How to Convert Wide Format Data to Long Format Data in R

1. This document considers how to convert data in wide format to data in long format. For an RET example, let the number 1 refer to the pretest, 2 refer to a mid-treatment assessment, 3 refer to the immediate posttest, and 4 to a three month follow-up. TREAT is a binary treatment dummy variable, MA is one mediator, MB is a second mediator, and SEX is biological sex, a covariate that is measured at the pretest. ID is a case ID variable. The wide format data are structured as follows (I show only the first three cases)

ID	TREAT	Y1	Y2	Y3	Y4	MA1	MA2	MA3	MA4	MB1	MB2	MB3	MB4	SEX
1	1	10.1	14.2	14.4	14.3	8.1	10.2	10.3	10.1	5.2	7.5	7.9	7.1	0
2	0	10.1	14.2	14.3	14.4	8.5	10.6	10.7	10.8	5.9	7.0	7.1	7.2	1
3	1	10.8	14.9	14.0	14.1	8.2	10.3	10.4	10.5	5.6	7.7	7.8	7.9	0

My task is to convert this data to long format in the following form:

ID	TIME	Y	MA	MB	TREAT	SEX
1	1	10.1	8.1	5.2	1	0
1	2	14.2	10.2	7.5	1	0
1	3	14.4	10.3	7.9	1	0
1	4	14.3	10.1	7.1	1	0
2	1	10.1	8.5	5.9	0	1
2	2	14.2	10.6	7.0	0	1
2	3	14.3	10.7	7.1	0	1
2	4	14.4	10.8	7.2	0	1
3	1	10.8	8.2	5.6	1	0
3	2	14.9	10.3	7.7	1	0
3	3	14.0	10.4	7.8	1	0
3	4	14.1	10.5	7.9	1	0

There are 4 rows per participant, one for each time point. A new variable, TIME, is created to reflect the four time points. The time varying variables (Y, MA, and MB) have unique scores for each time point. The time invariant variables (TREAT, SEX) repeat the same score at each time point for a given participant.

2. I show the R syntax you need to execute to make the conversion. R syntax uses the symbol <- to indicate =. In newer versions of R, you can just use the equal sign and that is what I do here. I do not explain all aspects of the R syntax, just enough for you to make the conversion. A # indicates the rest of the line following the # symbol is a comment and is to be ignored.

3. Read the data into R in its traditional wide format by using the following R commands (I store the data in a data set/frame called XXX:

```
XXX=read.table('c:/ret/temp.dat', header=TRUE)   # get the data; assumes variable names are in Line 1
attach(XXX)                                       # makes the data file salient in R
XXX[XXX==-9999]=NA                                # set missing data, which was scored -9999, to NA
```

4. Use the reshape command in R to make the conversion to a long data set. I use the name longdata here to refer to the converted data set but you can use any name you want:

```
longdata=reshape(data=XXX,direction='long',timevar='TIME',idvar='ID',sep="",
varying = c(' Y1', 'Y2','Y3 ', 'Y4', 'MA1', 'MA2', 'MA3', 'MA4', 'MB1', 'MB2', 'MB3', 'MB4'))
```

Here is a key to the different parameter specifications in the above command:

data: the name of the input wide data file that you want to convert

direction: tells reshape to convert from wide to long format

timevar: a new variable to be created to indicate the time. The label I give to the variable is TIME. In this case, it will have four values 1, 2, 3, or 4. The name needs to be in quotes, regular quotes or single quotes.

idvar: The name of your case ID variable in the wide data set

sep: Each time-varying variable must end with a number and have the same root. All the Y variables have the "root" Y and end in the number 1, 2, 3 or 4. The same is true for MA (root = MA) and MB (root = MB). Some people use a . or an underline to separate the root from the number (e.g., MA.1 or MA_1), in which case I would use sep="." or sep="_". In my case, I use nothing to separate the root from the number, so I indicate a quote with nothing in between, sep = "". Be sure that the root itself does not end with a number. It will confuse R. You may have to rename your variables.

varying: These are the names of all the variables that vary over time. In R, we use the c(   ) command to combine multiple labels, and each label is contained in quotes and they are comma separated.

5. Once the data set called longdata is created using the above command, I sort the data to make more sense of it when I do my final visual inspection of the data (reshape puts it in a not-so-coherent order):

```
longdata=longdata[order(ID, TIME),]
```

The above command orders the rows of the data by ID and then by TIME within ID

6. Sometimes I like to reorder the variables in the data set by moving the case ID and the TIME variables to be the first two variables and placing all the time invariant variables at the end of the data set. Here, I rename the data set along with way, to the name 'maindata'. Here is the command to reorganize the variables

```
maindata=longdata[,c(1,4,5,6,7,2,3)]
```

The numbers after c refer to the column numbers of each variable in the longdata data set (e.g., 1 = the first variable in the data set, 2 = the second variable in the data set, 3 = the third variable, and so on). I put the existing variables in the new order specified after c. In this case, the old variable 1 stays as the first variable. The old variable 4 is moved to become the second variable in the new data set. And so on. You can see the order of the variables in the original longdata set before you do execute the above command if you type

```
colnames(longdata)
```

7. R assigns row names to the data, which with the reshape command are cumbersome. I replace the row names with simple integers as follows:

```
rownames(maindata) = c()
```

This completes the long format data set. I can inspect it by typing in its name and hitting a carriage return:

```
maindata
```

9. Next, I might want to add variables to the data set in the form of product terms or some other transformation. One type of variable I might add is a dummy variable that reverse scores a dummy variable already in the data set, such as the variable SEX. The easiest way to do so is as follows:

```
maindata$RSEX = abs(SEX-1)
```

The name of the data set occurs to the left of the $ sign and the name of the new variable occurs to the right of the $ sign (RSEX). The function abs is an absolute value function. Work through the math and you will see that this statement accomplishes reverse coding.

Here is the syntax for a product term between SEX and MA:

```
maindata$PROD = SEX*MA
```

10. To inspect the final data set, type in its name and hit return

```
maindata
```

11. You then might want to save the data file. Here is the command to save it:

```
write.table(maindata, file='c:/ret/jimdata.dat', row.names=FALSE, col.names=TRUE,quote=FALSE,
fileEncoding='UTF-8')
```

This will save the data in a file called jimdata.dat in a way that is compatible with R and the program I use to read in the data. Use forward slashes instead of backslashes, and enclose the location in quotes. The first entry after the parentheses tells R the name of the data set to save. In the file portion, I can use any file name and extension I want. I used jimdata.dat.