Supplementary Materials: Step-by-Step R Tutorial

A Gentle Introduction to Bayesian Network Meta-Analysis Using an Automated R Package

Install and Load the Relevant R Packages

The purpose of this tutorial is to illustrate how to apply BNMA using an automated *BUGSnet* R package. R is an open-source statistical software program and can be downloaded at the R Project website (http://www.r-project.org). The *BUGSnet* R package implements BNMA by running JAGS in the background, thus researchers need to install JAGS to use *BUGSnet* (https://sourceforge.net/projects/mcmc-jags/). To ensure the replicability of each step in this tutorial, users should download and install JAGS version 4.3.0 (https://sourceforge.net/projects/mcmc-jags/) as well as the latest version of *R* (version 4.0.3 or above) and *RStudio* and all relevant R packages. The *BUGSnet* R package is not on the CRAN repository and must be installed following the instructions provided on the *BUGSnet* R package homepage (https://bugsnetsoftware.github.io/instructions). These instructions may be updated with the further development of *BUGSnet* in the future. The *BUGSnet* version used in this tutorial is 1.1.0.

Once the installation is completed, *BUGSnet* needs to be loaded from the R library prior to running the R code, as follows:

```
library(BUGSnet)
```

Step-1. Specifying Research Questions and Treatment Categorization

Details can be found in the paper.

Step-2. Prepare Data for *BUGSnet*

The data should be prepared so that each row represents one arm in each study. The raw data as well as the full name of treatments used in this demonstration can be found in Appendix A. For example, in Appendix A the first two rows correspond to the two arms of study "20839311," with treatments, *psy_edu* and *CBT.exp*, respectively. The data set contains the study id (*study*), treatment names (*treatment*), and the mean and standard deviation of the outcome measured at both pre- and post-tests (*mean_pre; sd_pre; mean_post; sd_post*). We also included one treatment effect modifier, *sessions*.

The next step is to compute the mean difference scores (*mean_diff*, i.e., post-pre mean difference scores) and SD of mean difference scores (*sd_diff*). The *mean_diff* can be easily obtained by subtracting the pre-test scores from the post-test scores; *sd_diff* can be obtained via Equation 3 and the R code is provided below. The new variables, *mean_diff* and *sd_diff*, can be found in the last two columns of Table 1.

```
rho=0.5 # users can use a different value for this correlation
ptsd1$mean_diff <- ptsd1$mean_post - ptsd1$mean_pre
ptsd1$sd_diff <- sqrt(ptsd1$sd_post^2 + ptsd1$sd_pre^2
                      - 2*rho*ptsd1$sd_post*ptsd1$sd_pre)

head(ptsd1)
```

Table 1

A Sample of Restructured Data

study	treatment	n	mean_pre	sd_pre	mean_post	sd_post	sessions	mean_diff	sd_diff
20839311	psy_edu	63	79.5	15.3	74.9	19.5	6	-4.6	17.8
20839311	CBT.exp	61	81.3	14.0	74.0	20.4	6	-7.3	18.1
21874605	TAU	75	82.8	19.4	72.6	25.0	6	-10.2	22.7
21874605	mindful	71	83.1	16.2	66.2	23.6	6	-16.9	20.9
22264669	placebo	10	72.3	12.2	61.7	11.1	10	-10.6	11.7

22264669	somatic	10	81.6	9.5	53.9	15.3	10	-27.7	13.4

Step-3. Check Network Plot and Network Characteristics

The ***data.prep*** function is used to format data for making the network plot and for all the following analyses, which requires that the researcher specify the data object name, the name of the treatment column and the study id column.

```
ptsd2 <- data.prep(arm.data = ptsd1,
                   varname.t = "treatment",
                   varname.s= "study")
```

Researchers can try different scale values to visualize the network. The ***net.plot*** function is used to make the network plot; the default of this function is ***net.plot(data = , node.scale = 5, edge.scale = 2, lab.cex = 1)***. Researchers can modify these values as the following example.

```
net.plot(data = ptsd2, node.scale = 6, edge.scale = 3, node.lab.cex = 1.5)
```

The network plot is provided in main text of the paper (See Figure 2).

In addition to the network plot, *BUGSnet* allows us to see the characteristics of the network and treatments using the ***net.tab()*** function. This function requires us to provide the data object name, the column name of the outcome variable, the column name of the sample size and the type of outcome. In this demonstration, the type of outcome is continuous. Other types of outcomes are also available in *BUGSnet*, such as *binomial* for dichotomous variable, and *rate* for count variable, and *rate2* for survival analysis.

```
##Network characteristic summary tables
network.char <- net.tab(data = ptsd2,
                        outcome = "mean_diff",
                        N = "n",
                        type.outcome = "continuous")
network.char
```

The output provides three summary tables: Network Characteristics Table (Table 2), Intervention Table (Table 3), and Comparison Table (Table 4). The Comparison Table shows all the direct comparisons.

Table 2

Network Characteristic Table

Characteristic	Value
Number of Interventions	10
Number of Studies	14
Total Number of Patients in Network	1172
Total Possible Pairwise Comparisons	45
Total Number of Pairwise Comparisons With Direct Data	10
Is the network connected?	TRUE
Number of Two-arm Studies	14
Number of Multi-Arms Studies	0
Average Outcome	-17.66

Table 3

Intervention Table

treatment	n.studies	n.patients	min.outcome	max.outcome	av.outcome
CBT.exp	7	336	-49.2	-7.3	-23.9
CBT.exp.vtc	1	23	-29.4	-29.4	-29.4
CBT.sat	1	29	-37.6	-37.6	-37.6
mindful	4	186	-30.1	-8.4	-15.7
PCT	4	233	-33.6	-10.8	-17.6
placebo	1	10	-10.6	-10.6	-10.6
psy_edu	2	91	-10.3	-4.6	-6.4
somatic	2	38	-35.0	-27.7	-33.1
TAU	3	128	-10.8	-6.3	-9.5
waitlist	3	98	-14.3	-3.4	-7.5

Table 4

Comparison Table

comparison	n.studies	n.patients
CBT.exp vs. CBT.exp.vtc	1	52
CBT.exp vs. PCT	3	341
CBT.exp vs. psy_edu	1	124
CBT.exp vs. waitlist	2	152
CBT.sat vs. psy_edu	1	57

mindful vs. PCT	1	116
mindful vs. TAU	2	208
mindful vs. waitlist	1	47
placebo vs. somatic	1	20
somatic vs. TAU	1	55

Step-4. Perform BNMA Analysis

Model Specification. The ***nma.model()*** function generates BUGS code. The random effects model is defined by specifying ***effects = "random"***. We also need to provide the *data* (data object name), *outcome* (mean), *n* (sample size), *sd* and *reference* (a reference treatment condition). A control group is typically used as the comparator, but a particular treatment can be used as well. As our outcome is on continuous scale, we should choose *family = "normal"* (normal distribution) and correspondingly *link = "identity"*. The R code for the fixed effects model can be adapted by changing the argument, *effects = "fixed"*.

```
RE_model <- nma.model(data = ptsd2,
                      outcome = "mean_diff",
                      N = "n",
                      sd = "sd_diff",
                      reference = "waitlist",
                      family = "normal",
                      link = "identity",
                      effects = "random")

FE_model <- nma.model(data = ptsd2,
                      outcome = "mean_diff",
                      N = "n",
                      sd = "sd_diff",
                      reference = "waitlist",
                      family = "normal",
                      link = "identity",
                      effects = "fixed")
```

Conducting MCMC Simulation. BUGS code generated from ***nma.model()*** is used as the input for the ***nma.run()*** function, which is used to estimate the posterior distributions of parameters and thus generate the results of the network meta-analysis.

Researchers need to define several parameters of the Bayesian model in ***nma.run()*** function: *n.adapt*, *n.burnin*, *n.iter*, *n.chain*, and *thin*. JAGS has an adaptive mode, in which sampling is optimized. The argument *n.adapt* controls the length of this adaptive phase. For a simple model, *n.adapt = 100* is enough, but complex models may require a longer adaptive phase, such as *n.adapt* = 1000, which is used in this example. The argument *n.burnin = 5000* specifies that the first 5,000 iterations will be discarded to ensure that samples are retained after reaching stationarity. Then the sampling phase consists of *n.iter=25000* iterations. The argument *thin = 2* allows us to keep every 2nd value and discard all other values.

```
RE_results <- nma.run(RE_model,
                      n.adapt = 1000,
                      n.burnin = 5000,
                      n.iter = 20000,
                      n.chains = 3,
                      thin = 2)

FE_results <- nma.run(FE_model,
                      n.adapt = 1000,
                      n.burnin = 5000,
                      n.iter = 20000,
                      n.chains = 3,
                      thin = 2)
```

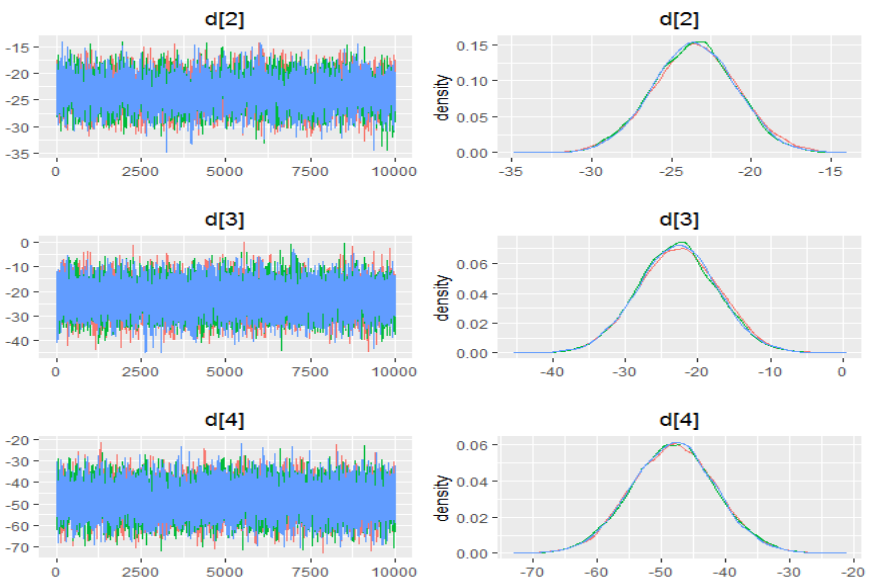
Assessing Convergence. The model convergence can be checked by trace plot, density plot, Gelman-Rubin diagnostic, and Geweke diagnostic in *BUGSnet*. To obtain the diagnostics, we need to use the posterior distributions of parameters estimated by ***nma.run()*** function as the input for the ***nma.diag()*** function.

```
nma.diag(RE_results)
nma.diag(FE_results)
```

An example of trace plots and density plots for a random mixed-effects model is provided in the main text of our paper (See Figure 3); an example for a fixed mixed-effects model is provided below.

Figure 1

Trace Plot and Density Plot Obtained from a Fixed Effects Model



The output below shows the results obtained from both Gelman-Rubin and Geweke diagnostic tools using random effects (Table 5) and fixed effects models (Table 6).

Table 5

Random effects model: Gelman-Rubin and Geweke Diagnostics

```
## $gelman.rubin
## $psrf
##       Point est. Upper C.I.
## d[2]    1.001779   1.001990
## d[3]    1.001658   1.001664
## d[4]    1.001398   1.001516
## d[5]    1.000677   1.000739
## d[6]    1.001485   1.002010
## d[7]    1.001934   1.003819
## d[8]    1.001942   1.001970
## d[9]    1.000899   1.001331
## d[10]   1.001401   1.001585
## sigma   1.006966   1.024364
##
## $mpsrf
## [1] 1.006228
##
## attr(,"class")
## [1] "gelman.rubin.results"
##
## $geweke
## $stats
##          Chain 1    Chain 2    Chain 3
```

```
## d[2]   0.2655819  0.6015692 -0.3922512
## d[3]   1.5123286  1.3879926  0.2927799
## d[4]  -0.2664212  1.7728999  0.5730064
## d[5]   0.8995889  0.1167714  0.2633151
## d[6]   1.2761186  1.0039694 -0.5528726
## d[7]   0.3362364  1.5917975  0.5982672
## d[8]   0.1969061  1.2134575 -0.1554691
## d[9]   0.6357164  1.1816888  0.4093193
## d[10]  0.7979636  0.4443575  0.2340179
## sigma  0.3064416 -0.2404249  0.5284582
##
## $frac1
## [1] 0.1
##
## $frac2
## [1] 0.5
##
## attr(,"class")
## [1] "geweke.results"
```

Table 6

Fixed effects model: Gelman-Rubin and Geweke Diagnostics

```
## $gelman.rubin
## $psrf
##       Point est. Upper C.I.
## d[2]    1.000683   1.001925
## d[3]    0.999976   1.000108
## d[4]    1.000595   1.001884
## d[5]    1.001541   1.004426
## d[6]    1.000876   1.002454
## d[7]    1.000508   1.001804
## d[8]    1.000384   1.001305
## d[9]    1.000540   1.001853
## d[10]   1.001015   1.002716
##
## $mpsrf
## [1] 1.001414
##
## attr(,"class")
## [1] "gelman.rubin.results"
##
## $geweke
## $stats
##           Chain 1    Chain 2     Chain 3
## d[2]  -0.19082338 -2.1951394  0.85640681
## d[3]   0.35013534 -0.8300362  0.24831995
## d[4]   0.80693644 -1.8182278  0.51191602
## d[5]   0.10263765 -2.6435571  0.21393167
## d[6]   0.10650830 -2.4781113  0.28353748
## d[7]   0.71800796 -1.6820201  0.15620734
## d[8]  -0.37942242 -2.0192717 -0.06735033
## d[9]   0.04963934 -1.3944001 -0.20434188
## d[10]  0.21674030 -2.2471862 -0.05778025
##
```

```
## $frac1
## [1] 0.1
##
## $frac2
## [1] 0.5
##
## attr(,"class")
## [1] "geweke.results"
```

Specifying Priors by Users. *BUGSnet* allows researchers to define their own priors if they are interested in using more informative prior or perform sensitivity analyses. For example, priors for the μ_{jb} and d_{bk} parameters could be specified as a normal distribution with mean zero and variance 10,000. Also, in the RE model, which is expressed in Equation (2), the σ^2 parameter could be given a uniform prior from 0 to 20.

```
RE_model_2 <- nma.model(data = ptsd2,
                        outcome = "mean_diff",
                        N = "n",
                        sd = "sd_diff",
                        reference = "waitlist",
                        family = "normal",
                        link = "identity",
                        effects = "random"
                        prior.mu = "dnorm(0,0.00001)",
                        prior.d = "dnorm(0,0.00001)",
                        prior.sigma = "dunif(0,20)"

RE_results <- nma.run(RE_model_2,
                        n.adapt = 1000,
                        n.burnin = 5000,
                        n.iter = 20000,
                        n.chains = 3,
                        thin = 2)
```

Step-5. Assess Model Data Fit

The ***nma.fit()*** function produces a model fit plot comparing the leverage of each data point against their contribution to the total posterior deviance. The R code is provided below. The output obtained from both FE and RE models is provided in the main text of the paper (See Figure 4).

```
par(mfrow = c(1,2))
fe_model_fit <- nma.fit(FE_results, main = "fixed effects Model")
re_model_fit <- nma.fit(RE_results, main = "random effects Model")
```

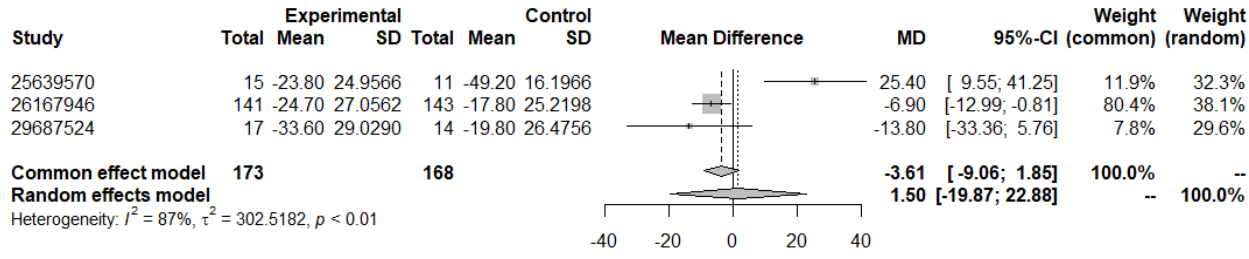
Step 6. Assumptions Checking

Checking Homogeneity Assumption for Pairwise Comparisons. Like the standard meta-analysis, we can examine the homogeneity assumption for the direct evidence. Since the methods are the same as the standard meta-analysis, we included R code and detailed explanation here. Using *mpa()* function, we can conduct the pairwise comparison and obtain Higgins & Thompson's I^2 statistic for checking the homogeneity assumption. Using the example R code below, we examined three studies with a pairwise comparison between CBT.exp and PCT. The results showed that CBT.exp achieved a much more significant reduction on PTSD symptoms than PCT in the first study, but the other two studies showed the opposite even though the third study did not have a statistically significant difference (Figure 2). The I^2 statistic = 87% suggests a substantial between-study heterogeneity, that is, about 87% of the variation is due to between-study heterogeneity. In this case, the RE model is recommended.

```
pma(data=ptsd2, name.trt1 = "CBT.exp", name.trt2 = "PCT", N = "n",
outcome = "mean_diff", sd = "sd_diff", type.outcome = "continuous",
sm = "MD")
```

Figure 2

Pairwise Comparison Meta-analysis with I^2 statistic



Exploring the Sources of Heterogeneity. Using the ***data.plot()*** function, researchers need to provide the covariate name and decide to plot by *treatment* or *study*. The number of treatment sessions, *sessions*, is used as the covariate here, and the plot is organized by *treatment*. The R code is provided below, and the output is provided in the main text of the paper (See Figure 5).

```
data.plot(data = ptsd2,
          covariate = "sessions",
          by = "treatment", #alternatively change to "study"
          avg.hline = TRUE, #add overall average line
          text.size = 11)
```

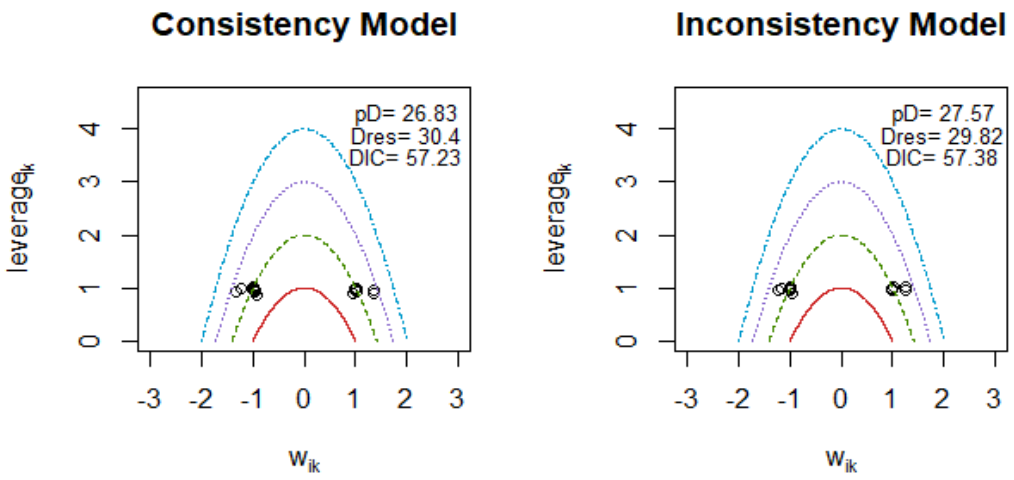
Checking Inconsistency. The R code we used in step-4 is the default that assumes the consistency assumption holds. To test the inconsistency assumption, we need to add one more argument, *type = "inconsistency"*, in the ***nma.model()*** function. The R code and the output are provided below. The output (Figure 3) shows that the fit indices obtained from both models are similar, which suggests the consistency assumption is met.

```
inconsistency_model <- nma.model(data = ptsd2,
                            outcome = "mean_diff",
                            N = "n",
                            sd = "sd_diff",
                            reference = "waitlist",
                            family = "normal",
                            link = "identity",
                            type = "inconsistency",
                            effects = "random")


inconsistency_results <- nma.run(inconsistency_model,
                                 n.adapt = 3000,
                                 n.burnin = 5000,
                                 n.iter = 20000)
par(mfrow=c(1,2))
consist_model_fit <- nma.fit(RE_results, main = "Consistency Model" )
inconsist_model_fit <- nma.fit(inconsistency_results, main = "Inconsistency
Model")
```

Figure 3

Comparing the Model Data Fit: Consistency vs. Inconsistency Models



Step-7. Summarize the Results

Forest Plot. We can produce a forest plot of the final results using the ***nma.forest()*** function. We need to use *RE_results* as the input for this function, which was obtained from Step-4. There are two options of central tendency, *mean* or *median*, for measuring relative effectiveness. Median is less affected by outliers or a skewed distribution. For this demonstration, we used the mean, *central.tdcy = "mean"*, because the distribution was not a problem. Additionally, we chose waitlist as the comparator, *comparator = "waitlist"*. The R code is provided below, and the output is provided in the main text of the paper (See Figure 6).

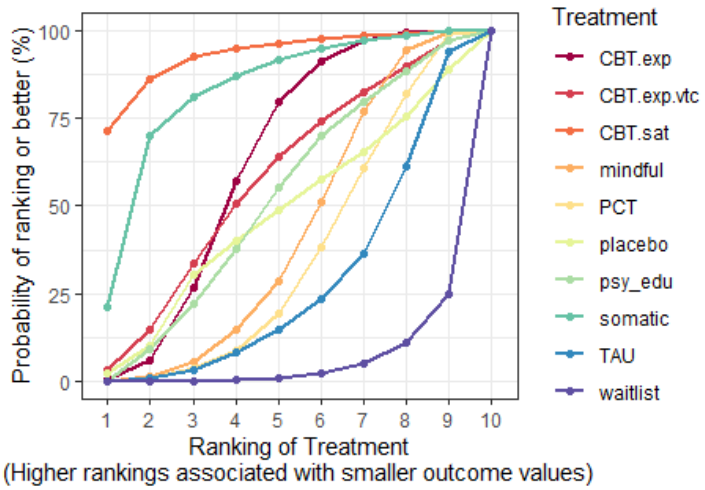
```
nma.forest(RE_results,
           central.tdcy = "median",
           comparator = "waitlist",
           log.scale = FALSE)
```

SUCRA Plot and Table. We can use the ***nma.rank()*** function to produce a summary table of ranking probabilities, a Surface Under the Cumulative Ranking (SUCRA) plot, and a summary table of cumulative probabilities. We first need to use the output obtained from the

random effects model as the input, *nma = RE_results*, and specify whether a larger probability should indicate a more effective treatment, *largerbetter = FALSE*. We use "FALSE" here because a larger reduction on the treatment effect is regarded as better. The default for the colour set is, *sucra.palette = "Set1"*. We used another colour set, *sucra.palette = "Spectral"*, which allows readers to see the treatments easily, arranged by alphabetic order and coloured from dark red to dark blue. The table of ranking probabilities can be obtained by *sucra.out$ranktable* (See Table 2 in the main text of the paper), SUCRA plot by *sucra.out$sucraplot* (Figure 4), and the table of cumulative probabilities by *sucra.out$sucratable*.

```
sucra.out <- nma.rank(RE_results,
                      largerbetter = FALSE,
                      sucra.palette = "Spectral")
sucra.out$ranktable
sucra.out$sucraplot
sucra.out$sucratable
```

Figure 4
SURCRA Plot for All the Treatments



League Table Heat Plot. *BUGSnet* package has a built-in function, ***nma.league()***, which allows us to produce a heat plot together with a pairwise comparison table for all the comparisons. The random effects model results are used as the input for this function.

Researchers can choose either *median* or *mean* as the measure of central tendency. To show the ranking order of treatment effectiveness, we used the order obtained from SUCRA, *order = sucra.out$order*. Given the large size of the table, we set up one decimal point for the numeric values. The negative values are set to the colour red (*low.colour = "red"*), the values around zero are set to white (*mid.colour = "white"*), and the positive values are set to green (*high.colour = "springgreen3"*). The R code is provided below, and the output is provided in the main text of the paper (See Figure 7).

```
league.out <- nma.league(RE_results,
                         central.tdcy = "median",
                         order = sucra.out$order,
                         log.scale = FALSE,
                         low.colour = "red",
                         mid.colour = "white",
                         high.colour = "springgreen3",
                         digits = 1)
league.out$heatplot
```

R Code for Conducting Meta-regression Analysis

As we explained in the paper, we only provided R code here and do not interpret the output because of the small number of studies and insufficient number of studies per treatment in the demonstration data. To conduct meta-regression, we can use ***nma.model()*** to create BUGS code that will be used as the input for ***nma.run()***. The reference parameter indicates the name of the treatment used as the 'referent' comparator, which can be a control condition or a particular treatment condition. The link function "identity" is used because our outcome is continuous. The covariate treatment "sessions" was chosen for this analysis. Users need to define the prior on the meta-regression coefficient and can choose one of three modeling options, "EQUAL", "EXCHANGEABLE", or "UNERLATED" for *prior.beta()*. These three options are common approaches used for meta-regression analysis. Detailed information can be found in Dias, et al. (2018).

```
nmaregmodel <- nma.model(ptsd2,
                         reference = "waitlist",
                         outcome = "mean_diff",
                         sd = "sd_diff",
                         N = "n",
                         link = "identity",
                         family = "normal",
                         effects = "random",
                         covariate = "sessions",
                         prior.beta = "EQUAL")
                        #prior.beta can also be "EXCHANGEABLE" or "UNRELATED"

nmareg_run <- nma.run(nmaregmodel,
                      n.adapt = 1000,
                      n.burnin = 5000,
                      n.iter = 20000,
                      n.chains = 3,
                      thin = 2)

nma.regplot(nmareg_run)

nma.fit(nmareg_run, main = "Random Effects Model")
```