Lab 1

Reshaping Data Files
Psychology 319 (GCM)

Instructions. Work through the lab, saving the output as you go. If you work in Microsoft Word, you can easily copy any graph to Word via the clipboard. Numerical output may also be copied easily by highlighting, moving it to the clipboard, then copying into Word. However, you should format R output in TrueType Courier New font so that it is *monospaced*. Output from this lab is to be handed in by Monday, February 1. Your output file should be named `LAST_FIRST_LAB1.DOC`, where `LAST` is your last name, and `FIRST` is your first name. Any additional files should have the same naming scheme, except the file extension should be correct. You may add any description text you wish after `LAB1` in the file name.

Preamble. This exercise *reshaping* a data file to and from *long* and *wide* formats. The long format is what Singer and Willett refer to as *person-period* format. The wide format is what Singer and Willet refer to as *person-level* data format.

1 Introduction

As we mentioned in lecture, there are two distinctly different ways to store longitudinal data. The more familiar way to most beginning students is the *wide* or *person-level* format, in which each person's data has one row in the data file, and all variables measured for that person, including repeated measurements of the same variable, occur in columns. A less familiar format, but either useful or essential for analysis by contemporary statistical software, is the *long* or *person-period* format. In this format, each person has a row of data for each time period in the analysis.

It is sometimes necessary to convert from one of these two formats to the other. In this exercise, we do this, using the `reshape()` function.

2 Getting Started – The Tolerance Data

The `tolerance` data set is discussed in an introductory example in Singer and Willett, Chapter 2. Go to the course website and download the `tolerance1.txt`

and `tolerance1_pp.txt` data files. The `tolerance.txt` file is in wide format. Let's load it in and take a look. If you examine this file, you'll see that the first line is a header with variable names, and that the fields are comma-delimited.

```
> tol.wide <- read.table("tolerance1.txt",header=TRUE,sep=",")
> tol.wide
      id tol11 tol12 tol13 tol14 tol15 male exposure
1      9  2.23  1.79  1.90  2.12  2.66    0     1.54
2     45  1.12  1.45  1.45  1.45  1.99    1     1.16
3    268  1.45  1.34  1.99  1.79  1.34    1     0.90
4    314  1.22  1.22  1.55  1.12  1.12    0     0.81
5    442  1.45  1.99  1.45  1.67  1.90    0     1.13
6    514  1.34  1.67  2.23  2.12  2.44    1     0.90
7    569  1.79  1.90  1.90  1.99  1.99    0     1.99
8    624  1.12  1.12  1.22  1.12  1.22    1     0.98
9    723  1.22  1.34  1.12  1.00  1.12    0     0.81
10   918  1.00  1.00  1.22  1.99  1.22    0     1.21
11   949  1.99  1.55  1.12  1.45  1.55    1     0.93
12   978  1.22  1.34  2.12  3.46  3.32    1     1.59
13  1105  1.34  1.90  1.99  1.90  2.12    1     1.38
14  1542  1.22  1.22  1.99  1.79  2.12    0     1.44
15  1552  1.00  1.12  2.23  1.55  1.55    0     1.04
16  1653  1.11  1.11  1.34  1.55  2.12    0     1.25
```

Note that variables `tol11`–`tol15` represent repeated measures of the `tolerance` variable at ages 11 through 15. However, there are also some *time-invariant covariates* in the file. The values on these variates are only measured once. Each person has only one row of data.

3 Reshaping the Tolerance Data to Long (Person-Period) Format

To accomplish reshaping the file, we use the `reshape` command.[1]

[1] There is also a package called *reshape* available. The documentation for this package is extremely "skeletal," and examples of even the most basic, common operations are missing. Commentary on the web indicates that these kinds of examples have proven troublesome for users of the package, so I cannot recommend it at this time.

The documentation for the `reshape` command is extremely terse (to put it mildly). Examine the example below in conjunction with the documentation until you are sure you understand it!

```
> tol.long <- reshape(tol.wide,varying=list(2:6),
+ v.names="tol",direction="long",times=11:15,
+ timevar="age")
```

Take a look at the tol.long data frame on your computer. You'll note that unlike the example in Singer and Willett, the file is not yet sorted by `id`, so a final step is to sort the file. If you go to *www.statmethods.net*, you'll find hints on how to sort data lurking in the *Data Management* section under the heading *Sorting Data*. We'll simply sort the file "in place." Notice that, since I have deliberately not `attach`ed the `tol.long` file, I need to refer to the `id` variable within the `tol.long` file using the syntax `tol.long$id`.

```
> tol.long <- tol.long[order(tol.long$id),]
> tol.long
```

	id	male	exposure	age	tol
9.11	9	0	1.54	11	2.23
9.12	9	0	1.54	12	1.79
9.13	9	0	1.54	13	1.90
9.14	9	0	1.54	14	2.12
9.15	9	0	1.54	15	2.66
45.11	45	1	1.16	11	1.12
45.12	45	1	1.16	12	1.45
45.13	45	1	1.16	13	1.45
45.14	45	1	1.16	14	1.45
45.15	45	1	1.16	15	1.99
268.11	268	1	0.90	11	1.45
268.12	268	1	0.90	12	1.34
268.13	268	1	0.90	13	1.99
268.14	268	1	0.90	14	1.79
268.15	268	1	0.90	15	1.34
314.11	314	0	0.81	11	1.22
314.12	314	0	0.81	12	1.22
314.13	314	0	0.81	13	1.55
314.14	314	0	0.81	14	1.12
314.15	314	0	0.81	15	1.12
442.11	442	0	1.13	11	1.45

```
442.12   442    0     1.13  12 1.99
442.13   442    0     1.13  13 1.45
442.14   442    0     1.13  14 1.67
442.15   442    0     1.13  15 1.90
514.11   514    1     0.90  11 1.34
514.12   514    1     0.90  12 1.67
514.13   514    1     0.90  13 2.23
514.14   514    1     0.90  14 2.12
514.15   514    1     0.90  15 2.44
569.11   569    0     1.99  11 1.79
569.12   569    0     1.99  12 1.90
569.13   569    0     1.99  13 1.90
569.14   569    0     1.99  14 1.99
569.15   569    0     1.99  15 1.99
624.11   624    1     0.98  11 1.12
624.12   624    1     0.98  12 1.12
624.13   624    1     0.98  13 1.22
624.14   624    1     0.98  14 1.12
624.15   624    1     0.98  15 1.22
723.11   723    0     0.81  11 1.22
723.12   723    0     0.81  12 1.34
723.13   723    0     0.81  13 1.12
723.14   723    0     0.81  14 1.00
723.15   723    0     0.81  15 1.12
918.11   918    0     1.21  11 1.00
918.12   918    0     1.21  12 1.00
918.13   918    0     1.21  13 1.22
918.14   918    0     1.21  14 1.99
918.15   918    0     1.21  15 1.22
949.11   949    1     0.93  11 1.99
949.12   949    1     0.93  12 1.55
949.13   949    1     0.93  13 1.12
949.14   949    1     0.93  14 1.45
949.15   949    1     0.93  15 1.55
978.11   978    1     1.59  11 1.22
978.12   978    1     1.59  12 1.34
978.13   978    1     1.59  13 2.12
978.14   978    1     1.59  14 3.46
978.15   978    1     1.59  15 3.32
1105.11 1105    1     1.38  11 1.34
```

```
1105.12 1105    1     1.38  12 1.90
1105.13 1105    1     1.38  13 1.99
1105.14 1105    1     1.38  14 1.90
1105.15 1105    1     1.38  15 2.12
1542.11 1542    0     1.44  11 1.22
1542.12 1542    0     1.44  12 1.22
1542.13 1542    0     1.44  13 1.99
1542.14 1542    0     1.44  14 1.79
1542.15 1542    0     1.44  15 2.12
1552.11 1552    0     1.04  11 1.00
1552.12 1552    0     1.04  12 1.12
1552.13 1552    0     1.04  13 2.23
1552.14 1552    0     1.04  14 1.55
1552.15 1552    0     1.04  15 1.55
1653.11 1653    0     1.25  11 1.11
1653.12 1653    0     1.25  12 1.11
1653.13 1653    0     1.25  13 1.34
1653.14 1653    0     1.25  14 1.55
1653.15 1653    0     1.25  15 2.12
```

4 A Simple Example

Problem 1. Dr. Sun-Joo Cho had this table in a recent exercise in her advanced IRT course.

```
> person <- 1:5
> item1 <- c(1,0,1,1,0)
> item2 <- c(0,0,0,1,1)
> item3 <- c(1,1,0,1,1)
> table1.wide <- data.frame(person,item1,item2,item3)
> table1.wide
  person item1 item2 item3
1      1     1     0     1
2      2     0     0     1
3      3     1     0     0
4      4     1     1     1
5      5     0     1     1
```

See if you can reshape it with R to person-period (long) format. Note that the ID variable is named `person` in this case, so you will have to be

careful to let R know about. Read the documentation about the `idvar` parameter!

5 Deal with More Than One Time-Varying Measure

Of course, often we will have more than one time-varying measure as well as more than one time-invariant measure. Download the comma-delimited file *twovar.txt* and load it into the data frame *twovar.wide*. Examine the file. You can see that, for each of the 4 individuals, there is an ID, Gender, IQ, and scores on two time-varying variables V and M at grades 10,11,12.

```
  ID Male  IQ V10 V11 V12 M10 M11 M12
1  1    1 108  51  55  59  44  44  52
2  2    0 112  50  59  66  46  51  60
3  3    0  90  33  35  40  39  44  46
4  4    1  99  40  43  46  44  46  51
```

Problem 2. I want you to reshape the *twovar.wide* data frame to look like this:

```
     ID Male  IQ grade  V  M
1.10  1    1 108    10 51 44
1.11  1    1 108    11 55 44
1.12  1    1 108    12 59 52
2.10  2    0 112    10 50 46
2.11  2    0 112    11 59 51
2.12  2    0 112    12 66 60
3.10  3    0  90    10 33 39
3.11  3    0  90    11 35 44
3.12  3    0  90    12 40 46
4.10  4    1  99    10 40 44
4.11  4    1  99    11 43 46
4.12  4    1  99    12 46 51
```

Note that we are trying to create a file where scores for both V and M are given at each time period in person-period (long) format. How do we handle this?

Hint. The extension is actually rather straightforward. Note that the syntax for `reshape` includes the facility to give a *list* of `vectors` of variable

positions. If there is more than one variable, you include more than one vector. Each vector of indices should correspond to one variable.

6 Going from Long to Wide Format

You can reshape a file from long to wide format, although it is less likely that we will have to do that. To see what happens, after getting your `twovar.long` data frame finalized, try the following:

```
> reshape(twovar.long,direction="wide")
     ID Male  IQ V10 M10 V11 M11 V12 M12
1.10  1    1 108  51  44  55  44  59  52
2.10  2    0 112  50  46  59  51  66  60
3.10  3    0  90  33  39  35  44  40  46
4.10  4    1  99  40  44  43  46  46  51
```

7 Saving the File

After going to all the trouble to create the file in long format, you should probably save it! For example

```
> write.table(tol.long,"tol.long.txt",col.names=TRUE,sep=",",
+ row.names=FALSE)
```