Multilevel Modeling in R (2.2)

A Brief Introduction to R, the multilevel package and the nlme package

Paul Bliese (paul.bliese@us.army.mil)

October 28, 2006

Chapters 1 and 2 of this document of this document borrow heavily from *An Introduction to R* (see the copyright notice below)

Table of Contents

1 Introduction

This is an introduction to how R can be used to perform a wide variety of multilevel analyses. "Multilevel analysis" has lately become a term to describe random coefficient modeling (see Bryk & Raudenbush, 1992; Kreft & De leeuw, 1998; Snijders & Bosker, 1999). Without a doubt, random coefficient models (RCM) are well-suited to multilevel analyses; nonetheless, a number of multilevel analytic techniques existed before random coefficient modeling emerged as the tool of choice. In addition, RCM analyses are often augmented by work in related areas such as work in within-group agreement and group-mean reliability. Consequently, the definition of multilevel analyses that I use in this document reflects a wide range of inter-related multilevel topics (see also Klein & Kozlowski, 2000). Specifically, I will cover:

- Within-group agreement and reliability
- Contextual OLS models
- Covariance theorem decomposition
- Random Coefficient Modeling
- Random Group Resampling

Because of the wide variety of topics covered in this definition of multilevel analyses, it is necessary to use several "packages" written for R. The first of these packages is the "base" package that comes with R. This package is automatically loaded and provides the basic structure of R along with routines to estimate ANOVA and regression models important in contextual OLS models.

In addition to the base package, I will rely heavily on a package that I have developed while conducting multilevel analyses – the "multilevel" package. This package provides tools to estimate within-group agreement and reliability; it has routines to conduct Random Group Resampling (Bliese & Halverson, 2002; Bliese, Halverson & Rothberg, 2000); and it has routines to conduct covariance theorem decomposition (Robinson, 1950; Dansereau, Alutto & Yammarino, 1984).

Finally, I will make use of the non-linear and linear mixed-effects (`nlme`) model package, (Pinheiro & Bates, 2000). This package is a powerful set of programs that can be used to estimate a variety of random coefficient models. The programs in the nlme package have remarkable flexibility, allowing excellent control over statistical models.

The layout of this document is as follows. First I briefly introduce R. The material that I discuss in this introduction is in many cases lifted word-for-word from the document entitled "An Introduction to R" (see the copyright notice on page 2). This brief introduction is intended to give readers a feel for R. Following the introduction to R, I illustrate the use of R in multilevel analyses.

2 An Introduction to R

2.1 Overview

R is an integrated suite of software facilities for data manipulation, calculation and graphical display. R is a vehicle for developing methods of interactive data analysis. Among other things it has

- an effective data handling and storage facility,
- a suite of operators for calculations on arrays, in particular matrices,
- a large, integrated collection of tools for data analysis,
- graphical facilities for data analysis and display either directly at the computer or on hardcopy, and
- a well-developed and effective programming language.

The term "environment" is intended to characterize R as a fully planned and coherent system, rather than an incremental growth of specific and inflexible tools, as is frequently the case with other data analysis software.

2.1.1 Related software and documentation

R can be regarded as a re-implementation of the S language developed at AT&T by Rick Becker, John Chambers and Allan Wilks. A number of the books and manuals about S bear some relevance to R.

The basic reference is *The New S Language: A Programming Environment for Data Analysis and Graphics* by Richard A. Becker, John M. Chambers and Allan R. Wilks. The features of the 1991 release of S (S version 3) are covered in *Statistical Models in S* edited by John M. Chambers and Trevor J. Hastie. Both of these texts would be highly useful to users of R.

2.1.2 R and statistics

The developers of R think of it as an environment within which many classical and modern statistical techniques have been implemented. Some of these are built into the base R environment, but many are supplied as packages. There are a number of packages supplied with R (called "standard" packages) and many more are available through the CRAN family of Internet sites (via http://cran.r-project.org).

There is an important difference in philosophy between R and the other main statistical systems. In R a statistical analysis is normally done as a series of steps with intermediate results stored in objects. Thus, whereas SAS and SPSS will give copious output from an analysis, R will give minimal output and store the results in a fit object for subsequent interrogation by functions such as `summary`.

For multilevel analyses, we will be interested primarily in two packages. The first is the `multilevel` package. This package provides routines to estimate within-group agreement and reliability indices; it performs Random Group Resampling (RGR), and also has routines to

conduct covariance theorem decomposition of multilevel correlations. As with all other R packages, the `multilevel` package is open-source and can be obtained from http://cran.r-project.org or installed directly using the "packages" GUI option in R.

The second package that will be used for multilevel random coefficient modeling (i.e., Hierarchical Linear Modeling) is the mixed-effects package `nlme` (Pinheiro & Bates, 2000). This package provides a complete set of resources for estimating random coefficient models. The `nlme` package is a standard package available once R is installed.

Finally, we will make use of both the `foreign` package and the `lattice` package. The former provides functions for importing data files from SAS, SPSS, etc. The latter is an advanced graphical package that allows one to produce production quality graphics.

2.1.3 Starting R in a Windows environment

The CRAN websites and mirrors (http: //cran.r-project.org) provide binary files for installing R in Windows computing environments. The base program and a number of default packages can be downloaded and installed using a single executable file (*.exe).

2.1.4 Data permanency and removing objects

In R, one works in an area called the "workspace." The workspace is a working environment where objects are created and manipulated. Objects that are commonly kept in the workspace are (a) entire data sets (i.e. dataframes) and (b) the output of statistical analyses. It is also relatively common to keep programs (i.e., functions) that do special project-related tasks within the workspace.

The R commands

```
> objects()
```

or

```
> ls()
```

display the names of the objects in the workspace. As given above, the `objects()` command lists the objects in search position 1 corresponding to the workspace (or technically the ".GlobalEnv"). The open and closed parentheses containing no content are a shortcut for `(1)`. It will later become apparent that it is often useful to list objects in other search positions.

Within the workspace, one removes objects using the `rm` function:

```
> rm(x, y, ink, temp, foo)
```

It is important to keep in mind that there are two types of objects listed in the workspace. The first type of object is permanently stored in the ".Rdata" file in the working directory. The second type of object is created during the current session. These latter objects reside entirely in memory unless explicitly written to the ".Rdata" file. In other words, if you fail to save objects that you create in the current session, they will NOT be there next time you start R.

There are two ways to save current objects, both of which use the `save.image` function. First, one can use the "Save Workspace" option from the File menu to specify where to save the workspace. This option is GUI based, and allows the user to use a mouse to specify a location. The other option is to call the `save.image` function directly from the command line, as in:

```
> save.image("F:/Temp/Project 1.RData")
```

In this case, the `save.image` function writes the objects in memory to the "Project 1.Rdata" file in the TEMP subdirectory on the F: Drive. If calling `save.image` directly, it is advisable to end the file name with ".RData" so that R recognizes the file as an R workspace.

2.1.5 Running R for Different Projects

As one develops proficiency with R, the program will inevitably end up being used for multiple projects. It will become necessary, therefore, to keep separate workspaces. Each workspace will likely contain one or more related datasets, model results and programs written for specific projects.

For instance, I often use R to analyze data files for manuscripts that are being written, revised and (theoretically) eventually published. Often because of the length of the review process I may go several months between analyses on specific projects. Consequently, I store the R Workspace in the same location as the manuscript. Therefore, when I return to a revision of a manuscript, the data and statistical models supporting the manuscript are immediately at hand. To save workspaces, follow these steps:

1. Keep your initial workspace empty – no objects
2. Import the raw data (more no this later) and perform the analyses.
3. From the File menu, select "Save Workspace" and save the workspace in a project folder.

By working keeping separate workspaces, all data objects and analysis objects will be available for subsequent analyses and there will be no need to import the data again.

2.1.6 Recall and correction of previous commands

Under Windows, R provides a mechanism for recalling and re-executing previous commands. The vertical arrow keys on the keyboard can be used to scroll forward and backward through a command history. Once a command is located in this way, the cursor can be moved within the command using the horizontal arrow keys, and characters can be removed with the DEL key or added with the other keys.

2.1.7 Getting help with functions and features

R has a built in help facility. To get more information on any specific named function, for example solve, the command is

```
> help(solve)
```

An alternative is

```
> ?solve
```

For a feature specified by special characters, the argument must be enclosed in double or single quotes, making it a "character string":

```
> help("[[")
```

Either form of quote mark may be used to escape the other, as in the string "It's important". Our convention is to use double quote marks for preference.

On most versions of R help is available in html format by running

```
> help.start()
```

to launch a Web browser that allows the help pages to be browsed with hyperlinks.

Searches of help files can by conducted using the `help.search` function. For instance to find functions related to regression one would type:

```
> help.search("regression")
```

2.1.8 R commands, case sensitivity, etc.

Technically R is an expression language with a very simple syntax. It is case sensitive, so “A” and “a” are different symbols and would refer to different variables.

Elementary commands consist of either expressions or assignments. If an expression is given as a command, it is evaluated, printed, and the value is lost. An assignment also evaluates an expression and passes the value to a variable but the result is not automatically printed.

Commands are separated either by a semi-colon (‘;’), or by a new line. Elementary commands can be grouped together into one compound expression by braces (‘{’ .. ‘}’). Comments can be put almost anywhere, starting with a hashmark (‘#’), everything to the end of the line is a comment.

If a command is not complete at the end of a line, R will give a different prompt, by default

```
+
```

on second and subsequent lines and continue to read input until the command is syntactically complete. In providing examples, this document will generally omit the continuation prompt and indicate continuation by simple indenting.

2.2 Simple manipulations; numbers and vectors

2.2.1 Vectors and assignment

R operates on named data structures. The simplest such structure is the numeric vector, which is a single entity consisting of an ordered collection of numbers. To set up a vector named x, say, consisting of five numbers, namely 10.4, 5.6, 3.1, 6.4 and 21.7, use the R command

```
> x <- c(10.4, 5.6, 3.1, 6.4, 21.7)
```

This is an assignment statement using the function `c()` which in this context can take n arbitrary number of vector arguments and whose value is a vector gotten by concatenating its arguments end to end.

A number occurring by itself in an expression is taken as a vector of length one. Notice that the assignment operator (‘`<-`‘) consists of the two characters ‘`<`’ (“less than”) and ‘`-`’(“minus”) occurring strictly side-by-side and it ‘points’ to the object receiving the value of the expression. In current versions of R, assignments can also be made using the = sign.

```
> x=c(10.4, 5.6, 3.1, 6.4, 21.7)
```

Assignments can also be made in the other direction, using the obvious change in the assignment operator. So the same assignment could be made using

```
> c(10.4, 5.6, 3.1, 6.4, 21.7) -> x
```

If an expression is used as a complete command, the value is printed and lost. So now if we were to issue the command

```
> 1/x
```

the reciprocals of the five values would be printed at the terminal (and the value of x, of course, unchanged).

The further assignment

```
> y <- c(x, 0, x)
```

would create a vector y with 11 entries consisting of two copies of x with a zero in the middle place.

2.2.2 Missing values

In some cases the components of a vector may not be completely known. When an element or value is “not available” or a “missing value” in the statistical sense, a place within a vector may be reserved for it by assigning it the special value NA. In general, any operation on an NA becomes an NA. The motivation for this rule is simply that if the specification of an operation is incomplete, the result cannot be known and hence is not available.

Most of the functions in the multilevel package (that we will discuss in detail later) require data that have no missing values. To create such data, one may make use of the `na.exclude` function. The object returned from `na.exclude` is a new dataframe that has listwise deletion of missing values. So

```
> TDATA<-na.exclude(DATA)
```

will produce a dataframe TDATA that contains no missing values. The TDATA dataframe can then be used subsequent analyses. We discuss dataframes in more detail in the next section.

2.3 Dataframes

2.3.1 Introduction to dataframes

A dataframe is an object that stores data. Dataframes have multiple columns representing different variables and multiple rows representing different observations. The columns can be numeric vectors or non-numeric vectors, however each column must have the same number of observations. Thus, for all practical purposes one can consider dataframes to be spreadsheets with the limitation that each column must have the same number of observations.

Dataframes may be displayed in matrix form, and its rows and columns extracted using matrix indexing conventions. This means, for example, that one can access specific rows and columns of a dataframe using brackets [rows, columns]. For example to access rows 1-3 and all columns of a dataframe object named `TDAT`

```
> TDAT[1:3,]
```

To access rows 1:3 and columns 1,5 and 8

```
> TDAT[1:3,c(1,5,8)]
```

We will consider matrix bracket manipulations in more detail with a specific example in section 2.5.

2.3.2 Making dataframes

Data frames can be made using the `data.frame` function. The following example makes a dataframe object called `accountants`.

```
> accountants<-data.frame(home=c("MD","CA","TX"),income=c(45000,
+ 55000,60000),car=c("honda","acura","toyota"))
> accountants
  home income    car
1   MD  45000  honda
2   CA  55000  acura
3   TX  60000 toyota
```

In practice, however, one will generally make dataframes from existing files using data importing functions such as `read.table, read.csv` or `read.spss`. These functions read data sets from external files and create dataframes. We discuss these types of functions in section 2.4.

2.3.3 Using `attach()` and `detach()`

To access specific components of dataframes, we can use the `$` notation. For instance, `accountants$car` returns the `car` vector within the dataframe `accountants`. Sometimes it is useful to make the components of a list or dataframe temporarily visible as variables under their component name, without the need to quote the list name explicitly each time.

The `attach()` function, as well as having a directory name as its argument, may also have a dataframe. Thus

```
> attach(accountants)
```

places the dataframe in the search path at position 2. In this case if there are no variables `home`, `income` or `car` in position 1, then the dataframe `accountants` is searched and `home`, `income` or `car` are available as variables in their own right. In general, I do not recommend attaching specific dataframes just so that one can use short names such as "`car`" instead of the longer names "`accountants$car`". While it is theoretically a time saving option, my experience shows that it can lead to unanticipated consequences when one has fairly complex workspaces with several objects having similar names. Though a little more time consuming, it is better to be explicit about where specific objects are located using the `$` notation.

To detach a dataframe, use

```
> detach()
```

More precisely, this statement detaches from the search path the entity currently at position 2. Entities at positions greater than 2 on the search path can be detached by giving their number to detach, but it is much safer to always use a name, for example by `detach(accountants).`

To make a permanent change to the dataframe itself, the simplest way is to resort once again to the $ notation:

```
> accountants$income2<-accountants$income+100
> accountants
  home income    car income2
1   MD  45000  honda   45100
2   CA  55000  acura   55100
3   TX  60000 toyota   60100
```

2.3.4 Managing the search path

The function `search` shows the current search path and so is a useful way to keep track of what has been attached. Initially, it gives the global environment in search position 1 followed by various packages that are automatically loaded (actual results may vary depending upon the specific version of R).

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

where `.GlobalEnv` is the workspace. Basically, the search path means that if you type in an object such as `car` the program will look for something named `car` first in the workspace, then in the package `methods`, then in the package `stats`, etc. Because `car` does not exist in any of these places, the following error message will be returned:

```
> car
Error: Object "car" not found
```

If one attaches the dataframe `accountants`; however, the search path changes as follows:

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

In this case, typing `car` at the command prompt returns:

```
> car
[1] honda  acura  toyota
Levels: acura honda toyota
```

It is often useful to see what objects exist within various components of the search path. The function `objects()` with the search position of interest in the parentheses can be used to examine the contents of any object in the search path. For instance to see the contexts of search position 2 one types:

```
> objects(2)
[1] "car"     "home"    "income"  "income2"
```

Finally, we detach the dataframe and confirm it has been removed from the search path.

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

2.4 Reading data from files

In R sessions, large data objects will almost always be read from external files and stored as dataframes. There are several options available to read external files.

If variables are stored in spreadsheets such as EXCEL, entire dataframes can be read directly using the function `read.table()` and variants such as `read.csv()` and `read.delim()`. The help file for `read.table()` discusses the variants of `read.table()` in detail.

If variables are stored in other statistical packages such as SPSS or SAS, then the `foreign` package provides some useful programs for importing the data. This document will illustrate importing spreadsheet data and SPSS data.

2.4.1 Reading Spreadsheet (EXCEL) data

External spreadsheets normally have this form.

- The first line of the file has a name for each variable.
- Each additional line of the file has values for each variable.

So the first few lines of a spreadsheet data might look as follows.

UNIT	PLATOON	COH01	COH02	COH03	COH04	COH05
1044B	1ST	4	5	5	5	5
1044B	1ST	3	NA	5	5	5
1044B	1ST	2	3	3	3	3
1044B	2ND	3	4	3	4	4
1044B	2ND	4	4	3	4	4
1044B	2ND	3	3	2	2	1
1044C	1ST	3	3	3	3	3
1044C	1ST	3	1	4	3	4
1044C	2ND	3	3	3	3	3
1044C	2ND	2	2	2	3	2
1044C	2ND	1	1	1	3	3

One of the most reliable ways to import any type of data into R is to use EXCEL to process the data file into a comma delimited (*.csv) format. Note that most statistical packages (SAS, SPSS) can save data as an EXCEL file. Users who use SPSS and export data to EXCEL may encounter the error type value marker "#NULL!" for missing values. This value must be changed to NA as under the second entry under COH02 in the example above to avoid problems in R. In addition, all blank spaces and any other missing value markers should be replaced with NA to facilitate dataframe creation.

Once the comma delimited file is created using the "Save As" feature in EXCEL one can import it into R using either the `read.table()` or the `read.csv()` function. For instance, if the file above is saved as "cohesion.csv" in the root directory of C: (C:\) the function `read.table()` can be used to read the dataframe directly

```
>cohesion<-read.table("c:\\cohesion.csv", "header=T", sep=",")
```

Alternatively, one can use `read.csv()`

```
>cohesion<-read.csv("c:\\cohesion.csv","header=T")
```

Note that subdirectories are designated using the double slash instead of a single slash, also recall that R is case sensitive.

Typing in the name of the cohesion object displays all of the data:

```
> cohesion
    UNIT PLATOON COH01 COH02 COH03 COH04 COH05
1  1044B     1ST     4     5     5     5     5
2  1044B     1ST     3    NA     5     5     5
3  1044B     1ST     2     3     3     3     3
4  1044B     2ND     3     4     3     4     4
5  1044B     2ND     4     4     3     4     4
6  1044B     2ND     3     3     2     2     1
7  1044C     1ST     3     3     3     3     3
8  1044C     1ST     3     1     4     3     4
9  1044C     2ND     3     3     3     3     3
```

```
10 1044C     2ND     2     2     2     3     2
11 1044C     2ND     1     1     1     3     3
```

2.4.2 The extremely useful "clipboard" option

In R, users can directly read and write data to a Windows clipboard. This can be a tremendous time saving feature for it allows users to export and import data into EXCEL and other programs without saving intermediate files.

For instance, to read `cohesion` into R directly from EXCEL, one would:

1. Open the cohesion.xls file in EXCEL
2. Select and copy the relevant cells in Windows
3. Issue the R command:

```
> cohesion<-read.table(file="clipboard",sep="\t",header=T)
```

The file `"clipboard"` instructs `read.table` to read the file from the Windows clipboard, and the separator option of `"\t"` notifies `read.table` that elements are separated by tabs.

Because the "`clipboard`" option also works with `write.table`, (see section 2.4.7) it can be a useful way to export the results of data analyses to EXCEL or other programs. For instance, if we create a correlation matrix from the cohesion data set, we can export this correlation table directly to EXCEL.

```
> CORMAT<-cor(cohesion[,3:7],use="pairwise.complete.obs")
> CORMAT
          COH01     COH02     COH03     COH04     COH05
COH01 1.0000000 0.7329843 0.6730782 0.4788431 0.4485426
COH02 0.7329843 1.0000000 0.5414305 0.6608190 0.3955316
COH03 0.6730782 0.5414305 1.0000000 0.7491526 0.7901837
COH04 0.4788431 0.6608190 0.7491526 1.0000000 0.9036961
COH05 0.4485426 0.3955316 0.7901837 0.9036961 1.0000000

> write.table(CORMAT,file="clipboard",sep="\t",col.names=NA)
```

Going to EXCEL and issuing the "paste" command will put the matrix into the EXCEL worksheet. Note the somewhat counter-intuitive use of `col.names=NA` in this example. This command does not mean omit the column names (that would be achieved by `col.names=F`), instead the command puts an extra blank in the first row of the column names to line up the column names with the correct columns. Alternatively, one can use the option `row.names=F` to omit the row numbers.

2.4.3 The foreign package and SPSS files

Included in current versions of R is the “foreign” package. This package contains functions to import SPSS, SAS, Stata and minitab files.

```
> library(foreign)
```

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

> objects(2)
 [1] "data.restore"  "lookup.xport"  "read.dbf"      "read.dta"
 [5] "read.epiinfo"  "read.mtp"      "read.octave"   "read.S"
 [9] "read.spss"     "read.ssd"      "read.systat"   "read.xport"
[13] "write.dbf"     "write.dta"     "write.foreign"
```

For example, if the data in cohesion is stored in an SPSS sav file in a TEMP directory, then one could issue the following command to read in the data (text following the # mark is a comment):

```
> help(read.spss)     #look at the documentation on read.spss
> cohesion2<-read.spss("c:\\temp\\cohesion.sav")
> cohesion2       #look at the cohesion object
$UNIT
 [1] "1044B" "1044B" "1044B" "1044B" "1044B" "1044B" "1044C" "1044C" "1044C"
[10] "1044C" "1044C"
$PLATOON
 [1] "1ST" "1ST" "1ST" "2ND" "2ND" "2ND" "1ST" "1ST" "2ND" "2ND" "2ND"
$COH01
 [1] 4 3 2 3 4 3 3 3 3 2 1
$COH02
 [1]  5 NA  3  4  4  3  3  1  3  2  1
$COH03
 [1] 5 5 3 3 3 2 3 4 3 2 1
$COH04
 [1] 5 5 3 4 4 2 3 3 3 3 3
$COH05
 [1] 5 5 3 4 4 1 3 4 3 2 3
attr(,"label.table")
attr(,"label.table")$UNIT
NULL
attr(,"label.table")$PLATOON
NULL
attr(,"label.table")$COH01
NULL
attr(,"label.table")$COH02
NULL
attr(,"label.table")$COH03
NULL
attr(,"label.table")$COH04
NULL
attr(,"label.table")$COH05
NULL
```

The `cohesion2` object is stored as a list rather than a dataframe. With the default options, `read.spss` function imports the file as a list and reads information about data labels. In almost every case, users will want to convert the list object into a dataframe for manipulation in R. This can be done using the `data.frame` command.

```
> cohesion2<-data.frame(cohesion2)
> cohesion2
```

```
    UNIT PLATOON COH01 COH02 COH03 COH04 COH05
1  1044B     1ST     4     5     5     5     5
2  1044B     1ST     3    NA     5     5     5
3  1044B     1ST     2     3     3     3     3
4  1044B     2ND     3     4     3     4     4
5  1044B     2ND     4     4     3     4     4
6  1044B     2ND     3     3     2     2     1
7  1044C     1ST     3     3     3     3     3
8  1044C     1ST     3     1     4     3     4
9  1044C     2ND     3     3     3     3     3
10 1044C     2ND     2     2     2     3     2
11 1044C     2ND     1     1     1     3     3
```

Alternatively, users can change the default options in `read.spss` to read the data directly into a dataframe. Note the use of `use.value.labels=F` and `to.data.frame=T` below:

```
> cohesion2<-read.spss("c:\\temp\\cohesion.sav",
use.value.labels=F, to.data.frame=T)
> cohesion2
    UNIT PLATOON COH01 COH02 COH03 COH04 COH05
1  1044B     1ST     4     5     5     5     5
2  1044B     1ST     3    NA     5     5     5
3  1044B     1ST     2     3     3     3     3
4  1044B     2ND     3     4     3     4     4
5  1044B     2ND     4     4     3     4     4
6  1044B     2ND     3     3     2     2     1
7  1044C     1ST     3     3     3     3     3
8  1044C     1ST     3     1     4     3     4
9  1044C     2ND     3     3     3     3     3
10 1044C     2ND     2     2     2     3     2
11 1044C     2ND     1     1     1     3     3
```

The `cohesion` dataframe (made using the EXCEL and csv files) and `cohesion2` (imported from SPSS) are now identical.

2.4.4 Using `choose.files` to bring up a GUI to read data

One limitation with using command lines to specify where files are located is that in complex directory structures it can be hard to specify the correct location of the data. For instance, if data are embedded several layers deep in subdirectories, it may be difficult to specify the path. In these cases, the `choose.files` function is very handy. The `choose.files` function opens a Graphical User Interface (GUI) dialogue box allowing one to select files using the mouse. The `choose.files` function can be embedded within any function where one has to specifically identify a file. So, for instance, one can use `choose.files` with `read.spss`:

```
> cohesion2<-read.spss(choose.files(),
+ use.value.labels=F, to.data.frame=T)
```

Notice how "choose.files()" replaces "c:\\temp\\cohesion.sav" used in the final example in section 2.4.3. With the use of choose.files a GUI dialogue box opens, and one is able to select a specific SPSS sav file using a mouse.

2.4.5 Checking your dataframes with str , summary, and head

With small data sets it is easy to verify that the data has been read in correctly. Often, however, one will be working with large data sets that are too large to visual verify they have been read in correctly. Consequently, functions such as str (structure), summary and head provide easy ways to examine dataframes.

```
> str(cohesion)
`data.frame':   11 obs. of  7 variables:
 $ UNIT   : Factor w/ 2 levels "1044B","1044C": 1 1 1 1 1 1 2 2 2 2 ...
 $ PLATOON: Factor w/ 2 levels "1ST","2ND": 1 1 1 2 2 2 1 1 2 2 ...
 $ COH01  : int  4 3 2 3 4 3 3 3 3 2 ...
 $ COH02  : int  5 NA 3 4 4 3 3 1 3 2 ...
 $ COH03  : int  5 5 3 3 3 2 3 4 3 2 ...
 $ COH04  : int  5 5 3 4 4 2 3 3 3 3 ...
 $ COH05  : int  5 5 3 4 4 1 3 4 3 2 ...

> summary(cohesion)
    UNIT   PLATOON     COH01           COH02           COH03
 1044B:6   1ST:5   Min.   :1.000   Min.   :1.00   Min.   :1.000
 1044C:5   2ND:6   1st Qu.:2.500   1st Qu.:2.25   1st Qu.:2.500
                   Median :3.000   Median :3.00   Median :3.000
                   Mean   :2.818   Mean   :2.90   Mean   :3.091
                   3rd Qu.:3.000   3rd Qu.:3.75   3rd Qu.:3.500
                   Max.   :4.000   Max.   :5.00   Max.   :5.000
                                   NA's   :1.00
     COH04           COH05
 Min.   :2.000   Min.   :1.000
 1st Qu.:3.000   1st Qu.:3.000
 Median :3.000   Median :3.000
 Mean   :3.455   Mean   :3.364
 3rd Qu.:4.000   3rd Qu.:4.000
 Max.   :5.000   Max.   :5.000

> head(cohesion)  #list the first six rows of data in a dataframe
   UNIT PLATOON COH01 COH02 COH03 COH04 COH05
1 1044B     1ST     4     5     5     5     5
2 1044B     1ST     3    NA     5     5     5
3 1044B     1ST     2     3     3     3     3
4 1044B     2ND     3     4     3     4     4
5 1044B     2ND     4     4     3     4     4
6 1044B     2ND     3     3     2     2     1
```

2.4.6 Loading data from packages

One of the useful attributes of R is that the data used in the examples are almost always available to the user. These data are associated with specific packages. For instance, the

multilevel package uses a variety of data files to illustrate specific functions. To gain access to these data, one uses the `data` command:

```
>data(package="multilevel")
```

This command lists the data sets associated with the multilevel package, and the command

```
>data(bhr2000, package="multilevel")
```

copies the `bhr2000` data set to the workspace making it possible to work with the `bhr2000` dataframe.

If a package has been attached by library, its datasets are automatically included in the search, so that

```
>library(multilevel)
```

attaches the multilevel package;

```
>data()
```

lists all of available data sets in the multilevel package and in other packages, and

```
>data(bhr2000)
```

copies the data from the package to the workspace.

2.4.7 Exporting data to spreadsheets using `write()` and `write.table()`

There are likely to be occasions when it is useful to export data from R to spreadsheets. There are two functions that are useful for exporting data -- the `write` function and the `write.table` function. The `write` function is useful when one wants to export a vector while the `write.table` function is useful for exporting dataframes or matrices. Below both will be illustrated.

Let us assume that we were interested in calculating the average hours worked for the 99 companies in the `bh1996` data set, and then exporting these 99 group means to a spreadsheet. To calculate the vector of 99 group means and write them out to a file we can issue the following commands:

```
> HRSMEANS<-tapply(bh1996$HRS,bh1996$GRP,mean)

> write(HRSMEANS,file="c:\\temp\\ghours.txt",ncolumns=1)
```

The `tapply` command subdivides HRS by GRP, and then performs the function `mean` on the HRS data for each group. This command is similar to the `aggregate` function that will be discussed in more detail in section 3.2.2. The `write` function takes the 99 group means stored in the object `HRSMEANS`, and writes them to a file in the "c:\temp" subdirectory called ghours.txt. It is important to use the `ncolumns=1` option or else the `write` function will default to five columns. The ghours.txt file can be read into any spreadsheet as a vector of 99 values.

The `write.table` function is similar to the `write` function, except that one must specify the character value that will be used to distinguish columns. Common choices include tabs (designated as `\t`) and commas. Of these two common choices, commas are likely to be most useful in exporting dataframes or matrices to spreadsheets because programs like Microsoft EXCEL automatically read in comma delimited or csv files. Below I export the entire `bh1996` dataframe to a comma delimited file that can be read directly into Microsoft EXCEL.

```
> write.table(bh1996,file="c:\\temp\\bhdat.csv",sep=",",
row.names=F)
```

Notice the use of the `sep=","` option and also the `row.names=F` option. The `row.names=F` stops the program from writing an additional column of row names typically stored as a vector from 1 to the number of rows. Omitting this column is important because it ensures that the column names match up with the correct columns. Recall from section 2.4.2 that one can use the "`file=clipboard`" option to directly write to Window's clipboard.

2.5 More on using matrix brackets on dataframes

At this point, it may be useful to reconsider the utility of using matrix brackets to access various parts of `cohesion` (see also section 2.3.1). While this may initially appear cumbersome, mastering the use of matrix brackets provides considerable control over ones' dataframe.

Recall that one accesses various parts of the dataframe via [rows, columns]. So, for instance, we can access rows 1,5,and 8 and columns 3 and 4 of the `cohesion` dataframe as follows:

```
> cohesion[c(1,5,8),3:4]
  COH01 COH02
1     4     5
5     4     4
8     3     1
```

Alternatively, we can specify the column names (this helps avoid picking the wrong columns)

```
> cohesion[c(1,5,8),c("COH01","COH02")]
  COH01 COH02
1     4     5
5     4     4
8     3     1
```

It is often useful to pick specific rows that meet some criteria. So, for example, we might want to pick rows that are from the 1ST PLATOON

```
> cohesion[cohesion$PLATOON=="1ST",]
   UNIT PLATOON COH01 COH02 COH03 COH04 COH05
1 1044B     1ST     4     5     5     5     5
2 1044B     1ST     3    NA     5     5     5
3 1044B     1ST     2     3     3     3     3
```

```
7 1044C     1ST     3     3     3     3     3
8 1044C     1ST     3     1     4     3     4
```

Upon inspection, we might want to further refine our choice and exclude missing values. We do this by adding another condition using AND operator "&"

```
> cohesion[cohesion$PLATOON=="1ST"&is.na(cohesion$COH02)==F,]
   UNIT PLATOON COH01 COH02 COH03 COH04 COH05
1 1044B     1ST     4     5     5     5     5
3 1044B     1ST     2     3     3     3     3
7 1044C     1ST     3     3     3     3     3
8 1044C     1ST     3     1     4     3     4
```

By using matrix brackets, one can easily and quickly specify particular portions of a dataframe that are of interest.

2.6 Identifying Statistical models in R

This section presumes the reader has some familiarity with statistical methodology, in particular with regression analysis and the analysis of variance. Almost all statistical models from ANOVA to regression to random coefficient models are specified in a common format. The format is DV ~ IV1+IV2+IV3. In a regression model this dictates that the dependent variable (DV) will be regressed on three independent variables. By using + between the IV's, the model is requesting only main effects. If the IVs were separated by the * sign, it would designate both main effects and interactions (all two and three-way interactions in this case).

2.6.1 Examples

A few examples may be useful in illustrating some other aspects of model specification. Suppose `y`, `x`, `x0`, `x1` and `x2` are numeric variables, and `A`, `B`, and `C` are factors or categorical variables. The following formulae on the left side below specify statistical models as described on the right.

`y ~ x`
`y ~ 1 + x` Both imply the same simple linear regression model of y on x. The first has an implicit intercept term, and the second an explicit one.

`y ~ A` Single classification analysis of variance model of y, with classes determined by A. Basically a one-way analysis of variance.

`y ~ A + x` Single classification analysis of covariance model of y, with classes determined by A, and with covariate x. Basically an analysis of covariance.

2.6.2 Linear models

The basic function for fitting ordinary multiple regression models is `lm()`, and a streamlined version of the call is as follows:

```
> fitted.model <- lm(formula, data = data.frame)
```

For example

```
> fm2 <- lm(y ~ x1 + x2, data = production)
```

would fit a multiple regression model of y on x1 and x2 (with implicit intercept term). The important but technically optional parameter `data = production` specifies that any variables needed to construct the model should come first from the production dataframe. *This is the case regardless of whether the dataframe production has or has not been attached on the search (see section 2.3.3).*

2.6.3 Generic functions for extracting model information

The object created by `lm()` is a fitted model object; technically a list of results of class "lm". Information about the fitted model can then be displayed, extracted, plotted and so on by using generic functions that orient themselves to objects of class "lm". These include:

add1	coef	effects	kappa	predict	residuals
alias	deviance	family	labels	print	step
anova	drop1	formula	plot	proj	summary

A brief description of the most commonly used ones is given below.

`coefficients(object)`
: Extract the regression coefficients.
Short form: `coef(object)`.

`plot(object)`
: Produce four plots, showing residuals, fitted values and some diagnostics.

`predict(object, newdata=data.frame)`
: The dataframe supplied must have variables specified with the same labels as the original. The value is a vector or matrix of predicted values corresponding to the determining variable values in data.frame.

`print(object)`
: Print a concise version of the object. Most often used implicitly.

`residuals(object)`
: Extract the (matrix of) residuals, weighted as appropriate.
Short form: resid(object).

`summary(object)`
: Print a comprehensive summary of the results of the regression analysis. The summary function is widely used to extract more information from objects whether the objects are dataframes or products of statistical functions.

2.7 Graphical procedures

Graphical facilities are an important and extremely versatile component of the R environment. It is possible to use the facilities to display a wide variety of statistical graphs and also to build entirely new types of graphs. The graphics facilities can be used in both interactive and batch modes, but in most cases, interactive use is more productive. Interactive use is also easy because at startup time R initiates a graphics device driver that opens a special graphics window for the display of interactive graphics. Although this is done automatically, it is useful to know that the command used is `windows()` under Windows. Once the device driver is running, R plotting commands can be used to produce a variety of graphical displays and to create entirely new kinds of display.

2.7.1 The `plot()` function

One of the most frequently used plotting functions in R is the `plot()` function. This is a generic function: the type of plot produced is dependent on the type or class of the first argument.

`plot(x, y)` If x and y are vectors, plot(x, y) produces a scatterplot of y against x.

```
plot(df)
plot(~ a+b+c, data=df)
plot(y ~ a+b+c, data=df)
```

where df is a dataframe. The first example produces scatter plots of all of the variables in a dataframe. The second produces scatter plots for just the three named variables (a, b and c). The third example plots y against a, b and c.

2.7.2 Displaying multivariate data

R provides two very useful functions for representing multivariate data. If X is a numeric matrix or dataframe, the command

```
> pairs(X)
```

produces a pairwise scatterplot matrix of the variables defined by the columns of X, that is, every column of X is plotted against every other column of X and the resulting n(n - 1) plots are arranged in a matrix with plot scales constant over the rows and columns of the matrix.

When three or four variables are involved a coplot may be more enlightening. If a and b are numeric vectors and c is a numeric vector or factor object (all of the same length), then the command

```
> coplot(a ~ b | c)
```

produces a number of scatterplots of a against b for given values of c. If c is a factor, this simply means that a is plotted against b for every level of c. When c is numeric, it is divided into a number of conditioning intervals and for each interval a is plotted against b for values of c within

the interval. The number and position of intervals can be controlled with `given.values=` argument to `coplot()` -- the function `co.intervals()` is useful for selecting intervals. You can also use two given variables with a command like

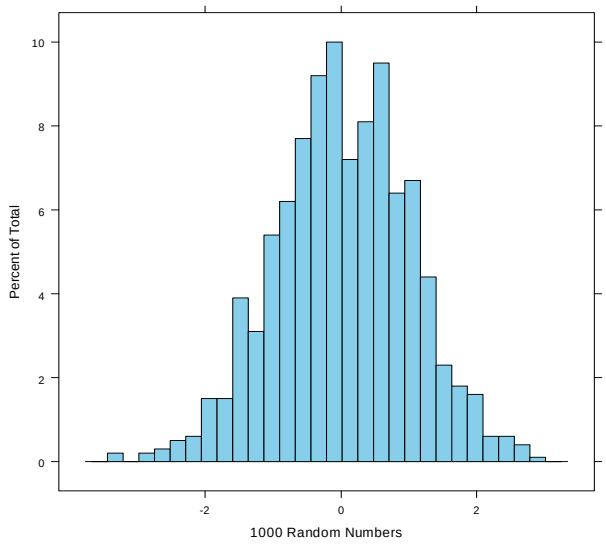
```
> coplot(a ~ b | c + d)
```

which produces scatterplots of a against b for every joint conditioning interval of c and d. The `coplot()` and `pairs()` function both take an argument `panel=` which can be used to customize the type of plot which appears in each panel. The default is `points()` to produce a scatterplot but by supplying some other low-level graphics function of two vectors x and y as the value of `panel=` you can produce any type of plot you wish. An example panel function useful for coplots is `panel.smooth()`.

2.7.3 Advanced Graphics and the `lattice` package

An advanced graphics package called `lattice` is included with the base program. The `lattice` package is an implementation of trellis graphics designed specifically for R. One of the keys to using the `lattice` package is set up an appropriate graphics window in the R session. It is often useful to set up a graphics window that creates graphs with a white or transparent background so that graphics can be copied directly into documents and presentations. Below is an example involving creating a histogram of 1000 random numbers on useful `theme` that involves a transparent background (`col.whitebg`).

```
> library(lattice)
> trellis.device(device="windows",theme="col.whitebg")
> histogram(rnorm(1000),nint=30,xlab="1000 Random Numbers",
 col="sky blue")
```



3 Multilevel Analyses

In the remainder of this document, I illustrate how one can use R in multilevel modeling. I begin by illustrating several R functions that I have found to be particularly useful in preparing the data for subsequent data analysis. After this I illustrate:

- Within-group agreement and reliability
- Contextual OLS models
- Covariance theorem decomposition
- Random Coefficient Modeling

In discussing within-group agreement and the covariance theorem decomposition, I also include sections on Random Group Resampling (or RGR). RGR is a resampling technique that is useful in contrasting actual group results to pseudo-group results (see Bliese & Halverson, 2002; Bliese, Halverson & Rothberg, 2000).

3.1 Attaching the `multilevel` and `nlme` packages

Several of the features in the following sections assume that the `multilevel` and `nlme` packages are accessible in R. Packages are attached in R using the library command. Thus, to attach the multilevel package one issues the command:

```
> library(multilevel)

> library(nlme)
```

The `nlme` package comes with the base R package. The `multilevel` can be obtained from http://cran.r-project.org or installed directly using the "packages" GUI option in R.

3.2 Helpful multilevel data manipulation functions

3.2.1 The `merge` Function

One of the key data manipulation tasks that must be accomplished prior to estimating several of the multilevel models (specifically contextual models and random coefficient models) is that group-level variables must be "assigned down" to the individual. To make a dataframe containing both individual and group-level variables, one typically begins with two separate dataframes. One dataframe contains individual-level data, and the other dataframe contains group-level data. By combining these two dataframes using a group identifying variable common to both, one is able to create a single data set containing both individual and group data. In R, combining dataframes is accomplished using the `merge` function.

For instance, let us consider the `cohesion` data that I introduced when I showed how to read data from external files. The `cohesion` data is included as a multilevel data set, so we can use it without having to use `read.csv` or `read.table` (see section 2.4.1).

```
> data(package="multilevel")
Data sets in package `multilevel':
```

```
bhr2000                 Bliese Halverson and Rothberg (2000)
                        agreement data
bh1996                  Bliese and Halversion (1996) data
cohesion                Platoon Cohesion file
klein2000               Klein et al. (2000) simulation data
univbct                 Univariate form data for growth modeling
                        examples
```

To use the `cohesion` dataframe in the immediate working environment, we issue the `data(cohesion)` command:

```
>data(cohesion)
>cohesion
    UNIT PLATOON COH01 COH02 COH03 COH04 COH05
1  1044B     1ST     4     5     5     5     5
2  1044B     1ST     3    NA     5     5     5
3  1044B     1ST     2     3     3     3     3
4  1044B     2ND     3     4     3     4     4
5  1044B     2ND     4     4     3     4     4
6  1044B     2ND     3     3     2     2     1
7  1044C     1ST     3     3     3     3     3
8  1044C     1ST     3     1     4     3     4
9  1044C     2ND     3     3     3     3     3
10 1044C     2ND     2     2     2     3     2
11 1044C     2ND     1     1     1     3     3
```

Now assume that we have another dataframe with platoon sizes. We can create this dataframe as follows:

```
> group.size<-data.frame(UNIT=c("1044B","1044B","1044C","1044C"),
PLATOON=c("1ST","2ND","1ST","2ND"),PSIZE=c(3,3,2,3))
> group.size  #look at the group.size dataframe
   UNIT PLATOON PSIZE
1 1044B     1ST     3
2 1044B     2ND     3
3 1044C     1ST     2
4 1044C     2ND     3
```

To create a single file (`new.cohesion`) that contains both individual and platoon information, use the `merge` command.

```
> new.cohesion<-merge(cohesion,group.size,
  by=c("UNIT","PLATOON"))
> new.cohesion
      UNIT PLATOON COH01 COH02 COH03 COH04 COH05 PSIZE
1  1044B     1ST     4     5     5     5     5     3
2  1044B     1ST     3    NA     5     5     5     3
3  1044B     1ST     2     3     3     3     3     3
4  1044B     2ND     3     4     3     4     4     3
```

```
5  1044B     2ND     4     4     3     4     4     3
6  1044B     2ND     3     3     2     2     1     3
7  1044C     1ST     3     3     3     3     3     2
8  1044C     1ST     3     1     4     3     4     2
9  1044C     2ND     3     3     3     3     3     3
10 1044C     2ND     2     2     2     3     2     3
11 1044C     2ND     1     1     1     3     3     3
```

Notice that every individual now has a value for `PSIZE` – a value that reflects the number of individuals in the platoon.

3.2.2 The `aggregate` function

In many cases in multilevel analyses, one will be interested in creating a group-level variable from individual responses. For example, one might be interested in calculating the group mean and reassigning it back to the individual. In these cases, the `aggregate` function in combination with the `merge` function is particularly useful. In our cohesion example, for instance, we want to have the platoon means for variables COH01 and COH02 reassigned back to the individuals.

The first step in this process is to create a group-level file. Creating this file is where one uses the `aggregate` function. The `aggregate` function has three key arguments. The first argument is a vector or matrix of variables that one wants to convert to group-level variables. Second is the grouping variable(s) included as a list, and third is the function (`mean`, `var`, `length`, etc.) executed on the variables. To calculate the means of COH01 and COH02 (columns 3 and 4 of the cohesion dataframe) issue the command:

```
>TEMP<-aggregate(cohesion[,3:4],
list(cohesion$UNIT,cohesion$PLATOON),mean)
> TEMP
  Group.1 Group.2    COH01    COH02
1   1044B     1ST 3.000000       NA
2   1044C     1ST 3.000000 2.000000
3   1044B     2ND 3.333333 3.666667
4   1044C     2ND 2.000000 2.000000
```

Notice that COH02 has an "NA" value for the mean. This is because there was a missing value in the individual-level file. If we decide to base the group mean on the non-missing group values we can add the parameter `na.rm=T`, to designate that NA values should be removed prior to calculating the group mean.

```
> TEMP<-aggregate(cohesion[,3:4],
list(cohesion$UNIT,cohesion$PLATOON),mean,na.rm=T)
> TEMP
  Group.1 Group.2    COH01    COH02
1   1044B     1ST 3.000000 4.000000
2   1044C     1ST 3.000000 2.000000
3   1044B     2ND 3.333333 3.666667
```

```
4   1044C     2ND 2.000000 2.000000
```

To merge the TEMP dataframe with the new.cohesion dataframe, we must change the names of the group identifiers in the TEMP frame to match the group identifiers in the new.cohesion dataframe. We also want to change the names of COH01 and COH02 to reflect the fact that they are group means. We will use "G." to designate group mean.

```
> names(TEMP)<-c("UNIT","PLATOON","G.COH01","G.COH02")
```

Finally, we merge TEMP up with new.cohesion to get the complete data set.

```
> final.cohesion<-merge(new.cohesion,TEMP,
by=c("UNIT","PLATOON"))
> final.cohesion
    UNIT PLATOON COH01 COH02 COH03 COH04 COH05 PSIZE  G.COH01  G.COH02
1  1044B     1ST     4     5     5     5     5     3 3.000000 4.000000
2  1044B     1ST     3    NA     5     5     5     3 3.000000 4.000000
3  1044B     1ST     2     3     3     3     3     3 3.000000 4.000000
4  1044B     2ND     3     4     3     4     4     3 3.333333 3.666667
5  1044B     2ND     4     4     3     4     4     3 3.333333 3.666667
6  1044B     2ND     3     3     2     2     1     3 3.333333 3.666667
7  1044C     1ST     3     3     3     3     3     2 3.000000 2.000000
8  1044C     1ST     3     1     4     3     4     2 3.000000 2.000000
9  1044C     2ND     3     3     3     3     3     3 2.000000 2.000000
10 1044C     2ND     2     2     2     3     2     3 2.000000 2.000000
11 1044C     2ND     1     1     1     3     3     3 2.000000 2.000000
```

With the aggregate and merge functions, one has the tools necessary to manipulate data and prepare it for subsequent multilevel analyses (excluding growth modeling which I consider later). Note that I have illustrated a relatively complex situation where there are two levels of nesting (Unit and Platoon). In cases where there is only one grouping variable (for example, UNIT) the commands for aggregate and merge contain the name of a single grouping variable. For instance,

```
>TEMP<-aggregate(cohesion[,3:4],list(cohesion$UNIT),mean,na.rm=T)
```

3.3 Within-Group Agreement and Reliability

The data used in this section are taken from Bliese, Halverson & Rothberg (2000). The examples are based upon the bhr2000 data set from the multilevel package. Thus, the first step is to examine the bhr2000 data set and make it available for analysis.

```
> help(bhr2000)
> data(bhr2000,package="multilevel")#puts data in working environment
> names(bhr2000)
 [1] "GRP"   "AF06"  "AF07"  "AP12"  "AP17"  "AP33"  "AP34"
 "AS14"  "AS15" "AS16"  "AS17"  "AS28"  "HRS"   "RELIG"
> nrow(bhr2000)
[1] 5400
```

The `names` function tells us that there are 14 variables. The first one, `GRP`, is the group identifier. The variables in columns 2 through 12 are individual responses on 11 items that make up a leadership scale. `HRS` represents individuals' reports of work hours, and `RELIG` represents individuals' reports of the degree to which religion is a useful coping mechanism. The `nrow` command indicates that there are 5400 observations. To find out how many groups there are we can use the `length` command in conjunction with the `unique` command

```
> length(unique(bhr2000$GRP))

[1] 99
```

There are several functions in the multilevel library that are useful for estimating and interpreting agreement indices. These functions are `rwg`, `rwg.j`, `rwg.sim`, `rwg.j.sim`, `rwg.j.lindell`, `ad.m`, `ad.m.sim` and `rgr.agree`. The `rwg` function estimates the James, Demaree & Wolf (1984) r_{wg} for single item measures; the `rwg.j` function estimates the James et al. (1984) $r_{wg(j)}$ for multi-item scales. The `rwg.j.lindell` function estimates $r^*_{wg(j)}$ (Lindell, & Brandt, 1997; 1999). The `ad.m` function estimates average deviation (AD) values for the mean or median (Burke, Finkelstein & Dusig, 1999). A series of functions with "`sim`" in the name (`rwg.sim`, `rwg.j.sim` and `ad.m.sim`) allow one to simulate agreement values from a random uniform distribution to test for statistical significance agreement. The simulation functions are based on work by Dunlap, Burke and Smith-Crowe (2003); Cohen, Doveh and Eich (2001) and Cohen, Doveh and Nuham-Shani (in press). Finally, the `rgr.agree` function performs a Random Group Resampling (RGR) agreement test (see Bliese, et al., 2000).

In addition to the agreement measures, there are two multilevel reliability measures, `ICC1` and `ICC2` than can be used on ANOVA models. As Bliese (2000) and others (e.g., Kozlowski & Hattrup, 1992; Tinsley & Weiss, 1975) have noted, reliability measures such as the ICC(1) and ICC(2) are fundamentally different from agreement measures; nonetheless, they often provide complementary information to agreement measures, so in this section, we illustrate the use of each of these functions with the dataframe `bhr2000`.

3.3.1 Agreement: r_{wg}, $r_{wg(j)}$, and $r^*_{wg(j)}$

Both the `rwg` and `rwg.j` functions are based upon the formulations described in James et al. (1984). Both functions require the user to specify three pieces of information. The first piece of information is the variable of interest (`x`), the second is the grouping variable (`grpid`), and third is the estimate of the expected random variance (`ranvar`). The default estimate of `ranvar` is 2, which is the expected random variance based upon the rectangular distribution for a 5-point item (i.e., σ_{EU}^2). See `help(rwg)`, James et al., (1984), or Bliese et al., (2000) for details on selecting appropriate `ranvar` values.

To use the `rwg` function to estimate agreement for the comfort from religion item (`RELIG` in the `bhr2000` dataframe) one would issue the following commands.

```
> RWG.RELIG<-rwg(bhr2000$RELIG,bhr2000$GRP,ranvar=2)
> RWG.RELIG[1:10,]  #examine first 10 rows of data
   grpid        rwg gsize
```

```
1      1 0.11046172    59
2      2 0.26363636    45
3      3 0.21818983    83
4      4 0.31923077    26
5      5 0.22064137    82
6      6 0.41875000    16
7      7 0.05882353    18
8      8 0.38333333    21
9      9 0.14838710    31
10    10 0.13865546    35
```

This returns a dataframe with three columns. The first column contains the group names (`grpid`), the second column contains the 99 r_{wg} estimates – one for each group. The third column contains the group size. To calculate the mean r_{wg} value use the `summary` command:

```
> summary(RWG.RELIG)
     grpid             rwg              gsize
 1      : 1       Min.   :0.0000   Min.   :  8.00
 10     : 1       1st Qu.:0.1046   1st Qu.: 29.50
 11     : 1       Median :0.1899   Median : 45.00
 12     : 1       Mean   :0.1864   Mean   : 54.55
 13     : 1       3rd Qu.:0.2630   3rd Qu.: 72.50
 14     : 1       Max.   :0.4328   Max.   :188.00
 (Other):93
```

The `summary` command informs us that the average r_{wg} value is .186 and the range is from 0 to 0.433. By convention, values at or above 0.70 are considered good agreement, so there appears to be low agreement among individuals with regard to religion. The `summary` command also provides information about the group sizes.

Other useful options might include sorting the values or examining the values in a histogram. Recall that the notation `[,2]` selects all rows and the second column of the `RWG.RELIG` object – the column with the r_{wg} results.

```
> sort(RWG.RELIG[,2])

> hist(RWG.RELIG[,2])
```

To estimate r_{wg} for work hours, we need to change the expected random variance (EV). Work hours was asked using an 11-point item, so EV based on the rectangular distribution (σ_{EU}^2) is 10.00 ($\sigma_{EU}^2=(11^2-1)/12$) – see the `rwg` help file for details).

```
> RWG.HRS<-rwg(bhr2000$HRS,bhr2000$GRP,ranvar=10.00)
> mean(RWG.HRS[,2])
[1] 0.7353417
```

There is apparently much higher agreement about work hours than there was about whether group members received comfort from religion in this sample. By convention, this mean value would indicate agreement because r_{wg} (and $r_{wg(j)}$) values above .70 are considered to provide evidence of agreement.

The use of the `rwg.j` function is nearly identical to the use of the `rwg` function except that the first argument to `rwg.j` is a matrix instead of a vector. In the matrix, each column represents one item in the multi-item scale, and each row represents an individual response. For instance, columns 2-12 in `bhr2000` represent 11 items comprising a leadership scale. The items were assessed using 5-point response options (Strongly Disagree to Strongly Agree), so the expected random variance is 2.

```
> RWGJ.LEAD<-rwg.j(bhr2000[,2:12],bhr2000$GRP,ranvar=2)
> summary(RWGJ.LEAD)
     grpid            rwg.j            gsize      
 1      : 1       Min.   :0.7859   Min.   :  8.00  
 10     : 1       1st Qu.:0.8708   1st Qu.: 29.50  
 11     : 1       Median :0.8925   Median : 45.00  
 12     : 1       Mean   :0.8876   Mean   : 54.55  
 13     : 1       3rd Qu.:0.9088   3rd Qu.: 72.50  
 14     : 1       Max.   :0.9440   Max.   :188.00  
 (Other):93
```

Note that Lindell and colleagues (Lindell & Brandt, 1997, 1999; 2000; Lindell, Brandt & Whitney, 1999) have raised important concerns about the mathematical underpinnings of the $r_{wg(j)}$ formula. Specifically, they note that this formula is based upon the Spearman-Brown reliability estimator. Generalizability theory provides a basis to believe that reliability should increase as the number of measurements increase, so the Spearman-Brown formula is defensible for measures of reliability. There may be no theoretical grounds, however, to believe that generalizability theory applies to measures of agreement. That is, there may be no reason to believe that agreement should increase as the number of measurements increase (but also see LeBreton, James & Lindell, 2005).

To address this potential concern with the $r_{wg(j)}$, Lindell and colleagues have proposed the $r^*_{wg(j)}$. The $r^*_{wg(j)}$ is calculated by substituting the average variance of the items in the scale into the numerator of r_{wg} formula in lieu of using the $r_{wg(j)}$ formula (rwg = 1- Observed Group Variance/Expected Random Variance). Note that Lindell and colleagues also recommend against truncating the Observed Group Variance value so that it matches the Expected Random Variance value in cases where the observed variance is larger than the expected variance. This results in a case where $r^*_{wg(j)}$ values can take on negative values. We can use the function `rwg.j.lindell` to estimate the $r^*_{wg(j)}$ values for leadership.

```
> RWGJ.LEAD.LIN<-rwg.j.lindell(bhr2000[,2:12],
bhr2000$GRP,ranvar=2)
> summary(RWGJ.LEAD.LIN)
     grpid         rwg.lindell         gsize      
```

```
1       : 1       Min.   :0.2502   Min.   :  8.00
10      : 1       1st Qu.:0.3799   1st Qu.: 29.50
11      : 1       Median :0.4300   Median : 45.00
12      : 1       Mean   :0.4289   Mean   : 54.55
13      : 1       3rd Qu.:0.4753   3rd Qu.: 72.50
14      : 1       Max.   :0.6049   Max.   :188.00
(Other):93
```

The average $r^*_{wg(j)}$ value of .43 is considerably lower than the average $r_{wg(j)}$ value of .89 listed earlier.

3.3.2 Significance testing of r_{wg} and $r_{wg(j)}$ using `rwg.sim` and `rwg.j.sim`

As noted in section 3.3.1, r_{wg} and $r_{wg(j)}$ values at or above .70 are conventionally considered providing evidence of within-group agreement. A series of studies by Charnes and Schriesheim (1995); Cohen, Doveh and Eick (2001); Dunlap, Burke, and Smith-Crowe (2003) and Cohen, Doveh and Nahum-Shani (in press) lay the groundwork for establishing tests of statistical significance for r_{wg} and $r_{wg(j)}$. The basic idea behind these simulations is to draw observations from a known distribution (generally a uniform random null), and repeatedly estimate r_{wg} or $r_{wg(j)}$. Because the observations are drawn from a uniform random null, r_{wg} or $r_{wg(j)}$ estimates will frequently be zero. Occasionally, however, the r_{wg} or $r_{wg(j)}$ values will be larger than zero because of the pattern of random numbers drawn. Repeatedly drawing random numbers and estimating r_{wg} and $r_{wg(j)}$ provides a way to calculate expected values and confidence intervals.

The simulations conducted by Cohen et al., (2001) varied a number of parameters, but the two found to be most important for the expected value of the $r_{wg(j)}$ were (a) group size and (b) the number of items. Indeed, Cohen et al., (2001) found that expected $r_{wg(j)}$ values vary considerably as a function of group size and number of items. This implies that the conventional value of .70 may be a reasonable cut-off value for significance with some configurations of group sizes and items, but may not be reasonable for others. Thus, they recommended researchers simulate parameters based on the specific characteristics of the researchers' samples when determining whether $r_{wg(j)}$ values are significant.

In 2003, Dunlap and colleagues estimated 95% confidence intervals for the single item r_{wg} using the idea of simulating null distributions. Their work showed that the 95% confidence interval for the single item measure varied as a function of (a) group size and (b) the number of response options. In the case of 5 response options (e.g., strongly disagree, disagree, neither, agree, strongly agree), the 95% confidence interval estimate varied from 1.00 with a group of 3 to 0.12 for a group of 150. That is, one would need an r_{wg} estimate of 1.00 with groups of size three to be 95% certain the groups agreed more than chance levels, but with groups of size 150 any value equal to or greater than 0.12 would represent significant agreement.

The function `rwg.sim` provides a way to replicate the results presented by Dunlap and colleagues. For instance, to estimate the 95% confidence interval for a group of size 10 on an item with 5 response options one would provide the following parameters to the `rwg.sim` function:

```
> RWG.OUT<-rwg.sim(gsize=10, nresp=5, nrep=10000)
```

```
> summary(RWG.OUT)
$rwg
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0000  0.0000  0.0000  0.1221  0.2167  0.8667

$gsize
[1] 10
$nresp
[1] 5
$nitems
[1] 1
$rwg.95
[1] 0.5277778
```

The results in the preceding example are based on 10,000 simulation runs. In contrast, Dunlap et al., (2003) used 100,000 simulation runs. Nonetheless, both Table 2 from Dunlap et al., (2003) and the example above suggest that 0.53 is the 95% confidence interval estimate for a group of size 10 with five response options. Note that a replication of these results may produce slightly different values.

Because the estimation of r_{wg} in the simulations produces a limited number of possible responses, the typical methods for establishing confidence intervals (e.g., the generic function `quantile`) cannot be used. Thus, the multilevel package provides a `quantile` method for the objects of class `agree.sim` created using `rwg.sim`. To obtain 90%, 95% and 99% confidence interval estimates (or any other values) one would issue the following command:

```
> quantile(RWG.OUT,c(.90,.95,.99))
  quantile.values confint.estimate
1            0.90        0.4222222
2            0.95        0.5277778
3            0.99        0.6666667
```

Cohen et al. (in press) expanded upon the work of Dunlap et al., (2003) and the early work by Cohen et al. (2001) by demonstrating how confidence interval estimation could be applied to multiple item scales in the case of $r_{wg(j)}$ values. The function `rwg.j.sim` is based upon the work of Cohen et al., (in press) and simulates $r_{wg(j)}$ values from a uniform null distribution for user supplied values of (a) group size, (b) number of items in the scale, and (c) number of response options on the items. The user also provides the number of simulation runs (repetitions) upon which to base the estimates. In most cases, the number of simulation runs will be 10,000 or more although the examples illustrated here will be limited to 1,000. The final optional argument to `rwg.j.sim` is `itemcors`. If this argument is omitted, the simulated items used to comprise the scale are assumed to be independent (non-correlated). If the argument is provided, the items comprising the scale are simulated to reflect a given correlational structure. Cohen et al., (2001) showed that results based on independent (non-correlated) items were similar to results based on correlated items; nonetheless, the model with correlated items is more realistic and thereby preferable (see Cohen et al., in press). Estimating

models with a correlational structure requires the MASS package in addition to the multilevel package.

For an example of using rwg.j.sim with non-correlated items, consider a case where a researcher was estimating the expected value and confidence intervals of $r_{wg(j)}$ on a sample where group size was 15 using a 7-item scale with 5 response options for the items (A=5). The call to rwg.j.sim would be:

```
> RWG.J.OUT<-rwg.j.sim(gsize=15,nitems=7,nresp=5,nrep=1000)
> summary(RWG.J.OUT)
$rwg.j
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
0.000000 0.000000 0.009447 0.161800 0.333900 0.713700
$gsize
[1] 15
$nresp
[1] 5
$nitems
[1] 7
$rwg.j.95
[1] 0.5559117
```

In this example, the upper expected 95% confidence interval is 0.56. This is lower than 0.70, and suggests that in this case the value of 0.70 might be too stringent. Based on this simulation, one might justifiably conclude that a value of 0.56 is evidence of significant agreement ($p<.05$). Note that if one replicates this example one will get slightly different results because each run is based on slightly different combinations of randomly generated numbers. Using the simulation, one can show that as group size increases and the number of items increase, the criteria for what constitutes significant agreement decreases.

To illustrate how significance testing of $r_{wg(j)}$ might be used in a realistic setting, we will examine whether group members agreed about three questions specific to mission importance in the lq2002 data set. This data set was also analyzed in Cohen et al., in press. We first begin by estimating the mean $r_{wg(j)}$ for the 49 groups in the sample. Notice that the mean estimate for $r_{wg(j)}$ is .58. This value is below the .70 conventional criteria and suggests a lack of agreement.

```
> RWG.J<-rwg.j(lq2002[,c("TSIG01","TSIG02","TSIG03")],
  lq2002$COMPID,ranvar=2)
> summary(RWG.J)
     grpid        rwg.j            gsize
 10     : 1   Min.   :0.0000   Min.   :10.00
 13     : 1   1st Qu.:0.5099   1st Qu.:18.00
 14     : 1   Median :0.6066   Median :30.00
 15     : 1   Mean   :0.5847   Mean   :41.67
 16     : 1   3rd Qu.:0.7091   3rd Qu.:68.00
 17     : 1   Max.   :0.8195   Max.   :99.00
 (Other):43
```

To determine whether the value of .58 is significant, one can use the `rwg.j.sim` function using item correlations and average group size (41.67 rounded to 42). In this case, notice the simulation suggests that a value of .35 is significant suggesting significant agreement. For illustrations of how the simulations might be used in a group-by-group basis see Cohen et al., (in press).

```
> library(MASS)
> RWG.J.OUT<-rwg.j.sim(gsize=42,nitems=3,nresp=5,
   itemcors=cor(lq2002[,c("TSIG01","TSIG02","TSIG03")]),
   nrep=1000)
> summary(RWG.J.OUT)
$rwg.j
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
0.000000 0.000000 0.007224 0.088520 0.162500 0.548600
$gsize
[1] 42
$nresp
[1] 5
$nitems
[1] 3
$rwg.j.95
[1] 0.346875
```

3.3.3 Average Deviation (AD) Agreement using `ad.m`

Burke, Finkelstein and Dusig (1999) proposed using average deviation (AD) indices as measures of within-group agreement. Cohen et al., in press note that AD indices are also referred to as Mean or Median Average Deviation or MAD. AD indices are calculated by first computing the absolute deviation of each observation from the mean or median. Second, these absolute deviations are averaged to produce a single AD estimate for each group. The formula for AD calculation on a single item is:

$$AD = \Sigma|x_{ij} - X_j|/N$$

where x_{ij} represents an individual observation (i) in group j; X_j represents the group mean or median, and N represents the group size. When AD is calculated on a scale, the AD formula above is estimated for each item on the scale, and each item's AD value is averaged to compute the scale AD score.

AD values are considered practically significant when the values are less than A/6 where A represents the number of response options on the item. For instance, A is 5 when items are asked on a Strongly Disagree, Disagree, Neither, Agree and Strongly Agree format.

The function `ad.m` is used to compute the average deviation of the mean or median. The function requires the two arguments, `x` and `grpid`. The `x` argument represents the item or scale upon which one wants to estimate the AD value. The `ad.m` function determines whether `x` is a vector (single item) or multiple item matrix or data frame (multiple item scale), and performs the AD calculation appropriate for the type of variable. The second function, `grpid`, is a vector

containing the group ids of the `x` argument. The third argument is optional. The default value is to compute the Average Deviation of the mean. The other option is to change the `type` argument to "`median`" and compute the Average Deviation of the median.

For instance, recall that columns 2-12 in `bhr2000` represent 11 items comprising a leadership scale. The items were assessed using 5-point response options (Strongly Disagree to Strongly Agree), so the practical significance of the AD estimate is 5/6 or 0.833. The AD estimates based on the mean for the first five groups and the overall sample in the `bhr2000` data set are provided below:

```
> data(bhr2000)
> AD.VAL <- ad.m(bhr2000[, 2:12], bhr2000$GRP)
> AD.VAL[1:5,]
  grpid      AD.M gsize
1     1 0.8481366    59
2     2 0.8261279    45
3     3 0.8809829    83
4     4 0.8227542    26
5     5 0.8341355    82
> mean(AD.VAL[,2:3])
      AD.M      gsize
 0.8690723 54.5454545
```

Two of the estimates are less than 0.833 suggesting these two groups (2 and 4) agree about ratings of leadership. The overall AD estimate is 0.87, which is also higher than 0.83 and suggests a general lack of agreement.

The AD value estimated using the median instead of the mean, in contrast, suggests practically significant agreement for the sample as a whole.

```
> AD.VAL <- ad.m(bhr2000[, 2:12], bhr2000$GRP,type="median")
> mean(AD.VAL[,2:3])
      AD.M      gsize
 0.8297882 54.5454545
```

To use the `ad.m` function for single item variables such as the work hours (HRS) variable in the bhr2000 data set it is only necessary to provide a vector instead of a matrix as the first argument to the `ad.m` function. Recall the work hours variable is asked on an 11-point response format scale so practical significance is 11/6 or 1.83. The average observed value of 1.25 suggests agreement about work hours.

```
> AD.VAL.HRS <- ad.m(bhr2000$HRS, bhr2000$GRP)
> mean(AD.VAL.HRS[,2:3])
     AD.M     gsize
 1.249275 54.545455
```

3.3.4 Significance testing of AD using `ad.m.sim`

The function `ad.m.sim` is used simulate AD values and test for significance of various combinations of group size, number of response options and number of items in multiple-item scales. The `ad.m.sim` function is similar to the `rwg.sim` and `rwg.j.sim` functions used to test the significance of r_{wg} and $r_{wg(j)}$; however, unlike the functions for the two forms of the r_{wg}, the `ad.m.sim` function works with both single items and multiple-item scales.

The `ad.m.sim` function is based upon the work of Cohen et al. (in press) and of Dunlap et al., (2003). The function simulates AD values from a uniform null distribution for user supplied values of (a) group size, (b) number of items in the scale, and (c) number of response options on the items. Based on Cohen et al. (in press), the final optional parameter allows one to include correlations among items when simulating multiple-item scales. The user also provides the number of simulation runs (repetitions) upon which to base the estimates. In most cases, the number of simulation runs will be 10,000 or more although the examples illustrated here will be limited to 1,000.

To illustrate the `ad.m.sim` function, consider the 11 leadership items in the bhr2000 dataframe. Recall the AD value based on the mean suggested that groups failed to agree about leadership. In contrast, the AD value based on the median suggested that groups agreed. To determine whether the overall AD value based on the mean is statistically significant, one can simulate data matching the characteristics of the bhr2000 sample:

```
  > library(MASS)
  > AD.SIM<-
ad.m.sim(gsize=55,nresp=5,itemcors=cor(bhr2000[,2:12]),
  + type="mean",nrep=1000)
  > summary(AD.SIM)
  $ad.m
     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    1.087   1.182   1.208   1.209   1.236   1.340

  $gsize
  [1] 55

  $nresp
  [1] 5

  $nitems
  [1] 11

  $ad.m.05
  [1] 1.138212

  $pract.sig
  [1] 0.8333333
```

The simulation suggests that any AD mean value less than or equal to 1.14 is statistically significant. Thus, while the AD value for the leadership items (0.87) may not meet the criteria for practical significance, it does for statistical significance. As with the r_{wg} simulation functions, the `ad.m.sim` function has a specifically associated `quantile` function to identify different cut-off points. The example below illustrates how to identify values corresponding to the .90 (.10), .95 (.05) and .99 (.01) significance levels. That is, to be 99% certain that a value was significant, it would need to be smaller than or equal to 1.114.

```
> quantile(AD.SIM,c(.10,.05,.01))
  quantile.values confint.estimate
1            0.10         1.155763
2            0.05         1.138212
3            0.01         1.114170
```

3.3.5 Agreement: Random Group Resampling

The final agreement related function in the multilevel library is `rgr.agree`. In some ways this function is similar to the `rwg.j.sim` function in that it uses repeated simulations of data to draw inferences about agreement. The difference is that the `rgr.agree` function uses the actual group data, while the `rwg.j.sim` function simulates from an expected distribution (the uniform null).

The `rgr.agree` function (a) uses Random Group Resampling to create pseudo groups and calculate pseudo group variances, (b) estimates actual group variances, and (c) performs tests of significance to determine whether actual group and pseudo group variances differ. To use `rgr.agree`, one must provide three variables. The first is a vector representing the variable upon which one wishes to estimate agreement. The second is group membership (`grpid`). The third parameter is the number of pseudo groups that one wants to create.

The third parameter requires a little explanation, because in many cases the number of pseudo groups returned in the output will not exactly match the third parameter. For instance, in our example, we will request 1000 pseudo groups, but the output will return only 990. This is because the `rgr.agree` algorithm creates pseudo groups that are identical in size characteristics to the actual groups. In so doing, however, the algorithm creates sets of pseudo groups in "chunks." The size of each chunk is based upon the size of the number of actual groups. So, for instance, if there are 99 actual groups, then the total number of pseudo groups must be evenly divisible by 99. Nine-hundred-and-ninety is evenly divisible by 99, while 1000 is not. Rather than have the user determine what is evenly divisible by the number of groups, `rgr.agree` will do this automatically. Below is an example of using `rgr.agree` on the work hours variable.

```
> RGR.HRS<-rgr.agree(bhr2000$HRS,bhr2000$GRP,1000)
```

The first step is to create an RGR Agreement object named `RGR.HRS`. The object contains a number of components. In most cases, however, users will be interested in the estimated z-value indicating whether the within-group variances from the actual groups are smaller than the variances from the pseudo groups. A useful way to get this information is to use the `summary`

command. When `summary` is applied to the RGR agreement object it provides standard deviations, variance estimates, an estimate of the z-value, and upper and lower confidence intervals.

```
> summary(RGR.HRS)
$"Summary Statistics for Random and Real Groups"
  N.RanGrps Av.RanGrp.Var SD.Rangrp.Var Av.RealGrp.Var  Z-value
1       990      3.322772      0.762333       2.646583 -8.82554

$"Lower Confidence Intervals (one-tailed)"
    0.5%       1%     2.5%       5%      10%
1.648162 1.795134 1.974839 2.168830 2.407337

$"Upper Confidence Intervals (one-Tailed)"
     90%      95%    97.5%      99%    99.5%
4.251676 4.545078 4.832813 5.642410 5.845143
```

The first section of the summary provides key statistics for contrasting within-group variances from real group with within-group variances from random groups. The second and third sections provide lower and upper confidence intervals. Keep in mind that if one replicates this example one is likely to get slightly different results. This is because the `rgr.agree` function uses a random number generator to create pseudo groups; thus, the results are partially a product of the specific numbers used in the random number generator. While the exact numbers may differ, the conclusions drawn should be nearly identical.

Notice in the first section that although we requested 1000 random groups, we got 990 (for reasons described previously). The first section also reveals that the average within-group variance for the random groups was 3.32 with a Standard Deviation of 0.76. In contrast, the average within-group variance for the real groups was considerably smaller at 2.65. The estimated z-value suggests that, overall, the within-group variances in ratings of work hours from real groups were significantly smaller than the within-group variances from the random groups. This suggests that group members agree about work hours. Recall that a z-value greater than or less than 1.96 signifies significance at $p<.05$, two-tailed.

The upper and lower confidence interval information allows one to estimate whether specific groups do or do not display agreement. For instance, only 5% of the pseudo groups had a variance less than 2.17. Thus, if we observed a real group with a variance smaller than 2.17, we could be 95% confident this group variance was smaller than the variances from the pseudo groups. Likewise, if we want to be 90% confident we were selecting groups showing agreement, we could identify real groups with variances less than 2.41.

To see which groups meet this criterion, use the `tapply` function in conjunction with the `sort` function. The `tapply` function partitions the first variable by the level of the second variable performs the specified function much like the `aggregate` function (see section 3.2.2). Thus, `tapply(HRS,GRP,var)` partitions HRS into separate Groups (GRP), and calculates the variance for each group (`var`). Using `sort` in front of this command simply makes the output easier to read.

```
> sort(tapply(bhr2000$HRS,bhr2000$GRP,var))
       33        43        38        19         6        39        69        17
0.8242754 1.0697636 1.1295681 1.2783251 1.3166667 1.3620690 1.4566667 1.4630282
       20        99        98        44         4        53        61        63
1.5009740 1.5087719 1.5256410 1.5848739 1.6384615 1.6503623 1.6623656 1.7341430
       66        14        76        71        21        18        59        50
1.7354302 1.7367089 1.7466200 1.7597586 1.7808500 1.7916027 1.8112599 1.8666667
       48        60        83         8        22         2        75        11
1.8753968 1.9267300 1.9436796 1.9476190 1.9679144 2.0282828 2.1533101 2.1578947
       96        23        54        47        55        26        74        57
2.1835358 2.1864802 2.2091787 2.2165242 2.2518939 2.2579365 2.2747748 2.2808858
       45        97        64        35        32        41         1        24
2.2975687 2.3386525 2.3535762 2.3563495 2.3747899 2.4096154 2.4284044 2.4391678
       82        37        81        68        42        73        34        25
2.4429679 2.4493927 2.5014570 2.5369458 2.5796371 2.6046154 2.6476418 2.6500000
       93        62        92        12        40        88         5        29
2.6602168 2.7341080 2.7746106 2.7906404 2.7916084 2.8505650 2.8672087 2.8748616
       85        70        77        51         3        13        79        87
2.8974843 2.9938483 3.0084034 3.0333333 3.0764032 3.1643892 3.1996997 3.2664569
        7        95        78        84        46        27        36        15
3.2712418 3.2804878 3.3839038 3.3886048 3.4084211 3.4309008 3.4398064 3.4425287
       89        16        58        49         9        31        90        72
3.4444444 3.4461538 3.4949020 3.5323440 3.6258065 3.6798419 3.8352838 3.9285714
       91        80        86        10        94        28        30        56
3.9565960 3.9729730 3.9753195 4.0336134 4.0984900 4.0994152 4.6476190 4.7070707
       65        52        67
4.7537594 5.2252964 5.3168148
```

If we starting counting from group 33 (the group with the lowest variance of 0.82) we find 46 groups with variances smaller than 2.41. That is, we find 46 groups that have smaller than expected variance using the 90% confidence estimate.

It may also be interesting to see what a "large" variance is when defined in terms of pseudo group variances. This information is found in the third part of the summary of the `RGR.HRS` object. A variance of 4.55 is in the upper 95% of all random group variances. Given this criterion, we have five groups that meet or exceed this standard. In an applied setting, one might be very interested in examining this apparent lack of agreement in groups 30, 56, 65, 52 and 67. That is, one might be interested in determining what drives certain groups to have very large differences in how individuals perceive work hours.

Finally, for confidence intervals not given in the summary, one can use the `quantile` function with the random variances (`RGRVARS`) in the `RGR.HRS` object. For instance to get the lower .20 confidence interval:

```
> quantile(RGR.HRS$RGRVARS, c(.20))
     20%
2.695619
```

Note that `rgr.agree` only works on vectors. Consequently, to use `rgr.agree` with the leadership scale we would need to create a leadership scale score. We can do this using the `rowMeans` function. We will create a leadership scale (`LEAD`) and put it in the `bhr2000` dataframe, so the specific command we issue is:

```
>bhr2000$LEAD<-rowMeans(bhr2000[,2:12])
```

Now that we have created a leadership scale score, we can perform the RGR agreement analysis on the variable.

```
> summary(rgr.agree(bhr2000$LEAD,bhr2000$GRP,1000))

$"Summary Statistics for Random and Real Groups"
  N.RanGrps Av.RanGrp.Var SD.Rangrp.Var Av.RealGrp.Var  Z-value
1       990     0.6011976     0.1317229      0.5156757 -6.46002

$"Lower Confidence Intervals (one-tailed)"
     0.5%        1%      2.5%        5%       10%
0.2701002 0.3081618 0.3605966 0.3939504 0.4432335

$"Upper Confidence Intervals (one-Tailed)"
      90%       95%     97.5%       99%     99.5%
0.7727185 0.8284755 0.8969857 0.9651415 1.0331922
```

The results indicate that the variance in actual groups about leadership ratings is significantly smaller than the variance in randomly created groups (i.e., individuals agree about leadership). For interesting cases examining situations where group members do not agree see Bliese & Halverson (1998a) and Bliese and Britt (2001).

3.3.6 Reliability: ICC(1) and ICC(2)

The multilevel package also contains the reliability functions, `ICC1` and `ICC2`. These two functions are applied to ANOVA models and are used to estimate ICC(1) and ICC(2) as described by Bartko, (1976), James (1982), and Bliese (2000). To use these functions, one first performs a one-way analysis of variance on the variable of interest. For instance, to calculate a one-way analysis of variance on work hours, we issue the `aov` (ANOVA) function from the R base package. Note that in using the `aov` function, we use the `as.factor` function on GRP. The `as.factor` function tells `aov` that GRP (which is numeric in this dataframe) is to be treated as a categorical variable; consequently, R creates N-1 dummy codes in the model matrix (the exact form of the effects coding can be controlled, but will not be discussed in detail here). In the present example, there are 99 groups, so the `as.factor` function results in the creation of 98 dummy coded categories (98 df). Interested readers who estimate the model without the

`as.factor` option will see that GRP erroneously only accounts for 1 df if the `as.factor` command is omitted.

```
> data(bhr2000)
> hrs.mod<-aov(HRS~as.factor(GRP),data=bhr2000)
> summary(hrs.mod)
                 Df  Sum Sq Mean Sq F value    Pr(>F)
as.factor(GRP)   98  3371.4    34.4  12.498 < 2.2e-16 ***
Residuals      5301 14591.4     2.8
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1
```

The `ICC1` and `ICC2` functions are then applied to the aov object.

```
> ICC1(hrs.mod)
[1] 0.1741008
> ICC2(hrs.mod)
[1] 0.9199889
```

Bliese (2000) provides a thorough interpretation of these values, but briefly, the ICC(1) value of .17 indicates that 17% of the variance in individual perceptions of work hours can be "explained" by group membership. The ICC(2) value of .92 indicates that groups can be reliably differentiated in terms of average work hours.

3.3.7 Visualizing an ICC(1) with `graph.ran.mean`

It is often valuable to visually examine the group-level properties of data to see the exact form of the group-level effects. For instance, Levine (1967) notes that a high ICC(1) value can be the product of one or two highly aberrant groups rather than indicating generally shared group properties among the entire sample.

One way to examine the group-level properties of the data is to contrast the observed group means with group means that are the result of randomly assigning individuals to pseudo groups. If the actual group means and the pseudo-group means are identical, there is no evidence of group effects. If one or two groups are clearly different from the pseudo-group distribution it suggests the ICC(1) value is simply caused by a few aberrant observations. If a number of groups have higher than expected means, and a number have lower than expected means, it suggests fairly well-distributed group-level properties.

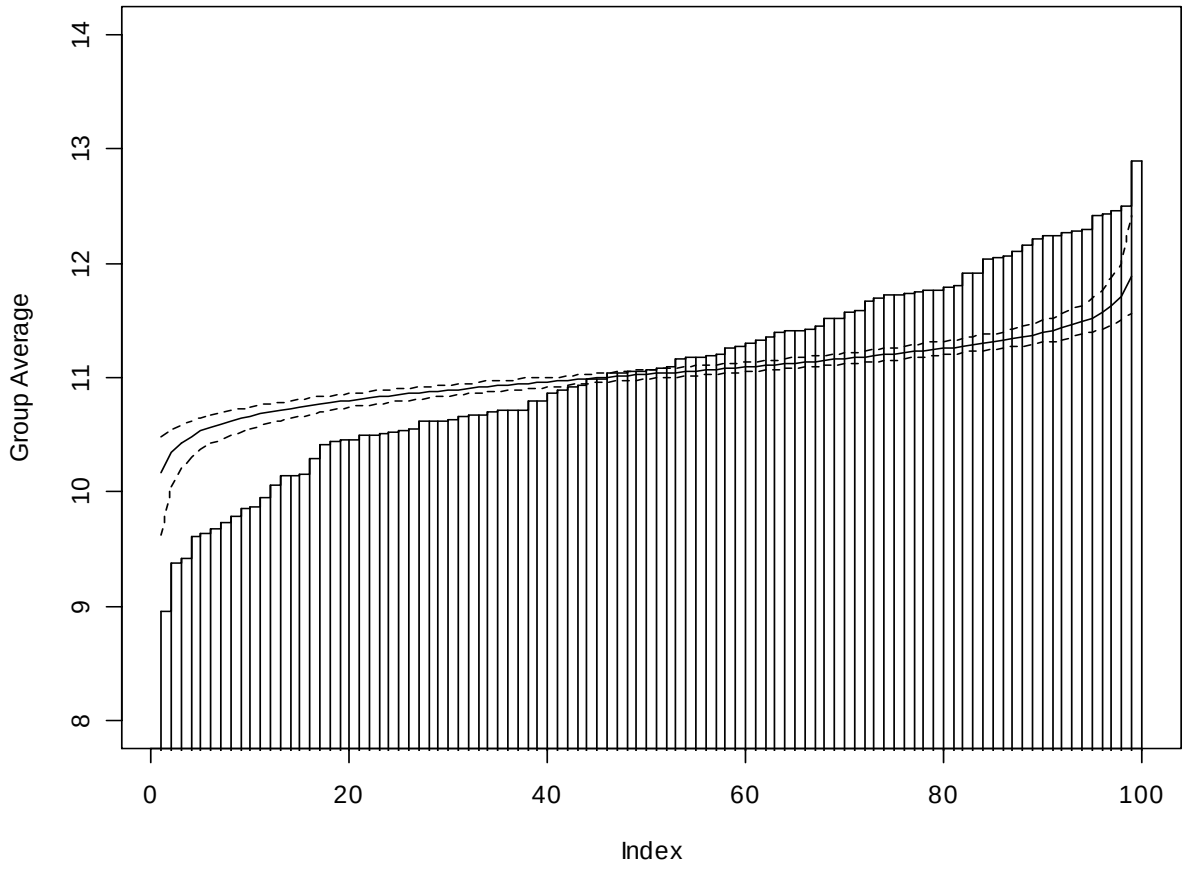
The `graph.ran.mean` function allows one to visually contrast actual group means with pseudo group means. The function requires three parameters. The first is the variable on which one is interested in examining. The second is the group designator, and the third is a smoothing parameter (`nreps`) determining how many sets of pseudo groups should be created to create the pseudo group curve. Low numbers (<10) for this last parameter create a choppy line while high numbers (>25) create smooth lines. In cases where the parameter `bootci` is TRUE (see optional parameters), `nreps` should equal 1000 or more.

Three optional parameters control the y axis limits (`limits`); whether a plot is created (`graph=TRUE`) or a dataframe is returned (`graph=FALSE`); and whether bootstrap confidence intervals are estimated and plotted (`bootci=TRUE`). The default for `limits` is to use the lower 10% and upper 90% values of the raw data. The default for `graph` is to produce a plot, but returning a dataframe can be useful for exporting results to other graphing software. Finally, the default for `bootci` is to return a plot or a dataframe without bootstrap confidence interval estimates.

In the following example, we plot the observed and pseudo group distribution of the work hours variable from the data set bhr2000. Recall, the ICC(1) value for this variable was .17 (see section 3.3.6).

```
> data(bhr2000)
> graph.ran.mean(bhr2000$HRS, bhr2000$GRP, nreps=1000,
limits=c(8,14),bootci=TRUE)
```

The command produced the resulting plot where the bar chart represents each groups' average rating of work hours sorted from highest to lowest, and the line represents a random distribution where 99 pseudo groups (with exact size characteristics of the actual groups) were created 100 times and the sorted values were averaged across the 1000 iterations. The dotted lines represent the upper and lower 95% confidence interval estimates. In short, the line represents the expected distribution if there were no group-level properties associated with these data. The graph suggests fairly evenly distributed group-level properties associated with the data. That is, the ICC(1) value of .17 does not seem to be caused by one or two aberrant groups.



3.4 Regression and Contextual OLS Models

Prior to the introduction of multilevel random coefficient models, OLS regression models were widely used to detect contextual effects. Firebaugh (1978) provides a good methodological discussion of these types of contextual models as does Kreft and DeLeeuw (1998) and James and Williams (2000).

The basic logic behind these models is that an aggregated group mean can explain unique variance over and above an individual variable of the same name. So, for instance, Bliese (2002) found that average group work hours explained unique variance in individual well-being over-and-above individual reports of work hours. This occurs because there is no mathematical reason why the group-level relationship between means must be the same as the individual-level relationship between raw variables. When the slope of the group-mean relationship differs from the slope of the individual-level relationship, a contextual effect is present (Firebaugh, 1978).

To estimate contextual regression models in R, one uses the OLS regression function `lm` to simultaneously test the significance of the individual and group mean variable. If the group-mean variable is significant it indicates the individual-level and group-level slopes are significantly different, and one has evidence of a contextual effect (Firebaugh, 1978; Snijders & Bosker, 1999). As discussed in the next section, there is an important caveat. Specifically, the

standard error associated with the group-level effect is almost always too small producing tests that are too liberal. For this reason random coefficient models (RCM) are a preferred way to identify contextual effects.

3.4.1 Contextual Effect Example

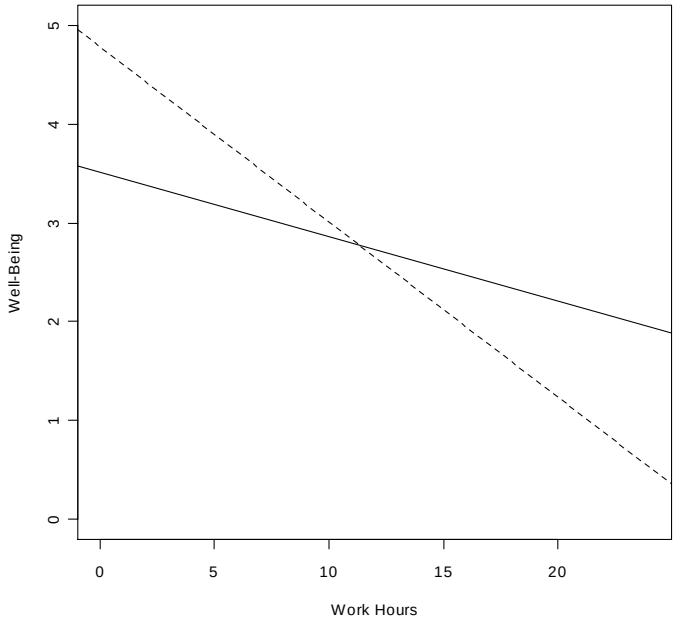
In this example we use the `bh1996` dataframe to illustrate the estimation of a contextual model. The `bh1996` dataframe has group mean variables included; however, we will pretend that it does not so we can illustrate the use of the `aggregate` and `merge` functions.

```
> data(bh1996)
> names(bh1996)
 [1] "GRP"      "COHES"    "G.COHES"  "W.COHES"  "LEAD"     "G.LEAD"
 [7] "W.LEAD"   "HRS"      "G.HRS"    "W.HRS"    "WBEING"   "G.WBEING"
[13] "W.WBEING"
> TDAT<-bh1996[,c(1,8,11)]  # a dataframe with GRP, HRS and WBEING
> names(TDAT)
[1] "GRP"    "HRS"    "WBEING"
> TEMP<-aggregate(TDAT$HRS,list(TDAT$GRP),mean,na.rm=T)
> names(TEMP)
[1] "Group.1" "x"
> names(TEMP)<-c("GRP","G.HRS")
> TBH1996<-merge(TDAT,TEMP,by="GRP")  #merge group and individual data
> names(TBH1996)
[1] "GRP"    "HRS"    "WBEING" "G.HRS"
> tmod<-lm(WBEING~HRS+G.HRS,data=TBH1996) #estimate the linear model
> summary(tmod,cor=F)
Call:
lm(formula = WBEING ~ HRS + G.HRS, data = TBH1996)
Residuals:
     Min       1Q   Median       3Q      Max
-2.87657 -0.57737  0.03755  0.64453  2.37267
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  4.783105   0.136395  35.068   <2e-16 ***
HRS         -0.046461   0.004927  -9.431   <2e-16 ***
G.HRS       -0.130836   0.013006 -10.060   <2e-16 ***
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1
Residual standard error: 0.8902 on 7379 degrees of freedom
Multiple R-Squared: 0.0398,     Adjusted R-squared: 0.03954
F-statistic: 152.9 on 2 and 7379 DF,  p-value:     0
```

Notice that `G.HRS` is significant with a t-value of –10.060. This provides evidence of significant contextual effects. If we want to examine the form of the relationship, we can plot the regression slopes for the two models using the following commands:

```
     > plot(TBH1996$HRS,TBH1996$WBEING,xlab="Work Hours",ylab="Well-
Being",type="n") #type = n omits the points which is important since
we have 7,382 observations
     > abline(lm(WBEING~HRS,data=TBH1996)) # plots the individual-
level slope
     > abline(lm(WBEING~G.HRS,data=TBH1996),lty=2) #group-level slope
```

This produces the plot provided below. Notice that the group-mean slope (the dotted line) is considerably steeper than the individual slope (the solid line).



While contextual models are valuable, a major limitation with them is that they do not account for the fact that individuals are nested within groups. In essence, the models "pretend" that individual observations are independent instead of acknowledging that responses from individuals might be more similar than would be expected by chance. For instance, individual responses on well-being are somewhat influenced by group membership (as we will show later). This has the effect of biasing the standard errors, and making one a little too likely to detect contextual effects. Specifically, it is likely that the standard error of 0.013 associated with G.HRS is too small. This in turn makes the t-value too large. Better models, such as random coefficient models, account for this non-independence. We will illustrate the estimation of these in section 3.6. For more details on the effects of non-independence see Bliese (2002); Bliese and Hanges (2004); Kenny and Judd, (1986) and Snijders and Bosker, (1999).

3.5 Correlation Decomposition and the Covariance Theorem

OLS contextual models provide a way of determining whether or not regression slopes based on group means differ from regression slopes of individual-level variables. The covariance theorem provides a way of doing a similar thing for correlations nested in a two-level structure. Essentially, the covariance theorem allows one to break down a raw correlation into two separate components – the portion of the raw correlation attributable to within-group (individual) processes, and the portion of the correlation attributable to between-group (group-level) processes.

Robinson (1950) was one of the first researchers to propose the covariance theorem, but Dansereau and colleagues increased the visibility of the theorem by incorporating it into an analysis system they labeled WABA for Within-And-Between-Analyses (Dansereau, Alutto &

Yammarino, 1984). WABA is actually two integrated procedures, WABA I and WABA II. WABA I uses a set of decision tools based on eta values to inform decisions about the individual or group-level nature of the data. Eta values, however, are highly influenced by group size, but WABA I makes no group size adjustments; consequently, there is little value in WABA I unless one is working with dyads (see Bliese, 2000; Bliese & Halverson, 1998b). Arguably a more useful way of drawing inferences from eta-values is to contrast eta-values from actual groups to eta-values from pseudo groups. We will illustrate this in a Random Group Resampling extension of the covariance theorem decomposition (see section 3.5.2).

3.5.1 The `waba` and `cordif` functions

Dansereau et al.'s (1984) WABA II revolves around the estimation of the covariance theorem components, and the `waba` function in the multilevel library provides the covariance theorem components for the relationship between two variables. For example, to decompose the correlation between work hours and well-being into the between-group and within-group component we would issue the following command. Note that for comparative purposes we use the same data as we did in OLS contextual model example (section 3.4.1).

```
> waba(bh1996$HRS,bh1996$WBEING,bh1996$GRP)
$Cov.Theorem
     RawCorr     EtaBX     EtaBY      CorrB     EtaWX     EtaWY      CorrW
1 -0.1632064 0.3787881 0.2359287 -0.7121729 0.9254834 0.9717704 -0.1107031
$n.obs
[1] 7382
$n.grps
[1] 99
```

The `waba` function returns a list with three elements. The first element is the covariance theorem with all its components. The second element is the number of observations used in the estimate of the covariance theorem. The third element is the number of groups. The latter two elements should routinely be examined because the `waba` function, by default, performs listwise deletion of missing values.

This formula shows that the raw correlation of -.163=(EtaBX*EtaBY*CorrB) + (EtaWX*EtaWY*CorrW) or (.379*.236*-.712)+(.925*.972*-.111). Everything in the first set of parentheses represents the between-group component of the correlation, and everything in the second set of parentheses represents the within-group component of the correlation.

The group-mean correlation of -.71 definitely looks larger than the within-group correlation of -.11. Furthermore, since these two correlations are independent, we can contrast them using the `cordif` function. This function performs an r to z' transformation of the two correlations (see also the `rtoz` function) and then tests for differences between the two z' values using the formula provided in Cohen and Cohen (1983, p. 54). There are four arguments that must be provided to `cordif`. These are (1) the first correlation of interest, (2) the second correlation of interest, (3) the N on which the first correlation is based, and (4) the N on which the second correlation is based. In our example, we already have the two correlations of interest (-.13 and -.66); to get the N for the between-group correlation, we need to know the number of groups. We can get this N by determining how many unique elements there are in GRP.

```
> length(unique(bh1996$GRP))
```

```
[1] 99
```

The N for the within-group correlation is slightly more complicated. It is calculated as the total N minus the number of groups (see Dansereau, et al., 1984). In our example, we already know that the total N is 7,382 from the `waba` function output. We also know that the number of groups is 99. Thus, the N for the within-group correlation is 7,382-99 or 7,283. For illustrative purposes, however, we will use the `nrow` function to get the number of observations.

```
> nrow(bh1996)-99
[1] 7283
```

With this information, we have all the necessary components for the `cordif` function.

```
> cordif(-.1107,-.7122,7283,99)
$"z value"
[1] 7.597172
```

The z-value is larger than 1.96, so we conclude that the two correlations are significantly different for each other. That is, the between-group correlation is significantly larger than the within-group correlation. This finding mirrors what we found in our contextual analysis. Note that the within-group correlation is based on X and Y deviation scores. These deviation scores are estimated by subtracting the group mean of X from X, and the group mean of Y from Y. In random coefficient modeling, these deviation scores are also called group-mean centered scores.

3.5.2 Random Group Resampling of Covariance Theorem (`rgr.waba`)

As noted above, it may be interesting to see how the eta-between, eta-within, between group and within-group correlations vary as a function of the group-level properties of the data. To do this, one can use the `rgr.waba` function. Essentially, the `rgr.waba` function allows one to answer questions such as "is my eta-between value for x larger than would be expected by chance?" The `rgr.waba` routine randomly assigns individuals into pseudo groups having the exact size characteristics as the actual groups, and then calculates the covariance theorem parameters. By repeatedly assigning individuals to pseudo groups and re-estimating the covariance theorem components, one can create sampling distributions of the covariance theorem components to see if actual group results differ from pseudo group results (see Bliese & Halverson, 2002). Below I illustrate the use of `rgr.waba`. Note that this is a very computationally intensive routine, so it may take some time to complete. For comparative purposes, I begin by re-estimating the covariance theorem components using the first 1000 observations.

```
> TDAT<-bh1996[1:1000,c(1,8,11)]
> waba(TDAT$HRS,TDAT$WBEING,TDAT$GRP) #Model for first 1000 obs
     RawCorr     EtaBX    EtaBY      CorrB     EtaWX     EtaWY      CorrW
1 -0.1500598 0.4136304 0.192642 -0.6302504 0.9104449 0.9812691 -0.1117537

> RGR.WABA<-rgr.waba(TDAT$HRS,TDAT$WBEING,TDAT$GRP,1000)
> round(summary(RGR.WABA),dig=4)
       RawCorr     EtaBX     EtaBY     CorrB     EtaWX    EtaWY     CorrW
NRep 1000.0000 1000.0000 1000.0000 1000.0000 1000.0000 1.00e+03 1000.0000
Mean   -0.1501    0.1236    0.1241   -0.1409    0.9921 9.92e-01   -0.1501
SD      0.0000    0.0209    0.0217    0.2463    0.0026 2.80e-03    0.0040
```

The `summary` of the `rgr.waba` object produces a table giving the number of random repetitions, the means and the standard deviations from analysis. Notice the raw correlation has a standard deviation of zero because it does not change. In contrast, the between-group correlation has the highest standard deviation (.25) indicating that it varied across pseudo group runs. It is apparent that all of covariance theorem components in the actual groups significantly vary from their counterparts in the pseudo group analysis. This is obvious because most actual group components are close to two standard deviations different from the pseudo group means. To test for significant differences in this resampling design, however, one can simply look at the sampling distribution of the random runs, and use the 2.5% and 97.5% sorted values to approximate 95% confidence intervals. Any values outside of this range would be considered significantly different from their pseudo group counterparts. To estimate the 95% confidence intervals we can use the `quantile` function.

```
> quantile(RGR.WABA,c(.025,.975))
           EtaBX      EtaBY      CorrB     EtaWX     EtaWY      CorrW
2.5%  0.08340649 0.08288485 -0.6048007 0.9861588 0.9857920 -0.1585368
97.5% 0.16580367 0.16797054  0.3613034 0.9965156 0.9965591 -0.1417005
```

Notice that all of the covariance theorem values based on the actual groups are outside of the 95% confidence interval estimates. That is, all of the actual group results are significantly different than would be expected by chance ($p<.05$). If we estimate the 99% confidence intervals we find that the between-group correlation is no longer outside of the 99% confidence interval, but the other values are.

```
> quantile(RGR.WABA,c(.005,.995))
           EtaBX      EtaBY      CorrB     EtaWX     EtaWY      CorrW
0.5%  0.07280037 0.07128845 -0.7216473 0.9843644 0.9831655 -0.1608020
99.5% 0.17614418 0.18271719  0.4825655 0.9973465 0.9974557 -0.1386436
```

Keep in mind in estimating the `rgr.waba` models that one's results are likely to differ slightly from those presented here because of the random generation process underlying random group sampling.

3.6 Multilevel Random Coefficient modeling

In this section, I illustrate the estimation of multilevel random coefficient (MRC) models using the `nlme` package (Pinhiero & Bates, 2000). Most of the examples described in this section are taken from Bliese (2002) and use the Bliese and Halverson (1996) data set (`bh1996`) included in the multilevel library. In describing the models, I use the notation from Bryk & Raudenbush (1992).

While a complete description of MRC modeling is beyond the scope of document, I will provide a short overview. For more detailed discussions see Bliese, (2002); Bryk and Raudenbush, (1992); Hofmann, (1997); Hox (2002); Kreft and De leeuw, (1998) and Snidjers and Bosker (1999).

One can think of MRC models as ordinary regression models that have additional variance terms for handling group membership effects. The key to understanding MRC models is understanding how group membership can lead to additional sources of variance in ones model.

The first variance term that distinguishes a MRC model from a regression model is a term that allows groups to differ in their mean values (intercepts) on the dependent variable. When this variance term, τ_{00}, is non-zero, it suggests that groups differ on the dependent variable. When groups differ by more than chance levels one can potentially model why some groups have high average DV values while other groups have low average DV values. One predicts group-mean differences with group-level variables. These are variables that differ across groups, but do not differ within-groups. Group-level variables are often called "level-2" variables. For example, a cohesion measure that is the same across all members of the same group would be a level-2 variable, and a level-2 cohesion variable might be related to the average level of well-being in a group.

The second variance term (or really class of terms) that distinguishes a MRC model from a typical regression model is the term that allows slopes between independent and dependent variables to differ across groups (τ_{11}). Single-level regression models generally assume that the relationship between the independent and dependent variable is constant across groups. In contrast, MRC models allow the slope to vary from one group to another. If slopes randomly vary, one can attempt to explain this slope variation as a function of group differences – again, one uses level-2 variables such as cohesion to explain why the slopes within some groups are stronger than the slopes within other groups.

A third variance term is common to both MRC and regression models. This variance term, σ^2, reflects the degree to which an individual score differs from its predicted value within a specific group. One can think of σ^2 as an estimate of within-group variance. One uses individual-level or level-1 variables to predict within-group variance, σ^2. Level-1 variables differ among members of the same group. For instance, a level-1 variable such as self-efficacy would vary among members of the same group.

In summary, in a complete MRC analysis, one wants to know (1) what level-1 factors are related to the within-group variance σ^2?; (2) what group-level factors are related to the between-group variation in intercepts τ_{00}?; and (3) what group-level factors are related to within-group slope differences, τ_{11}? In the next sections, I re-analyze portions of the Bliese and Halverson data set to illustrate a typical sequence of steps that one might use in multilevel modeling.

3.6.1 Steps in multilevel modeling

Step 1. Because multilevel modeling involves predicting variance at different levels, one typically begins a multilevel analysis by determining the levels at which significant variation exists. In the case of the two-level model (the only models that I will consider here), one generally assumes that there is significant variation in σ^2 – that is, one assumes that within-group variation is present. One does not necessarily assume, however, that there will be significant intercept variation (τ_{00}) or between-group slope variation (τ_{11}). Therefore, one typically begins by examining intercept variability (see Bryk & Raudenbush, 1992; Hofmann, 1997). If τ_{00} does not differ by more than chance levels, there may be little reason to use random coefficient modeling since simpler OLS modeling will suffice. Note that if slopes randomly vary even if

intercepts do not, there may still be reason to estimate random coefficient models (see Snijders & Bosker, 1999).

In Step 1 of a MRCM analysis, one explores the group-level properties of the outcome variable to determine three things: First, what is the ICC(1) (commonly referred to simply as the ICC in random coefficient models) associated with the outcome variable. That is, how much of the variance in the outcome can be explained by group membership. Second, one examines whether the group means of the outcome variable are reliable. By convention, one would like the group mean reliability to be around .70 because this indicates that groups can be reliably differentiated (see Bliese, 2000). Third, one wants to know whether the variance of the intercept (τ_{00}) is significantly larger than zero.

These three aspects of the outcome variable are examined by estimating an unconditional means model. An unconditional means model does not contain any predictors, but includes a random intercept variance term for groups. This model essentially looks at how much variability there is in mean Y values (i.e., how much variability there is in the intercept) relative to the total variability. In the two stage HLM notation, the model is:

$$Y_{ij} = \beta_{0j} + r_{ij}$$
$$\beta_{0j} = \gamma_{00} + u_{0j}$$

In combined form, the model is: $Y_{ij} = \gamma_{00} + u_{0j} + r_{ij}$. This model states that the dependent variable is a function of a common intercept γ_{00}, and two error terms: the between-group error term, u_{0j}, and the within-group error term, r_{ij}. The model essentially states that any Y value can be described in terms of an overall mean plus some error associated with group membership and some individual error. In the null model, one gets two estimates of variance; τ_{00} for how much each groups' intercept varies from the overall intercept (γ_{00}), and σ^2 for how much each individuals' score differs from the group mean. Bryk and Raudenbush (1992) note that this model is directly equivalent to a one-way random effects ANOVA – an ANOVA model where one predicts the dependent variable as a function of group membership.

The unconditional means model and all other random coefficient models that we will consider are estimated using the `lme` (for linear mixed effects) function in the `nlme` package (see Pinheiro & Bates, 2000). There are two formulas that must be specified in any `lme` call: a fixed effects formula and a random effects formula.

In the unconditional means model, the fixed portion of the model is γ_{00} (an intercept term) and the random component is $u_{0j} + r_{ij}$. The random portion of the model states that intercepts will be allowed to vary among groups. We begin the analysis by attaching the multilevel package (which also loads the nlme package) and making the `bh1996` data set in the multilevel package available for analysis.

```
> library(multilevel)
> library(nlme)
> data(bh1996)
> Null.Model<-lme(WBEING~1,random=~1|GRP,data=bh1996)
```

In the model, the fixed formula is `WBEING~1`. This states that the only predictor of well-being is an intercept term. The random formula is `random=~1|GRP`. This specifies that the intercept can vary as a function of group membership. This is the simplest random formula that one will

encounter, and in many situations a random intercept model may be all that is required to adequately account for the nested nature of the grouped data.

Estimating ICC. The unconditional means model provides between-group and within-group variance estimates in the form of τ_{00} and σ^2, respectively. As with the ANOVA model, it is often valuable to determine how much of the total variance is between-group variance. This can be accomplished by calculating the Intraclass Correlation Coefficient (ICC) using the formula: ICC = $\tau_{00}/(\tau_{00} + \sigma^2)$ (see, Bryk & Raudenbush, 1992; Kreft & De Leeuw, 1998). Bliese (2000) notes that the ICC is equivalent to Bartko's ICC(1) formula (Bartko, 1976) and to Shrout and Fleiss's ICC(1,1) formula (Shrout & Fleiss, 1979). To get the estimates of variance for an `lme` object, one uses the `VarCorr` function.

```
> VarCorr(Null.Model)
GRP = pdSymm(1)
            Variance   StdDev
(Intercept) 0.03580079 0.1892110
Residual    0.78949727 0.8885366
> 0.03580079/(0.03580079+0.78949727)
[1] 0.04337922
```

The estimate of τ_{00} (between-group variance or Intercept) is 0.036, and the estimate of σ^2 (within-group variance or Residual) is 0.789. The ICC estimate ($\tau_{00}/(\tau_{00} + \sigma^2)$) is .04.

To verify that the ICC results from the random coefficient modeling are similar to those from an ANOVA model and the `ICC1` function (see section 0) one can perform an ANOVA analysis on the same data.

```
> tmod<-aov(WBEING~as.factor(GRP),data=bh1996)
> ICC1(tmod)
[1] 0.04336905
```

The ICC value from the random coefficient model and the ICC(1) from the ANOVA model are basically identical.

Estimating Group-Mean Reliability. When exploring the properties of the outcome variable, it can also be of interest to examine the reliability of the group mean. The reliability of group means often affects one's ability to detect emergent phenomena. In other words, a prerequisite for detecting emergent relationships at the aggregate level is to have reliable group means (Bliese 1998). By convention, one strives to have group mean reliability estimates around .70. Group mean reliability estimates are a function of the ICC and group size (see Bliese, 2000; Bryk & Raudenbush, 1992). The `GmeanRel` function from the multilevel package calculates the ICC, the group size, and the group mean reliability for each group.

When we apply the `GmeanRel` function to our `Null.Model` based on the 99 groups in the `bh1996` data set, we are interested in two things. First, we are interested in the average reliability of the 99 groups. Second, we are interested in determining whether or not there are specific groups that have particularly low reliability.

```
> Null.Model<-lme(WBEING~1,random=~1|GRP,data=bh1996)
```

```
> GREL.DAT<-GmeanRel(Null.Model)
> names(GREL.DAT)
[1] "ICC"      "Group"    "GrpSize"  "MeanRel"
> GREL.DAT$ICC      #ICC estimate
[1] 0.04337922
> GREL.DAT$MeanRel
 [1] 0.7704119 0.7407189 0.8131975 0.6557120 0.8222325
 [6] 0.5594125 0.5680426 0.6065741 0.6387944 0.7466758
[11] 0.6387944 0.6201282 0.7996183 0.8099782 0.7860071
[16] 0.6759486 0.8116016 0.7860071 0.6557120 0.7437319
[21] 0.8066460 0.6661367 0.7839102 0.8131975 0.5920169
[26] 0.7210397 0.8222325 0.6065741 0.7245244 0.6134699
[31] 0.6557120 0.6852003 0.5843267 0.8178269 0.8066460
[36] 0.7940029 0.6896308 0.7174657 0.6610045 0.8131975
[41] 0.7376341 0.6610045 0.8193195 0.7061723 0.7727775
[46] 0.8207878 0.6557120 0.7407189 0.7795906 0.5680426
[51] 0.6201282 0.6265610 0.5994277 0.7407189 0.7137989
[56] 0.7750949 0.8163095 0.7437319 0.7959093 0.8099782
[61] 0.7022044 0.8207878 0.6939384 0.7022044 0.7704119
[66] 0.7376341 0.8099782 0.6661367 0.5994277 0.8193195
[71] 0.7860071 0.4048309 0.6502517 0.7604355 0.7279232
[76] 0.7959093 0.6852003 0.7523651 0.7210397 0.6939384
[81] 0.8964926 0.7210397 0.9110974 0.8795291 0.8788673
[86] 0.9088937 0.8863580 0.7860071 0.8277854 0.9100090
[91] 0.8083266 0.8379118 0.8886532 0.8330020 0.8250530
[96] 0.6661367 0.7551150 0.4204716 0.5504306
> mean(GREL.DAT$MeanRel)  #Average group-mean reliability
[1] 0.7335212
```

Notice that the overall group-mean reliability is acceptable at .73, but that several groups have quite low reliability estimates. Specifically, group 71 and group 98 have reliability estimates below .50.

We can show that the group-mean reliability from the random coefficient model is equivalent to the ICC(2) from the ANOVA model by using the `bh1996` data to estimate the ICC(2) in an ANOVA framework (see section 0.).

```
> tmod<-aov(WBEING~as.factor(GRP),data=bh1996)
> ICC2(tmod)
[1] 0.7717129
```

In this case the ICC(2) estimate from the ANOVA model is slightly higher than the group-mean reliability estimate from the random coefficient model. This occurs because group sizes are unequal. If all the groups were the same size, then the two measures would be nearly identical.

With reference to ICC(2) values and group-mean reliability, note that there are alternate ways of estimating group-mean reliability. Snijders and Bosker (1999) show, for example, that one can estimate overall group-mean reliability by determining what percentage of the total group variance is made up by τ_{00}.

Finally, keep in mind that the estimates of within-group and between-group variance from the random coefficient model will be nearly identical to those from the ANOVA model as long as restricted maximum likelihood estimation (REML) is used in the random coefficient modeling (this is the default in the `lme` routine of the `nlme` package). If full maximum likelihood is used, the variance estimates may differ somewhat from the ANOVA estimates particularly in small sample situations. In our running example, the use of REML versus full maximum likelihood makes little difference. Interested readers may calculate ICC values from an `lme` model with maximum likelihood to verify this result.

```
> mod.ml<-lme(WBEING~1,random=~1|GRP,data=bh1996,method="ML")
> VarCorr(mod.ml)
GRP = pdLogChol(1)
            Variance   StdDev
(Intercept) 0.03531699 0.1879282
Residual    0.78949525 0.8885354
```

The maximum likelihood estimate of the ICC is 0.042, and is very similar to the 0.043 REML estimate.

Determining whether τ_{00} is significant. Returning to our original analysis involving well-being from the `bh1996` data set, we might be interested in knowing whether the intercept variance (i.e.,τ_{00}) estimate of 0.036 is significantly different from zero. To do this we compare –2 log likelihood values between (1) a model with a random intercept, and (2) a model without a random intercept.

A model without a random intercept is estimated using the `gls` function in the `nlme` package. The –2 log likelihood values for an `lme` or `gls` object are obtained using the `logLik` function and multiplying this value by –2. If the –2 log likelihood value for the model with random intercept is significantly larger than the model without the random intercept (based on a Chi-square distribution), then one concludes that the model with the random intercept fits the data significantly "better" than does the model without the random intercept. In the `nlme` package, model contrasts via –2 log likelihood values are facilitated by using the `anova` function.

```
> Null.Model.2<-gls(WBEING~1,data=bh1996)
> logLik(Null.Model.2)*-2
`log Lik.' 19536.17 (df=2)
> logLik(Null.Model)*-2
`log Lik.' 19347.34 (df=3)
> 19536.17-19347.34
[1] 188.83
> anova(Null.Model,Null.Model.2)
             Model df      AIC      BIC    logLik   Test  L.Ratio p-value
Null.Model       1  3 19353.34 19374.06 -9673.669
Null.Model.2     2  2 19540.17 19553.98 -9768.084 1 vs 2 188.8303  <.0001
```

The –2 log likelihood value for the `gls` model without the random intercept is 19536.17. The –2 log likelihood value for the model with the random intercept is 19347.34. The difference of 188.8 is significant on a Chi-Squared distribution with one degree of freedom (one model

estimated a random intercept, the other did not, and this results in the one df difference). These results suggest that there is significant intercept variation.

In summary, we would conclude that there is significant intercept variation in terms of general well-being scores across the 99 Army companies in our sample. We also estimate that 4% of the variation in individuals' well-being score is a function of the group to which he or she belongs. Thus, a model that allows for random variation in well-being among Army companies is better than a model that does not allow for this random variation.

Step 2. At this point in our example we have two sources of variation that we can attempt to explain in subsequent modeling – within-group variation (σ^2) and between-group intercept (i.e., mean) variation (τ_{00}). In many cases, these may be the only two sources of variation we are interested in explaining so let us begin by building a model that predicts these two sources of variation.

To make things interesting, let us assume that individual well-being is related to individual reports of work hours. We expect that individuals who report high work hours will report low well-being. At the same time, however, let us assume that average work hours in an Army Company are related to the average well-being of the Company over-and-above the individual-level work-hours and well-being relationship. Using Hofmann and Gavin's (1998) terminology, this means that we are testing an incremental model where the level-2 variable predicts unique variance after controlling for level-1 variables. This is also directly equivalent to the contextual model that we estimated in section 3.4.1.

The form of the model using Bryk and Raudenbush's (1992) notation is:

$$WBEING_{ij} = \beta_{0j} + \beta_{1j}(HRS_{ij}) + r_{ij}$$
$$\beta_{0j} = \gamma_{00} + \gamma_{01}(G.HRS_j) + u_{0j}$$
$$\beta_{1j} = \gamma_{10}$$

Let us consider each row of the notation. The first row states that individual well-being is a function of the groups' intercept plus a component that reflects the linear effect of individual reports of work hours plus some random error. The second line states that each groups' intercept is a function of some common intercept (γ_{00}) plus a component that reflects the linear effect of average group work hours plus some random between-group error. The third line states that the slope between individual work hours and well-being is fixed—it is not allowed to randomly vary across groups. Stated another way, we assume that the relationship between work hours and well-being is identical in each group.

When we combine the three rows into a single equation we get an equation that looks like a common regression equation with an extra error term (u_{0j}). This error term indicates that WBEING intercepts (i.e., means) can randomly differ across groups. The combined model is:

$$WBEING_{ij} = \gamma_{00} + \gamma_{10}(HRS_{ij}) + \gamma_{01}(G.HRS_j) + u_{0j} + r_{ij}$$

This model is specified in `lme` as:

```
> Model.1<-lme(WBEING~HRS+G.HRS,random=~1|GRP,data=bh1996)

> summary(Model.1)
Linear mixed-effects model fit by REML
 Data: bh1996
```

```
       AIC      BIC   logLik
  19222.28 19256.81 -9606.14

Random effects:
 Formula: ~1 | GRP
        (Intercept)  Residual
StdDev:   0.1163900 0.8832353

Fixed effects: WBEING ~ HRS + G.HRS
               Value  Std.Error   DF   t-value p-value
(Intercept)  4.740829 0.21368746 7282 22.185808  <.0001
HRS         -0.046461 0.00488798 7282 -9.505056  <.0001
G.HRS       -0.126926 0.01940357   97 -6.541368  <.0001
 Correlation:
      (Intr) HRS
HRS    0.000
G.HRS -0.965 -0.252

Standardized Within-Group Residuals:
        Min          Q1         Med          Q3         Max
-3.35320562 -0.65024982  0.03760797  0.71319835  2.70917777

Number of Observations: 7382
Number of Groups: 99
```

Notice that work hours are significantly negatively related to individual well-being. Furthermore after controlling the individual-level relationship, average work hours (G.HRS) are related to the average well-being in a group.

At this point one can also estimate how much of the variance was explained by these two predictors. Because individual work hours were significantly related to well-being, we expect that it will have "explained" some of the within-group variance σ^2. Similarly, since average work hours were related to the group well-being intercept we expect that it will have "explained" some of intercept variance, τ_{00}. Recall that in the null model, the variance estimate for the within-group residuals, σ^2, was 0.789; and the variance estimate for the intercept, τ_{00}, was 0.036. The `VarCorr` function on the `Model.1` object reveals that each variance component has changed slightly.

```
> VarCorr(Model.1)
GRP = pdSymm(1)
            Variance   StdDev
(Intercept) 0.01354663 0.1163900
Residual    0.78010466 0.8832353
```

Specifically, the variance estimates from the model with the two predictors are 0.780 and 0.014. That is, the variance of the within-group residuals decreased from 0.789 to 0.780 and the variance of the between-group intercepts decreased from 0.036 to 0.014. We can calculate the percent of variance explained by using the following formula:

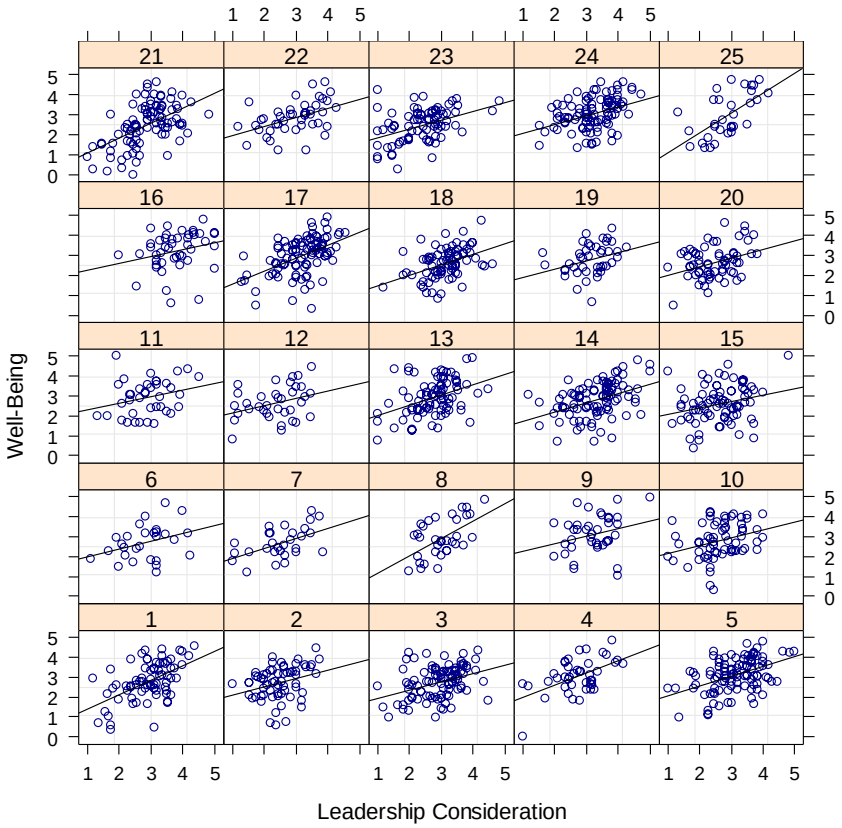
Variance Explained = 1 – (Var with Predictor/Var without Predictor)

To follow through with our example, work hours explained 1 – (0.780/0.789) or 0.011 (1%) of the within-group variance in σ^2, and group-mean work hours explained 1 – (0.014/0.036) or 0.611 (61%) of the between-group intercept variance τ_{00}. While the logic behind variance estimates appears pretty straightforward (at least in models without random slopes), the variance estimates should be treated with some degree of caution because they are partially dependent upon how one specifies the models. Interested readers are directed to Snijders and Bosker (1994; 1999) for an in-depth discussion of variance estimates.

Step 3. Let us continue our analysis by trying to explain the third source of variation, namely, variation in our slopes (τ_{11}, τ_{12}, etc.). To do this, let us examine another variable from the Bliese and Halverson (1996) data set. This variable represents Army Company members' ratings of leadership consideration (LEAD). Generally individual soldiers' ratings of leadership are related to well-being. In this analysis, however, we will consider the possibility that the strength of the relationship between individual ratings of leadership consideration and well-being varies among groups.

We begin by examining slope variation among the first 25 groups. Visually we can do this using `xyplot` from the `lattice` package.

```
> library(lattice)
> trellis.device(device="windows",theme="col.whitebg")
> xyplot(WBEING~LEAD|as.factor(GRP),data=bh1996[1:1582,],
  type=c("p","g","r"),col="dark blue",col.line="black",
  xlab="Leadership Consideration",
  ylab="Well-Being")
```



From the plot of the first 25 groups in the `bh1996` data set, it seems likely that there is some slope variation. The plot, however, does not tell us whether or not this variation is significant. Thus, the first thing to do is to determine whether the slope variation differs by more than chance levels.

Is slope variation significant? We begin our formal analysis of slope variability by adding leadership consideration to our model and testing whether or not there is significant variation in the leadership consideration and well-being slopes across groups. The model that we test is:

$$WBEING_{ij} = \beta_{0j} + \beta_{1j}(HRS_{ij}) + \beta_{2j}(LEAD_{ij}) + r_{ij}$$
$$\beta_{0j} = \gamma_{00} + \gamma_{01}(G.HRS_j) + u_{0j}$$
$$\beta_{1j} = \gamma_{10}$$
$$\beta_{2j} = \gamma_{20} + u_{2j}$$

The last line of the model includes the error term u_{2j}. This term indicates that the leadership consideration and well-being slope is permitted to randomly vary across groups. The variance term associated with u_{2j} is τ_{12}. It is this variance term that interests us in the cross-level interaction hypothesis. Note that we have not permitted the slope between individual work hours and individual well-being to randomly vary across groups.

In combined form the model is: $WBEING_{ij} = \gamma_{00} + \gamma_{10}(HRS_{ij}) + \gamma_{20}(LEAD_{ij}) + \gamma_{01}(G.HRS_j) + u_{0j} + u_{2j} * LEAD_{ij} + r_{ij}$. In R this model is designated as:

```
> Model.2<-lme(WBEING~HRS+LEAD+G.HRS,random=~LEAD|GRP, data=bh1996)
> summary(Model.2)
  Linear mixed-effects model fit by REML
   Data: bh1996
         AIC      BIC   logLik
    17838.58 17893.83 -8911.29

  Random effects:
   Formula: ~LEAD | GRP
   Structure: General positive-definite, Log-Cholesky parametrization
              StdDev    Corr
  (Intercept) 0.3794891 (Intr)
  LEAD        0.1021935 -0.97
  Residual    0.8008079

  Fixed effects: WBEING ~ HRS + LEAD + G.HRS
                   Value  Std.Error   DF   t-value p-value
  (Intercept)  2.4631348 0.20832607 7281 11.823459  <.0001
  HRS         -0.0284776 0.00446795 7281 -6.373764  <.0001
  LEAD         0.4946550 0.01680846 7281 29.428928  <.0001
  G.HRS       -0.0705047 0.01789284   97 -3.940387   2e-04
  ...

Number of Observations: 7382
Number of Groups: 99
```

In line with our expectations, leadership consideration is significantly related to well-being. What we are interested in from this model, however, is whether τ_{12}, the slope between leadership consideration and well-being significantly varies across groups. To determine whether the slope is significant, we test the –2 log likelihood ratios between a model with and a model without a random slope for leadership consideration and well-being. We have already estimated a model with a random slope. To estimate a model without a random slope we use `update` on `Model.2` and change the random statement so that is only includes a random intercept.

```
> Model.2a<-update(Model.2,random=~1|GRP)
> anova(Model.2,Model.2a)
         Model df      AIC      BIC    logLik   Test  L.Ratio p-value
Model.2      1  8 17838.58 17893.83 -8911.290
Model.2a     2  6 17862.68 17904.12 -8925.341 1 vs 2 28.10254  <.0001
```

The difference of 28.10 is significant on <u>two</u> degrees of freedom. Note that there are two degrees of freedom because the model with the random slope also estimates a covariance term for the slope-intercept relationship. The log likelihood results indicate that model with the random effect for the leadership consideration and well-being slope is significantly better than the model without this random effect. This indicates significant slope variation.

Now we know we have significant variation in the leadership and well-being slope, we can attempt to see what group-level properties are related to this variation. In this example, we hypothesize that when groups are under a lot of strain from work requirements, the relationship

between leadership consideration and well-being will be relatively strong. In contrast, when groups are under little strain, we expect a relatively weak relationship between leadership consideration and well-being. We expect these relationships because we believe that leadership is relatively unimportant in terms of individual well-being when groups are under little stress, but that the importance of leadership consideration increases when groups are under high stress. We are, in essence, proposing a contextual effect in an occupational stress model (see Bliese & Jex, 2002).

A proposition such as the one that we presented in the previous paragraph represents a cross-level interaction. Specifically, it proposes that the slope between leadership consideration and well-being within groups varies as a function of a level-2 variable, namely group work demands. In random coefficient modeling, we test this hypothesis by examining whether a level-2 variable explains a significant amount of the level-1 slope variation among groups. In our example, we will specifically be testing whether average work hours in the group "explains" group-by-group variation in the relationship between leadership consideration and well-being. In Bryk and Raudenbush's (1992) notation, the model that we are testing is:

$$
\begin{aligned}
WBEING_{ij} &= \beta_{0j} + \beta_{1j}(HRS_{ij}) + \beta_{2j}(LEAD_{ij}) + r_{ij} \\
\beta_{0j} &= \gamma_{00} + \gamma_{01}(G.HRS_j) + u_{0j} \\
\beta_{1j} &= \gamma_{10} \\
\beta_{2j} &= \gamma_{20} + \gamma_{21}(G.HRS_j) + u_{2j}
\end{aligned}
$$

In combined form the model is:

$WBEING_{ij} = \gamma_{00} + \gamma_{10}(HRS_{ij}) + \gamma_{20}(LEAD_{ij}) + \gamma_{01}(G.HRS_j) + \gamma_{21}(LEAD_{ij} * G.HRS_j) + u_{0j} + u_{2j} * LEAD_{ij} + r_{ij}$.

In `lme` we specify the cross-level interaction by adding an interaction term between leadership (LEAD) and average group work hours (G.HRS). Specifically, the model is:

```
> Final.Model<-lme(WBEING~HRS+LEAD+G.HRS+LEAD:G.HRS,
random=~LEAD|GRP,data=bh1996)
> round(summary(Final.Model)$tTable,dig=3)
             Value Std.Error   DF t-value p-value
(Intercept)  3.654     0.726 7280   5.032   0.000
HRS         -0.029     0.004 7280  -6.391   0.000
LEAD         0.126     0.217 7280   0.578   0.564
G.HRS       -0.175     0.064   97  -2.751   0.007
LEAD:G.HRS   0.032     0.019 7280   1.703   0.089
```

The `tTable` results from the final model indicate there is a significant cross-level interaction (the last row using a liberal p-value of less than .10). This result indicates that average work hours "explained" a significant portion of the variation in τ_{12} – the vertical cohesion and well-being slope.

We can examine the form of our interaction by predicting four points – high and low group work hours and high and low leadership consideration. We start by selecting values for G.HRS and LEAD that are one standard deviation above the mean and one standard deviation below the

mean. By using the Group Work Hours variable in the original data set, we have means and standard deviation values weighted by group size.

```
 > mean(bh1996$G.HRS)
[1] 11.2987
> sd(bh1996$G.HRS)
[1] 0.8608297
> 11.30-.86; 11.30+.86
[1] 10.44
[1] 12.16

> mean(bh1996$LEAD)
[1] 2.890665
> sd(bh1996$LEAD)
[1] 0.771938
> 2.89-.77; 2.89+.77
[1] 2.12
[1] 3.66
```

Once we have the high and low values we create a small data set (TDAT) with high and low values for the interactive variables, and mean values for the non-interactive variables (individual work hours in this case). We then use the `predict` function to get estimates of the outcome given the values of the variables.

```
> TDAT<-data.frame(HRS=c(11.2987,11.2987,11.2987,11.2987),
                   LEAD=c(2.12,2.12,3.66,3.66),
                   G.HRS=c(10.44, 12.16, 10.44, 12.16),
                   GRP=c(1,1,1,1))
> predict(Final.Model,TDAT,level=1)
       1        1        1        1
2.380610 2.198103 3.217337 3.120810
```

The predicted values in this case are specifically for GRP 1. Each group in the sample will have different predicted values because the slopes and intercepts randomly vary among groups. In many cases, one will not be specifically interested in the predicted values for specific groups, but interested in the patterns for the sample as a whole. If one is interested in estimating overall values, one can change the level of prediction to `level=0`.

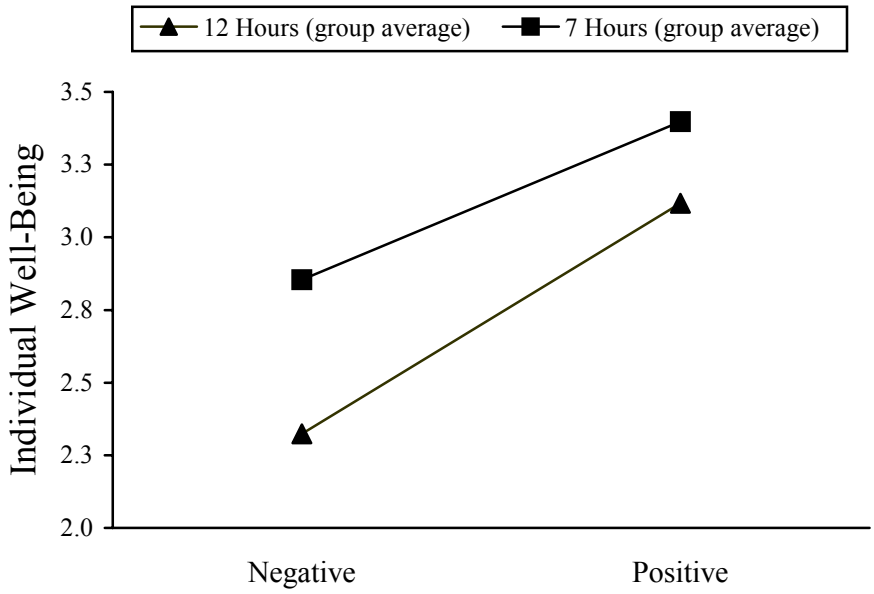
```
> predict(Final.Model,TDAT,level=0)
[1] 2.489508 2.307001 3.204766 3.108239
attr(,"label")
[1] "Predicted values"
```

Notice that the values for the sample as a whole differ from those for GRP 1.

When the values are plotted, the form of the interaction supports our proposition; however, to make the effect more dramatic I selected group work hour values of 7 and 12 to represent low and high average work hours and have plotted the predictions from these values. Note this plot was generated in PowerPoint.

```
> TDAT<-data.frame(HRS=c(11.2987,11.2987,11.2987,11.2987),
+                  LEAD=c(2.12,2.12,3.66,3.66),
+                  G.HRS=c(7, 12, 7, 12),
```

```
+                    GRP=c(1,1,1,1))
> predict(Final.Model,TDAT,level=0)
[1] 2.854523 2.323978 3.397820 3.117218
attr(,"label")
[1] "Predicted values"
```



Individual Ratings of Leadership Consideration

Soldiers' perceptions of leadership consideration are positively related to their well-being regardless of the number of hours that the group, on average, works; however, the relationship between leadership consideration and well-being is stronger (steeper slope) in groups with high work hours than in groups with low work hours. Another way to think about the interaction is to note that well-being really drops (in relative terms) when one perceives that leadership is low in consideration and one is a member of a group with high work hours. This supports our proposition that considerate leadership is relatively more important in a high work demand context.

In this model one can also estimate how much of the variation in the slopes is "explained" by the group work hours. The estimate of the between group slope variance, τ_{12}, in the model with a random slope for the relationship between leadership and well-being (Model.2) is 0.0104.

```
> VarCorr(Model.2)
GRP = pdLogChol(LEAD)
            Variance   StdDev    Corr
(Intercept) 0.14401197 0.3794891 (Intr)
LEAD        0.01044352 0.1021935 -0.97
Residual    0.64129330 0.8008079
```

The estimate after average work hours has "explained" some of the slope variance (Final.Model) is 0.0095.

```
> VarCorr(Final.Model)
GRP = pdLogChol(LEAD)
            Variance    StdDev     Corr
(Intercept) 0.131260632 0.36229909 (Intr)
LEAD        0.009545556 0.09770136 -0.965
Residual    0.641404947 0.80087761
```

Thus, average group work hours accounts for 1 – (0.0095/0.0104) or 8.6% of the slope variance. Once again, I emphasize that this is a rough estimate, and I direct readers to Snijders and Bosker (1994; 1999) for additional information on estimating effect sizes.

3.6.2 Some Notes on Centering

In multilevel modeling, one will eventually have to contend with centering issues. In our examples, we have simply used raw, untransformed variables as predictors. In some cases, though, there may be good reasons to consider centering the variables. Basically, there are two centering options with level-1 variables.

Level-1 variables such as leadership can be grand-mean centered or group-mean centered. Grand-mean centering is often worth considering because doing so helps reduce multicollinearity among predictors and random effect terms. In cases where interactive terms are included, grand-mean centering can be particularly helpful in reducing correlations between main-effect and interactive terms. Hofmann and Gavin (1998) and others have shown that grand-mean centered and raw variable models are basically identical; however, grand-mean centered models will often converge in situations where a model based on raw variables will not. The computational efficiency of grand-mean centered models is due entirely to reductions in multicollinearity because the computer algorithms tend have trouble converging when correlations among variables become too high.

Grand-mean centering can be accomplished in one of two ways. The explicit way is to subtract the overall mean from the raw variable. The less obvious way is to use the `scale` function. The `scale` function is typically used to standardize (mean=0, sd=1) variables, but can also be used to grand-mean center. Below I create grand-mean centered variables for leadership both ways.

```
> bh1996$GRAND.CENT.LEAD<-bh1996$LEAD-mean(bh1996$LEAD)
> bh1996$GRAND.CENT.LEAD<-scale(bh1996$LEAD,scale=F)
```

In the first example a single scalar element (the mean of leadership) is recycled and subtracted from each element in the vector of leadership scores to create a new variable. In the second example, the use of the option `scale=F` instructs `scale` to provide a grand-mean centered variable.

Group-mean centering is another centering option with level-1 variables. In group-mean centering, one subtracts the group mean from each individual score. The new variable reflects how much an individual differs from his or her group average. It is important to keep in mind that group-mean centering represents a completely different parameterization of the model than does the raw or grand-mean centered version (Hofmann & Gavin, 1998; Hox, 2002; Snijders & Bosker, 1999). Most authors recommend that one use group-mean centering only if there is a

strong theoretical reason to believe that a respondent's relative position within the group is more important than the absolute rating (Hox, 2002; Snijders & Bosker, 1999). For instance, one might use group-mean centering if one believed that the key predictive aspect of work hours was whether an individual worked more or less than his or her group members.

There may also be value in using group-mean centering when testing a cross-level interaction. Hofmann and Gavin (1998) contend that group-mean provides the "purest" estimate of the within-group slope in these situations. That is, slope estimates based on raw variables and grand-mean centered variables can be partially influenced by between-group factors. In contrast, group-mean centered variables have any between-group effects removed. Bryk and Raudenbush (1992) show that group-level interactions can some times pose as cross-level interactions, so a logical strategy is to use raw or grand-mean centered variables to test for cross-level interactions, but verify the final results with group-mean centered variables.

Group-mean centered variables are created by subtracting the group-mean from the raw variable. Thus they are identical to the within-group scores calculated in WABA (see section 3.5.1). To create group-mean centered variables in R, one needs two columns in the dataframe – the raw variable and the group-mean. In section 3.2 the `aggregate` and `merge` functions were illustrated as ways of creating a group-mean variable (via `aggregate`) and merging the group means back with the raw data (via `merge`). Below these functions are used to help create a group-centered leadership variable.

```
> TDAT<-bh1996[,c("GRP","LEAD")]
> TEMP<-aggregate(TDAT$LEAD,list(TDAT$GRP),mean)
> names(TEMP)<-c("GRP","G.LEAD")
> TDAT<-merge(TDAT,TEMP,by="GRP")
> names(TDAT)
[1] "GRP"    "LEAD"   "G.LEAD"
> TDAT$GRP.CENT.LEAD<-TDAT$LEAD-TDAT$G.LEAD
> names(TDAT)
[1] "GRP"           "LEAD"          "G.LEAD"        "GRP.CENT.LEAD"
```

One would typically choose a shorter name for the group-mean centered variables, but this name was chosen to be explicit.

The `bh1996` dataframe has group-mean centered variables for all the predictors. The group-mean centered variables begin with a "W" for "within-group". For comparison, the model below uses the group-mean centered leadership variable in lieu of the raw leadership variable used in the final model in the preceding section.

```
> Final.Model<-lme(WBEING~HRS+LEAD+G.HRS+LEAD:G.HRS,
+ random=~LEAD|GRP,data=bh1996)
> Final.Model.R<-lme(WBEING~HRS+W.LEAD+G.HRS+W.LEAD:G.HRS,
+ random=~LEAD|GRP,data=bh1996)
> round(summary(Final.Model.R)$tTable,dig=3)
             Value Std.Error   DF t-value p-value
(Intercept)  4.705     0.211 7280  22.250   0.000
HRS         -0.028     0.004 7280  -6.264   0.000
W.LEAD       0.044     0.222 7280   0.197   0.844
```

```
G.HRS         -0.142     0.019   97  -7.421   0.000
W.LEAD:G.HRS   0.040     0.019 7280   2.064   0.039
```

Notice that the cross-level interaction is now significant with a t-value of 2.064. In contrast, recall that the cross-level interaction in the model with the non-centered leadership variable had a t-value of 1.703 ($p<.10$). Thus, there are some minor differences between the two model specifications.

4 Growth Modeling

Growth models are an extremely useful variation of multilevel models (see section 3.6). In growth modeling, however, repeated observations from an individual represent the level-1 variables, and the attributes of the individual represent the level-2 variables. The fact that the level-1 variables are repeated over time poses some additional analytic considerations; however, the steps used to analyze the basic growth model and the steps used to analyze a multilevel model share many key similarities.

In this chapter, I begin by briefly reviewing some of the methodological challenges associated with growth modeling. Following this, I illustrate how data must be configured in order to conduct growth modeling. Finally, I illustrate a complete growth modeling analysis using the `nlme` package. Much of this material is taken from Bliese and Ployhart (2002).

4.1 Methodological challenges

In general, the methodological challenges associated with longitudinal analyses of any kind can be daunting. For instance, since longitudinal data is collected from single entities (usually persons) over multiple times, it is likely that there will be some degree of non-independence in the responses. Multiple responses from an individual will tend to be related by virtue of being provided by the same person, and this non-independence violates the statistical assumption of independence underlying many common data analytic techniques (Kenny & Judd, 1986). The issue of non-independence should be familiar to individuals who have worked with multilevel modeling since non-independence due to group membership is key characteristic of multilevel models. That is, multilevel models are fundamentally about modeling the non-independence that occurs when individual responses are affected by group membership.

In longitudinal designs, however, there are additional complications associated with the level-1 responses. First, it is likely that responses temporally close to each other (e.g., responses 1 and 2) will be more strongly related than responses temporally far apart (e.g., responses 1 and 4). This pattern is defined as a simplex pattern or lag 1 autocorrelation. Second, it is likely that responses will tend to become either more variable over time or less variable over time. For instance, individuals starting jobs may initially have a high degree of variability in performance, but over time the variance in job performance may diminish. In statistical terms, outcome variables in longitudinal data are likely to display heteroscedasticity. To obtain correct standard errors and to draw the correct statistical inferences, autocorrelation and heteroscedasticity both need to be incorporated into the statistical model.

The need to test for both autocorrelation and heteroscedasticity in growth models arises because the level-1 variables (repeated measures from an individual) are ordered by time. One of the main difference between growth models and multilevel models revolves around

understanding how to properly account for time in both the statistical models and in the data structures.

In R, growth modeling is conducted using the `nlme` package (Pinhiero & Bates, 2000) and the `lme` function in particular. These are, of course, the same functions used in multilevel modeling (see section 3.6). It will become apparent, however, that the `nlme` package has a wide variety of options available for handling autocorrelation and heteroscedasticity in growth models.

Prior to conducting a growth modeling analysis, one has to create a data set that explicitly includes time as a variable. This data transformation is referred to as changing a data set from multivariate to univariate form or "stacking" data set. In the next section, we show how to create a dataframe for growth modeling.

4.2 Data Structure and the `make.univ` Function

The first step in conducting a growth modeling analysis is to create a data set that is amenable to analysis. Often data is stored in a format where each row represents observations from one individual. For instance, an individual might provide three measures of job satisfaction in a longitudinal study, and the data might be arranged such that column 1 is the subject number; column 2 is job satisfaction at time 1; column 3 is job satisfaction at time 2, and column 4 is job satisfaction at time 3, etc.

The `univbct` dataframe in the multilevel library allows us to illustrate this arrangement. This data set contains three measures taken six-months apart on three variables – job satisfaction, commitment, and readiness. It also contains some stable individual characteristics such as respondent gender, marital status and age at the initial data collection time. These latter variables are treated as level-2 predictors in subsequent modeling.

For convenience, the `univbct` dataframe has already been converted into univariate or stacked form. Thus, it is ready to be analyzed in a growth model; however, for the purposes of illustration, we will select a subset of the entire `univbct` dataframe and transform it back into multivariate form. With this subset we will illustrate how to convert a typical multivariate dataframe into the format necessary for growth modeling.

```
> library(multilevel)
> data(univbct)
> names(univbct)
 [1] "BTN"     "COMPANY" "MARITAL" "GENDER"  "HOWLONG" "RANK"    "EDUCATE"
 [8] "AGE"     "JOBSAT1" "COMMIT1" "READY1"  "JOBSAT2" "COMMIT2" "READY2"
[15] "JOBSAT3" "COMMIT3" "READY3"  "TIME"    "JSAT"    "COMMIT"  "READY"
[22] "SUBNUM"
> nrow(univbct)
[1] 1485
> length(unique(univbct$SUBNUM))
[1] 495
```

These commands tell us that there are 1495 rows in the data set, and that there are data from 495 individuals. Thus each individual provides three rows of data. To create a multivariate data set out of the `univbct` dataframe, we can select every third row of the `univbct` dataframe. In

this illustration we restrict our analyses to the three job satisfaction scores and to respondent age at the initial data collection period.

```
> GROWDAT<-univbct[3*(1:495),c(22,8,9,12,15)]   #selects every third row
> GROWDAT[1:3,]
  SUBNUM AGE  JOBSAT1 JOBSAT2 JOBSAT3
3      1  20 1.666667       1       3
6      2  24 3.666667       4       4
9      3  24 4.000000       4       4
```

The dataframe GROWDAT now contains data from 495 individuals. The first individual was 20 years old at the first data collection time. At time 1, the first individual's job satisfaction score was 1.67; at time 2 it was 1.0, and at time 3 it was 3.0.

Because of the nature of the univbct dataframe in the multilevel package, we have added additional steps by converting a univariate dataframe to a multivariate dataframe; nonetheless, from a practical standpoint the important issue is that the GROWDAT dataframe represents a typical multivariate data set containing repeated measures. Specifically, the GROWDAT dataframe contains one row of information for each subject, and the repeated measures (job satisfaction) are represented by three different variables.

From a growth modeling perspective, the key problem with multivariate dataframes like GROWDAT is that they do not contain a variable that indexes time. That is, we know time is an attribute of this data because we have three different measures of job satisfaction; however, analytically we have no way of explicitly modeling time. Thus, it is critical to create a new variable that explicitly indexes time. Thus requires transforming the data to univariate or a stacked format.

The make.univ function from the multilevel package provides a simple way to perform this transformation. Two arguments are required (x and dvs), and two are optional (tname and outname). The first required argument is the dataframe in multivariate or wide format. The second required argument is a subset of the entire dataframe identifying the columns containing the repeated measures. The second required argument must be time-sorted -- column 1 must be time 1, column 2 must be time 2, and so on. The two optional arguments control the names of the two created variables: tname defaults to "TIME" and outname defaults to "MULTDV".

For instance, to convert GROWDAT into univariate form we issue the following command:

```
> UNIV.GROW<-make.univ(GROWDAT,GROWDAT[,3:5])
> UNIV.GROW[1:9,]
    SUBNUM AGE  JOBSAT1 JOBSAT2 JOBSAT3 TIME   MULTDV
X3       1  20 1.666667       1       3    0 1.666667
X31      1  20 1.666667       1       3    1 1.000000
X32      1  20 1.666667       1       3    2 3.000000
X6       2  24 3.666667       4       4    0 3.666667
X63      2  24 3.666667       4       4    1 4.000000
X64      2  24 3.666667       4       4    2 4.000000
X9       3  24 4.000000       4       4    0 4.000000
```

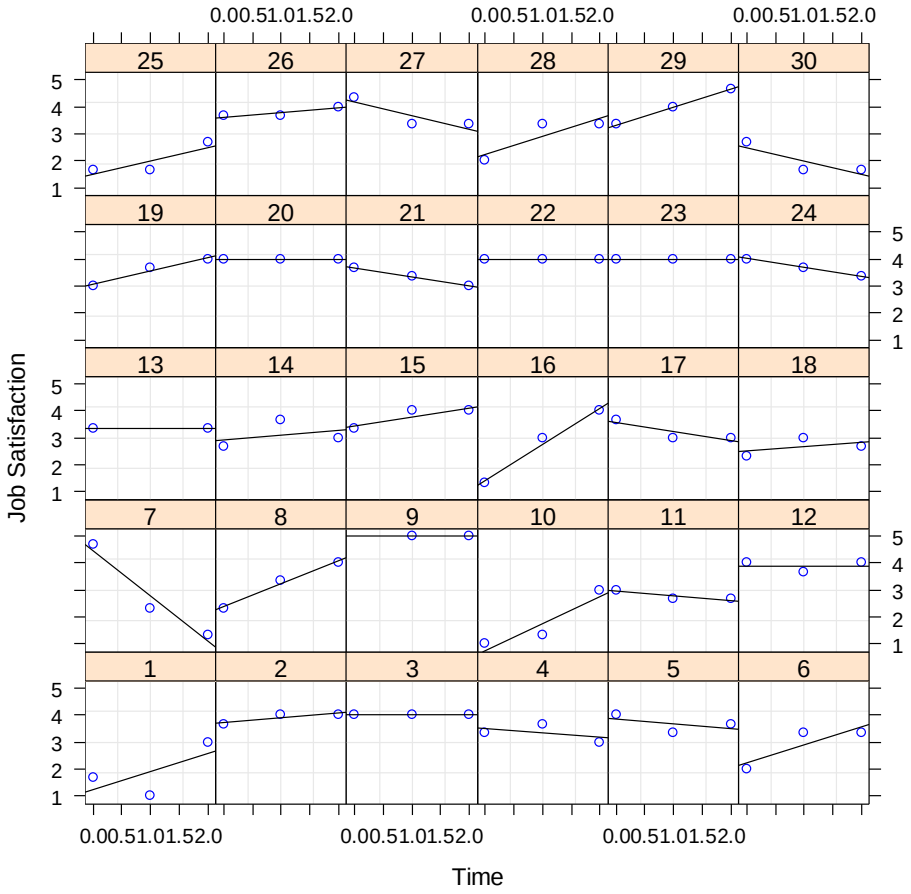
```
X95      3  24 4.000000       4       4    1 4.000000
X96      3  24 4.000000       4       4    2 4.000000
```

Note that each individual now has three rows of data indexed by the variable "TIME". Time ranges from 0 to 2. To facilitate model interpretation, the first time is coded as 0 instead of as 1. Doing so allows one to interpret the intercept as the level of job satisfaction at the initial data collection time. Second, notice that the `make.univ` function has created a variable called "MULTDV". This variable represents the multivariate dependent variable. The variable "TIME" and the variable "MULTDV" are two of the primary variables used in growth modeling. Finally, notice that AGE, SUBNUM and the values for the three job satisfaction variables were repeated three times for each individual. By repeating the individual variables, the `make.univ` function has essentially created a dataframe with level-2 variables in the proper format. For instance, subject age can now be used as a level-2 predictor in subsequent modeling.

4.3 Growth Modeling Illustration

With the data in univariate form, we can begin to visually examine whether or not we see patterns between time and the outcome. For instance, the commands below use the `lattice` package to produce a plot of the first 30 individuals:

```
>trellis.device(device="windows",theme="col.whitebg")
> xyplot(MULTDV~TIME|as.factor(SUBNUM),data=UNIV.GROW[1:90,],
 type=c("p","r","g"),col="blue",col.line="black",
 xlab="Time",ylab="Job Satisfaction")
```



From this plot, it appears as though there is considerable variability both in overall levels of job satisfaction and in how job satisfaction changes over time. The goal in growth modeling is to determine whether or not we can find consistent patterns in the relationship between time and job satisfaction. Thus, we are now ready to illustrate growth modeling in a step-by-step approach. In this illustration, we follow the model comparison approach outlined by Bliese and Ployhart (2002) and in also discussed in Ployhart, Holtz and Bliese (2002).

As an overview of the steps, the basic procedure is to start by examining the nature of the outcome. Much as we did in multilevel modeling, we are interested in estimating the ICC and determining whether the outcome (job satisfaction) randomly varies among individuals. Second, we are interested in examining the form of the relationship between time and the outcome. In essence, we want to know whether the outcome generally increases, decreases, or shows some other type of relationship with time. The plot of the first 30 individuals shows no clear pattern in how job satisfaction is changing over time, but the analysis might identify an overall trend among the 495 respondents. Third, we attempt to determine whether the relationship between time and the outcome is constant among individuals or whether it varies on an individual-by-individual basis. Fourth, we model in more complicated error structures such as autocorrelation, and finally we add level-2 predictors of intercept and slope variances.

4.3.1 Step 1: Examine the DV

The first step in growth modeling is to examine the properties of the dependent variable. As in multilevel modeling, one begins by estimating a null model and calculating the ICC.

```
> null.model<-lme(MULTDV~1,random=~1|SUBNUM,data=UNIV.GROW,
na.action=na.omit)
> VarCorr(null.model)
SUBNUM = pdLogChol(1)
            Variance  StdDev
(Intercept) 0.4337729 0.6586144
Residual    0.4319055 0.6571952
> 0.4337729/(0.4337729+0.4319055)
[1] 0.5010786
```

In our example using the UNIV.GROW dataframe, the ICC associated with job satisfaction is .50. This indicates that 50% of the variance in any individual report of job satisfaction can be explained by the properties of the individual who provided the rating. Another way to think about this is to note that individuals tend to remain fairly constant in ratings over time, and that there are differences among individuals. This observation is reflected in the graph of the first 30 respondents.

4.3.2 Step 2: Model Time

Step two involves modeling the fixed relationship between time and the dependent variable. In almost all cases, one will begin by modeling a linear relationship and progressively add more complicated relationships such as quadratic, cubic, etc. To test whether there is a linear relationship between time and job satisfaction, we regress job satisfaction on time in a model with a random intercept.

```
> model.2<-lme(MULTDV~TIME,random=~1|SUBNUM,data=UNIV.GROW,
na.action=na.omit)
> summary(model.2)$tTable
                Value  Std.Error  DF   t-value    p-value
(Intercept) 3.21886617 0.04075699 903 78.977040 0.00000000
TIME        0.05176461 0.02168024 903  2.387640 0.01716169
```

An examination of the fixed effects indicates that there is a significant linear relationship between time and job satisfaction such that job satisfaction increases by .05 each time period. Note that since the first time period was coded as 0, the intercept value in this model represents the average level of job satisfaction at the first time period. Specifically, at the first time period the average job satisfaction was 3.22.

More complicated time functions can be included in one of two ways – either through raising the time variable to various powers, or by converting time into power polynomials. Below both techniques are illustrated.

```
> model.2b<-lme(MULTDV~TIME+I(TIME^2),random=~1|SUBNUM,
data=UNIV.GROW,na.action=na.omit)
> summary(model.2b)$tTable
```

```
                  Value  Std.Error  DF    t-value   p-value
(Intercept)  3.23308157 0.04262697 902 75.8459120 0.0000000
TIME        -0.03373846 0.07816572 902 -0.4316273 0.6661154
I(TIME^2)    0.04276425 0.03756137 902  1.1385167 0.2552071

> model.2c<-lme(MULTDV~poly(TIME,2),random=~1|SUBNUM,
data=UNIV.GROW,na.action=na.omit)
> summary(model.2c)$tTable
                   Value Std.Error  DF   t-value    p-value
(Intercept)    3.2704416 0.0346156 902 94.478836 0.00000000
poly(TIME, 2)1 1.5778835 0.6613714 902  2.385775 0.01724863
poly(TIME, 2)2 0.7530736 0.6614515 902  1.138517 0.25520707
```

Both models clearly show that there is no significant quadratic trend. Note that a key advantage of the power polynomials is that the linear and quadratic effects are orthogonal. Thus, in the second model the linear effect of time is still significant even with the quadratic effect in the model. In either case, however, we conclude that time only has a linear relationship with job satisfaction.

4.3.3 Step 3: Model Slope Variability

A potential limitation with model 2 is that it assumes that the relationship between time and job satisfaction is constant for all individuals. Specifically, it assumes that each individual increases job satisfaction by .05 points at each time period. An alternative model that needs to be tested is one that allows slopes to randomly vary. Given the degree of variability in the graph of the first 30 respondents, a random slope model seems quite plausible with the current data. The random slope model is tested by adding the linear effect for time as a random effect. In the running example, we can simply `update` model.2 by adding a different random effect component and contrast model 2 and model 3.

```
> model.3<-update(model.2,random=~TIME|SUBNUM)
> anova(model.2,model.3)
        Model df      AIC      BIC    logLik   Test  L.Ratio p-value
model.2     1  4 3461.234 3482.194 -1726.617
model.3     2  6 3434.132 3465.571 -1711.066 1 vs 2 31.10262  <.0001
```

The results clearly indicate that a model that allows the slope between time and job satisfaction to randomly vary fits the data better than a model that fixes the slope to a constant value for all individuals.

In cases where higher-level trends were also significant, one would also be interested in determining whether allowing the slopes of the higher-level variables to randomly vary also improved model fit. For instance, one might find that a quadratic relationship varied in strength among individuals.

4.3.4 Step 4: Modeling Error Structures

The fourth step in developing the level-1 model involves assessing the error structure of the model. It is important to carefully scrutinize the level-1 error structure because significance tests may be dramatically affected if error structures are not properly specified. The goal of step 4 is

to determine whether one's model fit improves by incorporating (a) an autoregressive structure with serial correlations and (b) heterogeneity in the error structures.

Tests for autoregressive structure (autocorrelation) are conducted by including the `correlation` option in `lme`. For instance, we can update model.3 and include lag 1 autocorrelation as follows:

```
> model.4a<-update(model.3,correlation=corAR1())
> anova(model.3,model.4a)
         Model df      AIC      BIC    logLik   Test  L.Ratio p-value
model.3      1  6 3434.132 3465.571 -1711.066
model.4a     2  7 3429.771 3466.451 -1707.886 1 vs 2 6.360465  0.0117
```

A model that allows for autocorrelation fits the data better than does a model that assumes no autocorrelation. A summary of model 4a reveals that the autocorrelation estimate is .367 (see the Phi coefficient).

```
> summary(model.4a)
Linear mixed-effects model fit by REML
 Data: UNIV.GROW
       AIC      BIC    logLik
  3429.771 3466.451 -1707.886
.....
Correlation Structure: AR(1)
 Formula: ~1 | SUBNUM
 Parameter estimate(s):
      Phi
0.3676831
```

Finally, we can examine the degree to which the variance of the responses changes over time. A simple preliminary test of variance homogeneity can be conducted by examining the variance of job satisfaction at each time point using the `tapply` command.

```
> tapply(UNIV.GROW$MULTDV,UNIV.GROW$TIME,var,na.rm=T)
        0         1         2
0.9681912 0.8831397 0.7313358
```

The analysis suggests the variance of job satisfaction is decreasing over time. To model decreasing variance one can use the `varExp` option. In cases where variance increases can use the `varFixed` option (see Pinheiro & Bates, 2000 for details).

```
> model.4b<-update(model.4a,weights=varExp(form=~TIME))
> anova(model.4a,model.4b)
         Model df      AIC      BIC    logLik   Test  L.Ratio p-value
model.4a     1  7 3429.771 3466.451 -1707.886
model.4b     2  8 3428.390 3470.309 -1706.195 1 vs 2 3.381686  0.0659
```

The model that includes both autocorrelation and allows for decreases in variance fits the data marginally better (using a liberal p-value) than does the model that only includes autocorrelation. In subsequent analyses, however, `model.4b` ran into convergence problems. Consequently, we elect to use `model.4a` as our final level-1 model.

With the completion of step 4, we have exhaustively examined the form of the level-1 relationship between time and job satisfaction. This analysis has revealed that (a) individuals randomly vary in terms of their mean levels of job satisfaction, (b) there is a linear, but not quadratic, relationship between time and job satisfaction, (c) the strength of the linear relationships randomly varies among individuals, and (d) there is a fair amount of autocorrelation in the data. At this point, we are ready to add level-2 variables to try and explain the random variation in intercepts (i.e., mean job satisfaction) and in the time-job satisfaction slope.

4.3.5 Step 5: Predicting Intercept Variation

Step 5 in growth modeling is to examine factors that can potentially explain intercept variation. Specifically, in our case we are interested in examining factors that explain why some individuals have high job satisfaction while other individuals have low job satisfaction. In this example, we explore the idea that age is related to intercept variation.

To model this relationship, the individual-level characteristic, age, is used as a predictor of the job satisfaction intercept. The model that we will test is represented below using the Bryk and Raudenbush (1992) notation.

$$Y_{ij} = \pi_{0j} + \pi_{1j}(Time_{ij}) + r_{ij}$$

$$\pi_{0j} = \beta_{00} + \beta_{01}(Age_j) + u_{0j}$$

$$\pi_{1j} = \beta_{10} + u_{1j}$$

This equation states that respondent j's initial job satisfaction (π_{0j}) can be modeled as a function of two things. One is the mean level of job satisfaction (β_{00}) for all respondents. The second is a coefficient associated with the individual's age (β_{01}). Note that the error term for the intercept (u_{0j}) now represents the difference between an individuals' intercept and the overall intercept after accounting for age. In `lme` the model is specified as:

```
> model.5<-lme(MULTDV~TIME+AGE,random=~TIME|SUBNUM,
  correlation=corAR1(),na.action=na.omit,data=UNIV.GROW)

> round(summary(model.5)$tTable,dig=3)
            Value Std.Error  DF t-value p-value
(Intercept) 2.347     0.146 897  16.086   0.000
TIME        0.053     0.024 897   2.205   0.028
AGE         0.034     0.005 486   6.241   0.000
```

Model 5 differs only from Model 4a in that Model 5 includes a new fixed effect, AGE. Notice that age is positively related to initial levels of job satisfaction. Also notice that there are fewer degrees of freedom for age than for time since age is an individual (level-2) attribute.

In interpreting the coefficients from model 5, we conclude that in cases where age is 0 and where time is 0, the expected level of job satisfaction is 2.347. In some ways, this interpretation is strange because age will never actually be 0 in this population. Consequently, it may be useful to reparameterize age by grand-mean centering the variable (see Singer, 1998). Grand mean

centering involves subtracting the overall mean from each observation (see section 3.6.2). A model using a grand-mean centered version of age (AGE2) is presented below.

```
> UNIV.GROW$AGE2<-UNIV.GROW$AGE-mean(UNIV.GROW$AGE,na.rm=T)
> model.5b<-lme(MULTDV~TIME+AGE2,random=~TIME|SUBNUM,
  correlation=corAR1(),na.action=na.omit,data=UNIV.GROW)
> round(summary(model.5b)$tTable,dig=3)
            Value Std.Error  DF t-value p-value
(Intercept) 3.216     0.043 897  74.564   0.000
TIME        0.053     0.024 897   2.205   0.028
AGE2        0.034     0.005 486   6.241   0.000
```

With age grand-mean centered, the intercept estimate of 3.216 now represents the initial job satisfaction value for a respondent of average age (25.7 years old). Notice that the t-values for time and age did not change between this and the previous model. While we will continue our analyses using the untransformed age variable, readers should keep in mind that grand-mean centering is often valuable in terms of both enhancing the interpretability of models.

4.3.6 Step 6: Predicting Slope Variation

The final aspect of growth modeling involves attempting to determine attributes of individual respondents that are related to slope variability. In this section, we attempt to determine whether respondent age can explain some of the variation in the time-job satisfaction slope. The model that we test is presented below:

$$Y_{ij} = \pi_{0j} + \pi_{1j}(Time_{ij}) + r_{ij}$$

$$\pi_{0j} = \beta_{00} + \beta_{01}(Age_j) + u_{0j}$$

$$\pi_{1j} = \beta_{10} + \beta_{11}(Age_j) + u_{1j}$$

This model is similar to the model specified in step 5 except that we now test the assumption that the slope between time and job satisfaction for an individual (π_{1j}) is a function of an overall slope (β_{10}), individual age (β_{11}), and an error term (u_{1j}). In `lme`, the model is specified as:

```
> model.6<-lme(MULTDV~TIME*AGE,random=~TIME|SUBNUM,
correlation=corAR1(),na.action=na.omit,data=UNIV.GROW)
```

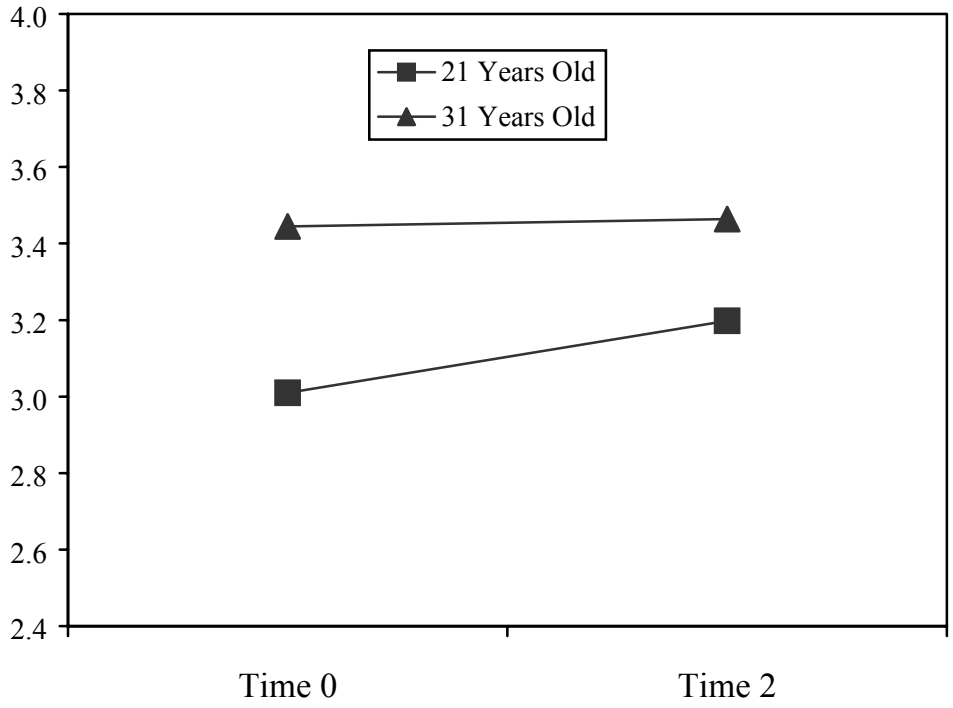
Note that the only difference between model 5 and model 6 is that we include an interaction term for time and age. A summary of model 6 reveals that there is a significant interaction between time and age.

```
> round(summary(model.6)$tTable,dig=3)
             Value Std.Error  DF t-value p-value
(Intercept)  2.098     0.186 896  11.264   0.000
TIME         0.271     0.104 896   2.608   0.009
AGE          0.043     0.007 486   6.180   0.000
TIME:AGE    -0.008     0.004 896  -2.153   0.032
```

In section 3.6.1 we illustrated how to use the predict command to generate points that could be used to plot out interactions. An alternative approach is to use the overall coefficients from the final model in conjunction with high and low values for the predictors to generate points for plots. Notice in the example that follows that the first row in the TDAT dataframe is a row of 1s for the intercept, while the other rows contain high and low values for time, age and the time*age interaction.

```
> TDAT<-data.frame(COEFS=(summary(model.6)$tTable)[,1],
CASE1=c(1,0,21,0),CASE1=c(1,0,31,0),
CASE3=c(1,2,21,42),CASE4=c(1,2,31,62))
> TDAT
                   COEFS CASE1 CASE1 CASE3 CASE4
(Intercept)  2.097720117     1     1     1     1
TIME         0.271036716     0     0     2     2
AGE          0.043449071    21    31    21    31
TIME:AGE    -0.008432157     0     0    42    62
> sum(TDAT[,1]*TDAT[,2])
[1] 3.010151
> sum(TDAT[,1]*TDAT[,3])
[1] 3.444641
> sum(TDAT[,1]*TDAT[,4])
[1] 3.198073
> sum(TDAT[,1]*TDAT[,5])
[1] 3.463921
```

These points are used in the plot of the interaction. Notice that older individuals reported higher job satisfaction initially, and tended to show a very slight increase over time. In contrast, younger respondents tended to report lower initial job satisfaction, but showed a more pronounced increase in job satisfaction over time.



5 Miscellaneous Functions

The `multilevel` package has a number of other functions that have either been referenced in appendices of published papers, or are of basic utility to applied organizational researchers. This section briefly describes these functions. Complete help files are available in the `multilevel` package for each of the functions discussed.

5.1 Scale reliability: `cronbach` and `item.total`

Two functions that are can be particularly useful in estimating the reliability of multi-item scales are the `cronbach` and the `item.total` functions. Both functions take a single argument, a dataframe with multiple columns where each column represents one item in a multi-item scale.

5.2 Random Group Resampling for OLS Regression Models

The function `rgr.OLS` allows one to contrast a group-level hierarchical regression model with an identically specified model where group identifiers are randomly generated. This type of model was estimated in Bliese and Halverson (2002).

5.3 Estimate multiple ICC values: `mult.icc`

The `mult.icc` function can be used to estimate multiple ICC(1) and ICC(2) values in a given data set. For instance, to estimate the ICC(1) and ICC(2) values for work hours, leadership, cohesion and well-being in the `bh1996` data set one provides a dataframe with the

variables of interest as the first argument in the `mult.icc` function, and a grouping variable as the second argument. The `mult.icc` function uses the `nlme` package, so it is important to have this package loaded.

```
> library(nlme)
> mult.icc(bh1996[,c("HRS","LEAD","COHES","WBEING")],bh1996$GRP)
  Variable       ICC1      ICC2
1      HRS 0.12923696 0.9171286
2     LEAD 0.14746131 0.9280442
3    COHES 0.04804840 0.7900745
4   WBEING 0.04337922 0.7717561
```

5.4 Estimating bias in nested regression models: `simbias`

Bliese and Hanges (2004) showed that a failure to model the nested properties of data in ordinary least squares regression could lead to a loss of power in terms of detecting effects. The article provided the `simbias` function to help estimate the degree of power loss in complex situations.

5.5 Detecting mediation effects: `sobel` and `sobel.lme`

MacKinnon, Lockwood, Hoffman, West and Sheets (2002) showed that many of the mediation tests used in psychology tend to have low power. One test that had reasonable power was Sobel's (1982) indirect test for mediation. The `sobel` function provides a simple way to run Sobel's (1982) test for mediation. A second function, `sobel.lme`, is a variant that allows one to include a single level of nesting by adding a group identifier. In `sobel.lme`, the three models used in the mediation test are estimated using a two-level linear mixed effects (`lme`) model. Using the `lme` model in the case of nested data helps provide accurate standard error estimates (Bliese & Hanges, 2004). Details on the use of the `sobel` and the `sobel.lme` functions are available in the help files.

6 Conclusion

This document has provided an overview of how R can be used in a wide variety of multilevel models. It should be apparent that R is a very powerful language that is well-suited to multilevel analyses. Clearly, in learning to use any new program, there is some degree of effort. I am convinced, however, that the benefits associated with learning R will be well worth the effort for scientists whose work revolves around making senses of data. Hopefully, the numerous examples in this document will go a long way towards helping researchers use R for their own multilevel problems, and for any number of other statistical procedures.

7 References

Bartko, J. J. (1976). On various intraclass correlation reliability coefficients. *Psychological Bulletin, 83*, 762-765.

Becker, R. A., Chambers, J. M., & Wilks, A. R. (1988). *The New S Language*. New York: Chapman & Hall.

Bliese, P. D. (1998). Group size, ICC values, and group-level correlations: A simulation. *Organizational Research Methods, 1*, 355-373.

Bliese, P. D. (2000). Within-group agreement, non-independence, and reliability: Implications for data aggregation and Analysis. In K. J. Klein & S. W. Kozlowski (Eds.), *Multilevel Theory, Research, and Methods in Organizations* (pp. 349-381). San Francisco, CA: Jossey-Bass, Inc.

Bliese, P. D. (2002). Multilevel random coefficient modeling in organizational research: Examples using SAS and S-PLUS. In F. Drasgow & N. Schmitt (Eds.), *Modeling in Organizational Research: Measuring and Analyzing Behavior in Organizations* (pp. 401-445). San Francisco, CA: Jossey-Bass, Inc.

Bliese, P. D., & Britt, T. W. (2001). Social support, group consensus and stressor-strain relationships: Social context matters. *Journal of Organizational Behavior, 22*, 425-436.

Bliese, P. D. & Hanges, P. J. (2004). Being both too liberal and too conservative: The perils of treating grouped data as though they were independent. *Organizational Research Methods, 7*, 400-417.

Bliese, P. D. & Halverson, R. R. (1996). Individual and nomothetic models of job stress: An examination of work hours, cohesion, and well-being. *Journal of Applied Social Psychology, 26*, 1171-1189.

Bliese, P. D., & Halverson, R. R. (1998a). Group consensus and psychological well-being: A large field study. *Journal of Applied Social Psychology, 28*, 563-580.

Bliese, P. D., & Halverson, R. R. (1998b). Group size and measures of group-level properties: An examination of eta-squared and ICC values. *Journal of Management, 24*, 157-172.

Bliese, P. D., & Halverson, R. R. (2002). Using random group resampling in multilevel research. *Leadership Quarterly, 13*, 53-68.

Bliese, P. D., & Halverson, R.R. & Rothberg, J. (2000). Using random group resampling (RGR) to estimate within-group agreement with examples using the statistical language R. *Unpublished Manuscript*.

Bliese, P. D. & Jex, S. M. (2002). Incorporating a multilevel perspective into occupational stress research: Theoretical, methodological, and practical implications. *Journal of Occupational Health Psychology, 7*, 265-276.

Bliese, P. D., & Jex S. M. (1999). Incorporating multiple levels of analysis into occupational stress research. *Work and Stress, 13*, 1-6.

Bliese, P. D., & Ployhart, R. E. (2002). Growth modeling using random coefficient models: Model building, testing and illustrations. *Organizational Research Methods, 5*, 362-387.

Bryk, A. S., & Raudenbush, S. W. (1992). *Hierarchical linear models*. Newbury Park, CA: Sage.

Burke, M. J., Finkelstein, L. M., & Dusig, M. S. (1999). On average deviation indices for estimating interrater agreement. *Organizational Research Methods, 2*, 49-68.

Chambers, J. M. & Hastie, T. J. (1992). *Statistical Models in S*. New York: Chapman & Hall.

Cohen, J. & Cohen, P. (1983). *Applied multiple regression/correlation analysis for the behavioral sciences* (2nd Ed.). Hillsdale, NJ: Lawrence Erlbaum Associates.

Cohen, A., Doveh, E. & Eick, U. (2001). Statistical properties of the rwg(j) index of agreement. *Psychological Methods, 6*, 297-310.

Cohen, A., Doveh, E. & Nahum-Shani, I. (in press). Testing agreement for multi-item scales with the indices rwg(j) and ADM(J). *Organizational Research Methods*.

Dansereau, F., Alutto, J. A., & Yammarino, F. J. (1984). *Theory testing in organizational behavior: The varient approach*. Englewood Cliffs, NJ: Prentice-Hall.

Dunlap, W. P., Burke, M. J., & Smith-Crowe, K. (2003). Accurate tests of statistical significance for rwg and average deviation interrater agreement indices. *Journal of Applied Psychology, 88*, 356-362.

Firebaugh, G. (1978). A rule for inferring individual-level relationships from aggregate data. *American Sociological Review, 43*, 557-572.

Hofmann, D. A. (1997). An overview of the logic and rationale of Hierarchical Linear Models. *Journal of Management, 23*, 723-744.

Hofmann, D. A. & Gavin, M. (1998). Centering decisions in hierarchical linear models: Theoretical and methodological implications for research in organizations. *Journal of Management, 24*, 623-641.

James, L. R. (1982). Aggregation bias in estimates of perceptual agreement. *Journal of Applied Psychology, 67*, 219-229.

James, L.R., Demaree, R.G., & Wolf, G. (1984). Estimating within-group interrater reliability with and without response bias. *Journal of Applied Psychology, 69*, 85-98.

James, L. R. & Williams, L. J. (2000). The cross-level operator in regression, ANCOVA, and contextual analysis. In K. J. Klein & S. W. Kozlowski (Eds.), *Multilevel Theory,*

Research, and Methods in Organizations (pp. 382-424). San Francisco, CA: Jossey-Bass, Inc.

Hox, J. J. (2002). *Multilevel analysis: Techniques and applications*. Mahwah, NJ: Lawrence Erlbaum Associates.

Klein, K. J. & Kozlowski, S. W. J, (2000). *Multilevel theory, research, and methods in organizations*. San Francisco, CA: Jossey-Bass, Inc.

Klein, K. J., Bliese, P.D., Kozlowski, S. W. J, Dansereau, F., Gavin, M. B., Griffin, M. A., Hofmann, D. A., James, L. R., Yammarino, F. J. & Bligh, M. C. (2000). Multilevel analytical techniques: Commonalities, differences, and continuing questions. In K. J. Klein & S. W. Kozlowski (Eds.), *Multilevel Theory, Research, and Methods in Organizations* (pp. 512-553). San Francisco, CA: Jossey-Bass, Inc.

Kozlowski, S. W. J., & Hattrup, K. (1992). A disagreement about within-group agreement: Disentangling issues of consistency versus consensus. *Journal of Applied Psychology, 77*, 161-167.

Kreft, I. & De Leeuw, J. (1998). *Introducing multilevel modeling*. London: Sage Publications.

LeBreton, J. M., James, L. R. & Lindell, M. K. (2005). Recent Issues Regarding rWG, rWG, rWG(J), and rWG(J). *Organizational Research Methods, 8*, 128-138.

Levin, J. R. (1967). Comment: Misinterpreting the significance of "explained variation." *American Psychologist, 22*, 675-676.

Lindell, M. K. & Brandt, C. J. (1997). Measuring interrater agreement for ratings of a single target. *Applied Psychological Measurement, 21*, 271-278.

Lindell, M. K. & Brandt, C. J. (1999). Assessing interrater agreement on the job relevance of a test: A comparison of CVI, T, rWG(J), and r*WG(J) indexes. *Journal of Applied Psychology, 84*, 640-647.

Lindell, M. K. & Brandt, C. J. (2000). Climate quality and climate consensus as mediators of the relationship between organizational antecedents and outcomes. *Journal of Applied Psychology, 85*, 331-348.

Lindell, M. K., Brandt, C. J. & Whitney, D. J. (1999). A revised index of interrater agreement for multi-item ratings of a single target. *Applied Psychological Measurement, 23*, 127-135.

MacKinnon, D. P., Lockwood, C. M., Hoffman, J. M., West, S. G., Sheets, V. (2002). A comparison of methods to test mediation and other intervening variable effects. *Psychological Methods, 7*, 83-104.

Pinheiro, J. C. & Bates, D. M. (2000). *Mixed-effects models in S and S-PLUS*. New York: Springer-Verlag.

Ployhart, R. E., Holtz, B. C. & Bliese, P. D. (2002). Longitudinal data analysis: Applications of random coefficient modeling to leadership research. *Leadership Quarterly, 13*, 455-486.

Robinson, W. S. (1950). Ecological correlations and the behavior of individuals. *American Sociological Review, 15*, 351-357.

Shrout, P. E., & Fleiss, J. L. (1979). Intraclass correlations: Uses in assessing rater reliability. *Psychological Bulletin, 86*, 420-428.

Singer, J. D. (1998). Using SAS PROC MIXED to fit multilevel models, hierarchical models, and individual growth models. *Journal of Educational and Behavioral Statistics, 24*, 323-355.

Snijders, T. A. B. & Bosker, R. J. (1994). Modeled variance in two-level models. *Sociological Methods and Research, 22*, 342-363.

Snijders, T. A. B. & Bosker, R. J. (1999). *Multilevel analysis: An introduction to basic and advanced multilevel modeling*. London: Sage Publications.

Sobel, M. E., (1982). Asymptotic confidence intervals for indirect effects in structural equation models. In S. Leinhardt (Ed.), *Sociological Methodology 1982* (pp. 290-312). Washington, DC: American Sociological Association.

Tinsley, H. E. A., & Weiss, D. J. (1975). Interrater reliability and agreement of subjective judgements. *Journal of Counseling Psychology, 22*, 358-376.