarking measures: A LibQUAL+™ study. *Libraries and the Academy*, *6,* 219–230.

Tierney, L. (2005). Some notes on the past and future of Lisp-Stat. *Journal of Statistical Software, 13*(9), 1–15.