Appendix I: Calculating and Displaying Confidence Intervals and Bands Using Meta-Regression in Stata

1) First make sure that Stata has installed the most recent version of the metareg command by using the "lookup" command:

```
. lookup metareg

Keyword search

        Keywords:  metareg
          Search:  (1) Official help files, FAQs, Examples, SJs, and STBs

Search of official help files, FAQs, Examples, SJs, and STBs

SJ-8-4  sbe23_1 . . . . . . . . . . . . . . . . . . . . Meta-regression in Stata
        (help metareg if installed) . . . . R. M. Harbord and J. P. T. Higgins
        Q4/08   SJ 8(4):493--519
        presents a revised version of the metareg command, which
        performs meta-analysis regression on study-level summary
        data
```

Selecting the sbe23_1 entry (above) lets a user install or update to incorporate Harbord's metareg command, which was first published in 2008 (Harbord and Higgins, 2008), updated from an earlier version that Stata published in 1998. If the Lipsey and Wilson (2001) metareg macro is also installed, it will need to be renamed in order for the alternative to operate. For the graphs we produced for this paper, we used Stata version 11.2.

2) After installing or updating the command, it can be used to estimate meta-analysis regression (viz. meta-regression) models. The basic form of the command follows this format:

```
. metareg T moderator1 moderator2, wsse(se_T)
```

where *T* is the effect size, *moderator1* and *moderator2* are moderator dimensions, and *se_T* is the standard error for each T. By default, this command will invoke a random-effects constant (see discussion in the text about "mixed-" vs. "random-effects" assumptions). If inverse-variance weights, *TW*, have been calculated for each *T* (see, e.g., Lipsey & Wilson, 2001, Table 3.2), but not *se_T* then to use the command, it is necessary to calculate *se_T*:

```
. gen se_T=1/sqrt(TW)
```

For example, in the antidepressants data described in the main text, the command

```
. metareg diffimp meanbase, wsse(se_diffimp) graph
```

where diffimp is the *T* (difference in improvement in depression between drug and placebo groups), meanbase is the mean severity of depression at baseline, and *se_diffimp* is the standard error for each T, produces statistical output for the meta-regression model and a so-called bubble graph, where the bubbles are individual *T*s sized according to their weight in the analysis, and inserts a line indicating the meta-regression line. The command can be used to generate confidence intervals at certain values of the moderator variable, such as the lowest and highest observed values. For example,

```
. gen tempvar=meanbase-17.0631
```

where 17.0631 was the lowest observed mean severity of depression at baseline, and *tempvar* is intended to replace *meanbase* in the re-instantiated command:

```
. metareg diffimp tempvar, wsse(se_diffimp) graph
```

which moves the constant or intercept to the value of 17 (the graph now literally shows the intercept at the zero point of the newly calculated variable, *tempvar*, on the left side of the graph). To obtain an estimate at the highest observed mean value, 29.444:

```
. replace tempvar=meanbase-29.444
```

and we re-use the *tempvar* variable and re-instantiate the metareg command:

```
. metareg diffimp tempvar, wsse(se_diffimp) graph
```

and the zero point now appears on the right side of the graph. (Note that the graph command will not work with more than one covariate or moderator.) These models provide confidence intervals for these point estimates but not confidence bands and other additions; for these, see the next steps.

3) Because the metareg command can be used with post-estimation commands, a number of other possibilities for graphing become possible. Specifically, after running the metareg command, run these commands:

Command	Explanation
`. predict fit`	Saves predicted values from the model for each *T* in a variable called *fit*
`. predict stdp, stdp`	Saves the standard error of the prediction in a variable called *stpd*
`. predict stdf, stdf`	Saves the standard error of the forecast in a variable called *stdf*
`. predict xbu, xbu`	Saves empirical Bayes estimates (predictions including random effects) in a variable called *xbu*
`. local t = invttail(e(df_r)-1, 0.025)`	*t* is used in calculating 95% confidence and prediction bands; for 90%, replace 0.025 with 0.05; for 99% replace it with 0.005
`` . gen confl = fit - `t'*stdp ``	Calculate lower confidence band (using standard error of the prediction), *confl*
`` . gen confu = fit + `t'*stdp ``	Calculate upper confidence band (using standard error of the prediction), *confu*
`` . gen predl = fit - `t'*stdf ``	Calculate lower prediction band (using standard error of the forecast), *predl*
`` . gen predu = fit + `t'*stdf ``	Calculate upper prediction band (using standard error of the forecast), *predu*

4) Now the variables saved in step 3 can be used for plotting. To produce a graph similar to that in Figure 1, panel c:

```
. twoway rarea confl confu meanbase || line fit meanbase ||
```

where *rarea* calls an area plot to depict the confidence bands and *line* plots the predicted values. Note that we created more elaborate versions of these commands to enhance their appearance (e.g., inserting reference lines, tick marks, etc.). In order to produce a bubble graph similar to Figure 3, panel b:

```
. twoway rarea confl confu meanbase || line fit meanbase || scatter diffimp meanbase
[aw=1/se_diffimp^2], msymbol(Oh) ||
```

This command sizes bubbles according to the fixed-effects variance estimate; the random-effects equivalent can be produced by summing the random-effects variance component, τ^2, and the fixed-effects variance:

```
. gen TWr=1/(fevar + τ2)
```

where *fevar* is 1/*TW* and τ^2 is a constant value taken from the statistical output. Now the equivalent command sizes the bubbles according to their total weight:

```
. twoway rarea confl confu meanbase || line fit meanbase || scatter diffimp meanbase
[aw=TWr], msymbol(Oh) ||
```

Note that just like the variables in step 3, *TWr* would need to be recalculated for each such model because τ^2 and the other variables vary from model to model.

5) In order to show the prediction interval surrounding the meta-regression line, such as in Figure 3, panel c:

```
. twoway rarea predl predu meanbase || line fit meanbase ||
```

6) In order to display the empirical Bayesian estimates along with the meta-regression line and confidence bands,

```
. twoway || rarea confl confu meanbase || line fit meanbase || scatter xbu meanbase,
msymbol(t) ||
```

Or, to overlay the observed *T*s along with their predicted true values assuming the fitted model is correct along with the confidence bands, similar to Figure 3, panel d:

```
. twoway || rarea confl confu meanbase || line fit meanbase || scatter diffimp meanbase
        [aw=TWr], msymbol(Oh) || scatter xbu meanbase, msymbol(t) ||
```

Appendix II: Calculating and Displaying Confidence Bands in Meta-Regression Using metafor with R

We used commands similar to these in order to create the graphs in Figures 2 and 3.

1) First install R and the package metafor, which is available via the Comprehensive R Archive Network (CRAN) at http://CRAN.R-project.org/package=metafor, the author's website at http://www.wvbauer.com/, or through a directly command within R: install.packages("metafor") (one needs an internet connection and appropriate access rights on the computer). Then open metafor and load the dataset, which in this example we have named QoL, using these commands:

```
library("metafor")
data("QoL", package="metafor")
print (QoL, row.names=FALSE)
```

2) Now fit the meta-analytic model with the quadratic effect of targeted aerobic METs and proportion of females using the command rma.uni() as follows:

```
aeromet <- rma(d, vi, mods=~aerobicmet+I(aerobicmet^2)+expwomen, subset=((design == 1
    |design == 0) & fup == 2), data = QoL, method="ME")
```

The model has 3 independent variables: aerobicmet, aerobicmet^2 (the two aerobic METs terms) and expwomen (percentage of the sample that is female). If fixed-effects assumptions are desired instead of mixed-effects, then replace ME with FE. The "subset=" portion of the command can be ignored; it is used in this case in order to narrow the sample to the cases that we wished to model.

3) To obtain the predicted values:

```
predsaeromet <- predict(aeromet, newmods=cbind(seq(1,8,.1), seq(1,8,.1)^2, 79))
wi   <- 1/sqrt(QoL$vi)
size <- 0.5 + 3 * (wi - min(wi))/(max(wi) - min(wi))
```

The value 79 is the sample mean for percentage of females.

4) Then plot the predicted values and the confidence bands:

```
plot(QoL$aerobicmet,QoL$d,type="n",xlab = "Targeted aerobic METs",
ylab = "QoL Effect Size (d)", xlim=c(1, 8), ylim=c(-0.1, 1.5))
lines(seq(1,8,.1), predsaeromet$pred, col="navy")
lines(seq(1,8,.1), predsaeromet$ci.lb, lty = "dashed", col=" blue")
lines(seq(1,8,.1), predsaeromet$ci.ub, lty = "dashed", col=" blue")
```

5) In order to plot the estimates beyond the observed range of the studies (in this case up to 12):

```
predsaeromet <- predict(aeromet, newmods=cbind(seq(1,12,.1), seq(1,12,.1)^2, 79))
wi   <- 1/sqrt(QoL$vi)
size <- 0.5 + 3 * (wi - min(wi))/(max(wi) - min(wi))
plot(QoL$aerobicmet,QoL$d,type="n",xlab = "Targeted aerobic METs",
ylab = "QoL Effect Size (d)", xlim=c(1, 12), ylim=c(-0.1, 6))
lines(seq(1,12,.1), predsaeromet$pred, col="red")
lines(seq(1,12,.1), predsaeromet$ci.lb, lty = "dashed", col=" blue")
lines(seq(1,12,.1), predsaeromet$ci.ub, lty = "dashed", col=" blue")
```

6) To plot the observe values for a particular subset of the sample, first create a subsample:

```
dwomen <- subset(QoL, expwomen==100 & (design == 1 |design == 0) & fup == 2)
dmen <- subset(QoL, expwomen==0)
```

7) Then plot studies with, for example, primarily men first:

```
plot(dmen$aerobicmet,dmen$d, pch = 21, col = "black", bg = "navy", cex = size,
```

```
xlab = "Targeted aerobic METs", ylab = "QoL", ylim=c(-0.6, 1.5), xlim=c(1, 6))
lines(seq(1,6,.1),  predsaeromet$pred, col="dark green")
lines(seq(1,6,.1),  predsaeromet$ci.lb, lty = "dashed", col=" blue")
lines(seq(1,6,.1),  predsaeromet$ci.ub, lty = "dashed", col=" blue")
abline(h = 0, lty = "dotted", col="red")
```