15, Number 4, pp. 951–985

Network meta-analysis

Ian R. White
MRC Biostatistics Unit
Cambridge Institute of Public Health
Cambridge, UK
ian.white@mrc-bsu.cam.ac.uk

Abstract. Network meta-analysis is a popular way to combine results from several studies (usually randomized trials) comparing several treatments or interventions. It has usually been performed in a Bayesian setting, but recently it has become possible in a frequentist setting using multivariate meta-analysis and meta-regression, implemented in Stata with `mvmeta`. I describe a suite of Stata programs for network meta-analysis that perform the necessary data manipulation, fit consistency and inconsistency models using `mvmeta`, and produce various graphics.

Keywords: st0410, network components, network convert, network forest, network import, network map, network meta, network pattern, network query, network rank, network setup, network sidesplit, network table, network unset, network meta-analysis, multiple treatments meta-analysis, mixed-treatment comparisons

1 Introduction

Network meta-analysis, also called multiple treatments meta-analysis or mixed-treatment comparisons, is a popular way to combine evidence on multiple studies (usually randomized trials) comparing multiple treatments (or other interventions). Its key feature is the ability to combine direct and indirect evidence; for example, the comparison of treatments A and B is performed both using studies that directly compare A with B (direct evidence) and using studies that compare A with C and B with C (indirect evidence). Good general introductions are given by Mills, Thorlund, and Ioannidis (2013) for the concepts and Salanti et al. (2008) for the statistical methods.

A key issue in network meta-analysis is whether the network is consistent—that is, whether the direct evidence agrees with the indirect evidence (and, if there are multiple sources of indirect evidence, whether they agree with each other). Statistical models for inconsistency have been proposed and can be used to assess consistency (Lu and Ades 2006; Higgins et al. 2012).

Estimation of network meta-analysis models has usually been done in a Bayesian framework, with fitting in WinBUGS (Lu and Ades 2004). Frequentist estimation is possible by expressing the consistency and inconsistency models as multivariate random-effects meta-analysis or meta-regression (White et al. 2012). Graphical methods are well developed for presenting the results of network meta-analysis (Salanti, Ades, and Ioannidis 2011), although the individual study data are often not displayed.

My `mvmeta` package performs multivariate meta-analysis and meta-regression (White 2009, 2011, 2015). It can therefore be used to perform frequentist estimation of network meta-analysis models (White et al. 2012). However, difficulties remain in getting the data into the correct format and in specifying the `mvmeta` models. Many graphical tools for displaying the evidence base and the results of network meta-analysis have been written by Chaimani et al. (2013). Estimation of indirect treatment comparisons along a single path can also be done using the `indirect` command (Miladinovic et al. 2014).

In this paper, I introduce a suite of programs for performing network meta-analysis in Stata. The main aim of these programs is to provide a convenient tool that 1) performs the necessary data manipulation, allowing different data formats, 2) fits consistency and inconsistency models using `mvmeta`, and 3) produces various graphics, including a display of the individual study data. I also point out some methodological advances.

2 Model for network meta-analysis

I briefly describe the general model here; more details are given by White et al. (2012). The general model is a model for treatment contrasts (the "contrast-based" model of Salanti et al. [2008]). It allows for both heterogeneity (variation in the true treatment effect between studies) and inconsistency (additional variation in the true treatment effect between designs), where a design is the set of treatments compared in a study.

Consider a network including a total of T treatments: A, B, C, etc. Any treatment can be chosen as a reference treatment; for simplicity, let's choose A. Initially, assume that treatment A is included in every study. Let $d = 1, \ldots, D$ index the designs. Let y_{di}^{AJ} be the observed contrast of treatment J (J = B, C, ...) with treatment A in the ith study in the dth design. y_{di}^{AJ} may represent any measure, such as a mean difference, a standardized mean difference, a log risk-ratio, or a log odds-ratio.

The model for the observed data is

$$y_{di}^{AJ} = \delta^{AJ} + \beta_{di}^{AJ} + \omega_d^{AJ} + \varepsilon_{di}^{AJ}, \quad J = \mathrm{B}, \mathrm{C}, \ldots \tag{1}$$

where the meaning of each term is described below. Equivalently, in vector notation with $\mathbf{y}_{di} = (y_{di}^{AB}, y_{di}^{AC}, \ldots)'$, the model is

$$\mathbf{y}_{di} = \boldsymbol{\delta} + \boldsymbol{\beta}_{di} + \boldsymbol{\omega}_d + \boldsymbol{\varepsilon}_{di} \tag{2}$$

where $\boldsymbol{\delta} = (\delta^{AB}, \delta^{AC}, \ldots)'$, $\boldsymbol{\beta}_{di} = (\beta_{di}^{AB}, \beta_{di}^{AC}, \ldots)'$, $\boldsymbol{\omega}_d = (\omega_d^{AB}, \omega_d^{AC}, \ldots)'$, and $\boldsymbol{\varepsilon}_{di} = (\varepsilon_{di}^{AB}, \varepsilon_{di}^{AC}, \ldots)'$ are described below.

Treatment contrasts. In (1) and (2), δ^{AJ} represents a treatment contrast (a summary effect) between J and A. The δ^{AJ} ($J = \mathrm{B}, \mathrm{C}, \ldots$) are regarded as fixed parameters and are the parameters of primary interest.

Heterogeneity. In (1) and (2), β_{di}^{AJ} represents heterogeneity in the J–A contrast between studies within designs. The heterogeneity terms $\boldsymbol{\beta}_{di}$ are taken as random effects

$$\boldsymbol{\beta}_{di} \sim N(\mathbf{0}, \boldsymbol{\Sigma}) \tag{3}$$

as in the conventional random-effects model for meta-analysis. Model (3) without constraint on $\boldsymbol{\Sigma}$ (the "unstructured" model) allows each contrast to have a different heterogeneity variance. The positive definiteness of $\boldsymbol{\Sigma}$ ensures the "second-order consistency conditions" of Lu and Ades (2009). However, there are rarely enough studies to identify the unstructured model, and it is usual to assume that all treatment contrasts have the same heterogeneity variance τ^2. Hence, it is usual to assume that

$$\boldsymbol{\Sigma} = \tau^2 \mathbf{P} \tag{4}$$

where $\mathbf{P}$ is a matrix with all diagonal entries equal to 1 and all off-diagonal entries equal to 0.5 (Higgins and Whitehead 1996).

Inconsistency. In (1) and (2), ω_d^{AJ} represents inconsistency in the J–A contrast between designs. The inconsistency terms ω_d^{AJ} are taken to be fixed parameters (White et al. 2012), although random inconsistency terms are also possible (Lumley 2002; Jackson et al. 2014). We can include a maximal set of inconsistency parameters, which White et al. (2012) term the "design-by-treatment interaction model", or a smaller set (Lu and Ades 2006). For the consistency model, we set $\omega_d^{AJ} = 0$ for all d, J.

Within-study error. In (1) and (2), ε_{di}^{AJ} is a within-study error term. We assume $\boldsymbol{\varepsilon}_{di} \sim N(0, \mathbf{S}_{di})$, where $\mathbf{S}_{di}$ is assumed to be known.

Two missing-data problems can arise. First, design d may contain the reference treatment A but not some other treatments. Here we use the likelihood implied by (2) for the observed subvector of $\mathbf{y}_{di}$. A harder problem arises when design d excludes A. Here we still use (2), but either we add a reference treatment arm with a very small amount of data (the "augmented" approach) or we apply the model only to the contrasts that are actually estimated in each particular design (the "standard" approach), as shown in section 3.1.

3 The network commands

3.1 Data formats

I illustrate the data formats with the smoking data that were given by Hasselblad (1998) and used by Lu and Ades (2006) and White (2011). The raw data comprise the number of individuals randomized to a treatment (`n`) and the number of those individuals who are quitting smoking (`d`) for each arm of each study. The four treatments are here coded A, B, C, and D. For brevity, I show only the first four studies. The raw data can be stored in a long format, as follows:

```
. list if study<=4, noobs clean
  study   treat    d     n
      1       A    9   140
      1       C   23   140
      1       D   10   138
      2       B   11    78
      2       C   12    85
      2       D   29   170
      3       A   79   702
      3       B   77   694
      4       A   18   671
      4       B   21   535
```

Alternatively, the data can be stored in a wide format:

```
. list if study<=4, noobs clean
  study   dA    nA   dB    nB   dC    nC   dD    nD
      1    9   140    .     .   23   140   10   138
      2    .     .   11    78   12    85   29   170
      3   79   702   77   694    .     .    .     .
      4   18   671   21   535    .     .    .     .
```

Note that studies 1 and 2 are three-arm studies and the others are two-arm studies.

The `network` suite uses three data formats: augmented, standard, and pairs formats. They are illustrated here with the log odds-ratio as the effect measure.

In the augmented format, all treatments are compared with a reference treatment (here, treatment A), and studies without the reference treatment (for example, study 2) have a reference treatment arm created with a small amount of data (White 2011). In the listing below, arm A has been created in study 2 with 0.001 observations and mean 0.156. Usually, augmentation has a negligible effect on results (see section 4.1), but in some models, unidentified parameters can be estimated with large standard errors.

```
. list study _y* _S_B_B - _S_C_D if study<=4, noobs clean
  study    _y_B    _y_C   _y_D    _S_B_B    _S_B_C    _S_B_D    _S_C_C    _S_C_D
      1       .    1.05   0.13         .         .         .      0.17      0.12
      2   -0.12   -0.12   0.11   7589.01   7588.90   7588.90   7589.00   7588.90
      3   -0.02       .      .      0.03         .         .         .         .
      4    0.39       .      .      0.11         .         .         .         .
. * _S_D_D omitted to save space
```

In the standard format, each study has its own reference treatment to which the other treatments are compared. Unlike for the augmented format, the treatment contrast (for example, variable _y_1 in the output below) represents different contrasts in different studies, and variable `_contrast_1` specifies the contrast. Three-arm studies also have values for _y_2 representing the contrast specified by `_contrast_2`.

```
. list study _contrast* _y* _S* if study<=4, noobs clean abbreviate(12)
  study   _contrast_1   _contrast_2    _y_1   _y_2   _S_1_1   _S_1_2   _S_2_2
      1         C - A         D - A    1.05   0.13     0.17     0.12     0.23
      2         C - B         D - B    0.00   0.23     0.20     0.11     0.15
      3         B - A                 -0.02      .     0.03        .        .
      4         B - A                  0.39      .     0.11        .        .
```

In the above formats, there is one record for each study. In the pairs format, used by Chaimani et al. (2013), there is one record for each possible contrast in each study, meaning that two-arm studies have a single record but three-arm studies have three records. Again, variable `_contrast` labels the contrasts:

```
. list study _contrast _y _stderr if study<=4, noobs clean abbreviate(12)
    study   _contrast      _y   _stderr
        1       C - A    1.05      0.41
        1       D - A    0.13      0.48
        1       D - C   -0.92      0.40
        2       C - B    0.00      0.45
        2       D - B    0.23      0.38
        2       D - C    0.22      0.37
        3       B - A   -0.02      0.17
        4       B - A    0.39      0.33
```

3.2 The network setup command

`network setup` imports data from a set of studies reporting count data (events, total number) or quantitative data (mean, standard deviation, total number) for two or more treatments. The data may be in long format (one record per treatment per study) or in wide format (one record per study) and may be imported into the augmented, standard, or pairs format.

After running `network setup`, the dataset contains various settings that are required by subsequent `network` commands; the settings are stored as characteristics and may be viewed using `network query`. In particular, each treatment has a code (typically A, B, C, but numerical codes are possible) and a name (typically a descriptive string). Subsequent analyses always use the treatment codes, while descriptive and graphical commands by default use the treatment names.

Syntax

For count data:

`network setup` *eventvar nvar* [*if*] [*in*], `studyvar(`*varname*`)` [`or` | `rr` | `rd` | `hr` `zeroadd(`#`)` *common_options*]

For quantitative data:

`network setup` *meanvar sdvar nvar* [*if*] [*in*], `studyvar(`*varname*`)` [`md` | `smd` *common_options*]

If the data are in wide format, then *eventvar nvar* or *meanvar sdvar nvar* are stubs for variable names and `trtvar(`*varname*`)` is not specified. For example, in the wide format in section 3.1, `network setup d n, studyvar(study)` would be appropriate.

If the data are in long format, then *eventvar nvar* or *meanvar sdvar nvar* are the variable names and `trtvar(`*varname*`)` is required. For example, in the long format in section 3.1, `network setup d n, studyvar(study) trtvar(treat)` would be appropriate.

common_options are the following:

`trtvar(`*varname*`) armvars(drop|keep[(`*varlist*`)]) trtlist(`*string*`) alpha numcodes nocodes format(augmented|standard|pairs) genprefix(`*string*`) gensuffix(`*string*`) ref(`*string*`) augment(#) augmean(#) augsd(#) augoverall`

Options describing the data

`studyvar(`*varname*`)` specifies the study variable. `studyvar()` is required.

`trtvar(`*varname*`)` specifies the treatment variable (implies the long format).

`armvars(drop|keep[(`*varlist*`)])` is relevant only when the data are in long format and there are extra arm-level variables in the data. In this case, the easiest option is `armvars(drop)` to drop all extra arm-level variables.

Options specifying how treatments are coded

`trtlist(`*string*`)` specifies the list of treatment names to be used, which is useful if you want to omit some treatments, for example, for a sensitivity analysis. It is also useful to specify how the treatments will be coded (first treatment will be A, B, etc.). The default is to use all treatments found in alphabetical order; except that when `trtvar()` is numeric, the default is to use all treatments in numerical order.

`alpha` forces treatments to be coded in alphabetical order. This is the default except in long format when `trtvar()` is numeric with value labels.

`numcodes` codes treatments as numbers 1, 2, 3, ... or (if more than 9 treatments) 01, 02, 03, ... The default is to code treatments as letters A, B, C, ... or (if more than 26 treatments) AA, AB, AC, ...

`nocodes` uses the current treatment names as treatment codes. Treatment names are modified only if this is needed to make them valid Stata names. This option becomes increasingly awkward as treatment names become longer.

Options for count data

`or` specifies that the treatment effect be measured by the log odds-ratio (the default).

`rr` specifies that the treatment effect be measured by the log risk-ratio.

`rd` specifies that the treatment effect be measured by the risk difference.

`hr` specifies that the treatment effect be measured by the log hazard-ratio (also called the log rate-ratio). In this case, *nvar* must be the total person-time at risk, not the number of individuals.

`zeroadd(#)` specifies the number of successes and (except with the `hr` option) failures added to all arms of any study that contains a zero cell in any arm. The default is `zeroadd(0.5)`.

Options for quantitative data

`md` specifies that the treatment effect be measured by the mean difference (the default).

`smd` specifies that the treatment effect be measured by the standardized mean difference, defined as the mean difference divided by the standard deviation, where the latter is computed pooled across all study arms. The formulas for Hedges's g in White and Thomas (2005) are used. These are unbiased estimators and involve corrections for small numbers of degrees of freedom. The covariance between g_1 and g_2 is taken as $J(\nu)^2\left\{\frac{\nu}{(\nu-2)N_0}+g_1g_2V(\nu)\right\}$, where ν is the degrees of freedom used to estimate the pooled standard deviation, N_0 is the sample size in the common reference group, and $V(\nu)$ and $J(\nu)$ are defined in White and Thomas (2005).

Use of the standardized mean difference has problems (Greenland, Schlesselman, and Criqui 1986). A new alternative is given by Lu, Brazier, and Ades (2013) and Lu, Kounali, and Ades (2014).

`sdpool(on|off)` specifies whether the standard deviation is pooled across arms in computing variances. The default, which follows `metan`, is `sdpool(off)` with `md` and `sdpool(on)` with `smd`. For multiarm studies, `sdpool(on)` pools across all arms.

Options for the format after setting up

`format(augmented|standard|pairs)` specifies the required format.

`genprefix(`*string*`)` specifies the prefix to be used before the default variable names (for example, `y` for treatment contrasts). The default is `genprefix(_)`, where treatment contrasts are named `_y*`, etc.

`gensuffix(`*string*`)` specifies the suffix to be used after the default variable names. The default is no suffix.

Options for augmented format

`ref(`*trtname*`)` specifies the name of the reference treatment.

`augment(`*#*`)` specifies the number of individuals to use when augmenting missing reference treatment arms. The default is `augment(0.001)`.

`augmean(`*#*`)` specifies the mean outcome to use when augmenting missing reference treatment arms. The default is for each augmented study to use the weighted average of its arm-specific means.

`augsd(`*#*`)` applies only for quantitative data and indicates the standard deviation to use when augmenting missing reference treatment arms. The default is for each augmented study to use the weighted average of arm-specific standard deviations.

`augoverall` changes the default behavior for `augmean()` and `augsd()` to use the overall mean and standard deviation across all studies.

3.3 The network map command

`network map` draws a map of a network; that is, it shows which treatments are directly compared against which other treatments, and roughly how much information is available for each treatment and for each treatment comparison. `network map` works by calling `networkplot` (Chaimani et al. 2013; Chaimani and Salanti 2015) and has all the facilities of that command for displaying quantity and quality of evidence through weighting and coloring. `network map`'s contribution is to offer more options for treatment placement, although better methods are available (Rücker and Schwarzer Forthcoming).

Syntax

`network map` [*if*] [*in*] [`,`
`circle`[`(`*#*`)`] | `square`[`(`*#*`)`] | `triangular`[`(`*#*`)`] | `random`[`(`*#*`)`] `centre`
`loc(`*matname*`)` `replace` `improve` `listloc` `trtcodes` *graph_options*
networkplot_options]

Options

`circle`[`(`*#*`)`] specifies that the treatments be placed around a circle. This is the most commonly used system and the default. The optional argument specifies the number of locations; the default is the number of treatments.

`square`[`(`*#*`)`] specifies that the treatments be placed in a square lattice. The optional argument specifies the number of rows and columns; the default is the square root of the number of treatments (rounded up).

`triangular`[`(#)`] specifies that the treatments be placed in a triangular lattice. The optional argument specifies the number of rows and the maximum row lengths; the default is approximately the square root of the number of treatments.

`random`[`(#)`] specifies that the treatments be randomly placed. The optional argument specifies the number of locations; the default is the number of treatments.

`centre`, only for use with `circle(#)`, specifies to also place a treatment in the center.

`loc(`*matname*`)` specifies a treatment location matrix. This specifies where the treatments are placed on the map. The matrix should have at least as many rows as treatments and three columns containing the x coordinate, the y coordinate, and the clock position for each label. If the matrix does not exist, or if `replace` is specified, then a new matrix is created. If `loc()` is not specified, then a new matrix is created and stored in `_network_map_location`.

Note that the rows of *matname* are taken as the locations of the treatments in alphabetical order; row names are ignored. Thus, `network map, loc(M)` and `network map if useit, loc(M)` may place the treatments differently.

`replace` specifies that a new treatment location matrix be created.

`improve` requests an iterative procedure to improve the placement of the treatments. The algorithm swaps pairs of treatments if this reduces the number of line crossings. With this option, it is useful to increase the number of locations above the default; for example, in section 4.2 with eight treatments, we use `triangular(5)`.

`listloc` specifies to print the treatment location matrix.

`trtcodes` specifies to use treatment codes rather than full treatment names.

graph_options are any of the options documented in [G-3] ***twoway_options***.

networkplot_options are any of the options documented in `help networkplot`, such as `edgeweight` and `nodeweight` (controlling which edges are thicker than others and which nodes are larger than others) and `edgecolor` (allowing edges to be colored according to evidence quality).

3.4 The network meta command

`network meta` defines and fits a consistency or inconsistency model. It can handle data in any of the three network formats. If data are in the augmented or standard formats, then the models are fit using `mvmeta`; if data are in the pairs format, then the models are fit using `metareg`. `mvmeta` or `metareg` must be installed. After fitting the model, the `mvmeta` or `metareg` command used can be recalled by pressing *F9*. `network meta` stores the results for use in `network forest` and `network rank`.

For inconsistency models, the design-by-treatment interaction model of Higgins et al. (2012) is used, unless the `luades` option is specified (see below). A Wald test for inconsistency is defined and performed; the command to test for inconsistency can be recalled by pressing *F8*.

Results using the three formats should be almost identical, except that results using the pairs format are wrong in the presence of multiarm studies; in this case, `network meta` issues a warning and stops but can be overruled with the `force` option.

Syntax

`network meta consistency|inconsistency` [*if*] [*in*] [`, regress(`*varlist*`)` `luades`[`(`*trtlist*`)`] `force nowarnings` *mvmeta_options*]

Options

`regress(`*varlist*`)` specifies covariates for network meta-regression. Every treatment contrast is allowed to depend on the covariate(s) listed. This option is currently only allowed in the augmented format.

`luades`[`(`*trtlist*`)`], for the inconsistency model, specifies the model of Lu and Ades (2006) as formalized by White et al. (2012). With only two-arm studies, this is the same as the design-by-treatment interaction model of White et al. (2012). With multi-arm studies, the Lu–Ades model is smaller than the design-by-treatment interaction model and depends on the treatment ordering. The optional argument specifies an ordering of the treatments. This option is only available in the augmented format.

`force` (not recommended) forces model fitting when `network meta` detects a difficulty. This could be a disconnected network; no degrees of freedom for inconsistency when an inconsistency model is specified; or no degrees of freedom for heterogeneity when a random-effects model is specified.

`nowarnings` (not recommended) suppresses warning messages.

mvmeta_options are any of the options documented in `help mvmeta`, such as `bscov()`. The default is to assume a common heterogeneity variance. This is implemented using `bscov(exch 0.5)`, a new shorthand for `bscov(prop P)`, where `P` is the matrix defined in (4).

3.5 The network rank command

`network rank` is used to rank treatments. It works only when the augmented format has been used to fit the model. Details are given in White (2011).

Syntax

`network rank min|max` [*if*] [*in*] [`, trtcodes` *mvmeta_pbest_options*]

Use `network rank min` if the best treatment is that with the lowest (most negative) treatment effect, and use `network rank max` if the best treatment is that with the highest (most positive) treatment effect.

Options

`trtcodes` specifies to use treatment codes rather than full treatment names.

mvmeta_pbest_options are any of the suboptions available for the `pbest()` option of `mvmeta`, but note that `network meta` makes sensible default choices for `if`, `in`, `zero`, and `id()`, which it would be unwise to change. The following options are likely to be useful:

`all` reports probabilities for all ranks. The default is to report only the probabilities of being the best treatment.

`reps(#)` sets the number of replicates; larger numbers reduce Monte Carlo error.

`seed(#)` sets the random-number seed for reproducibility.

`bar` draws a bar graph of ranks.

`line` draws a line graph of ranks.

`cumulative` makes the bar or line graph show cumulative ranks.

`predict` ranks the true effects in a future study with the same covariates, thus allowing for heterogeneity as well as parameter uncertainty, as in the calculation of prediction intervals (Higgins, Thompson, and Spiegelhalter 2009). The default behavior is instead to rank linear predictors and does not allow for heterogeneity.

`meanrank` tabulates the mean rank and the SUCRA (Salanti, Ades, and Ioannidis 2011). The SUCRA is the rescaled mean rank: it is 1 when a treatment is certain to be the best and 0 when a treatment is certain to be the worst.

`saving(`*filename*`[ , replace ])` writes the draws from the posterior distribution (indexed by the identifier and the replication number) to *filename*. The `replace` option allows an existing *filename* to be overwritten.

`clear` loads the rank data into memory and specifies the commands needed to reproduce the table and graph.

`mcse` adds the Monte Carlo standard errors to the tables.

3.6 The network sidesplit command

`network sidesplit` fits the node-splitting model of Dias et al. (2010), for whom a "node" is a treatment contrast, for example, B versus A. I call this "side-splitting" because treatment contrasts are sides in the network map. To split the side B versus A, different parameters are used for the contrast of B versus A in studies containing both A and B (the direct parameter) and in other studies (the indirect parameter). The two parameters are estimated jointly and reported together with their difference and a test of whether the true difference is 0.

`network sidesplit` currently only works with data in the augmented format.

Multi-arm studies

Multi-arm studies complicate this procedure. Suppose there is a study of A versus B versus C, and we split the B versus A contrast. The first part of table 1 shows the parameterization of the side-splitting model as proposed by Dias et al. (2010). With this model specification, the C versus A contrast is assumed to be the same in "direct" and "indirect" studies, while the C versus B contrast is allowed to differ. This model can be conceptualized by regarding B as a different treatment—say, B*—in the direct studies, where B* is ω units higher than B. But there is no reason why we should not reverse the roles of A and B; then the side-splitting model regards A, rather than B, as a different treatment A*, which is ω units lower than A. This gives the parameterization in the second block of table 1: here the C versus B contrast is assumed to be the same in direct and indirect designs, while the C versus A contrast is allowed to differ.

Table 1. Parameterizations of side-splitting models

Study	B versus A	C versus A	C versus B
Split B versus A as in Dias et al. (2010)			
ABC (direct)	$\delta^{AB} + \omega$	δ^{AC}	$\delta^{AC} - \delta^{AB} - \omega$
AC (indirect)		δ^{AC}	
BC (indirect)			$\delta^{AC} - \delta^{AB}$
Split A versus B as in Dias et al. (2010)			
ABC (direct)	$\delta^{AB} + \omega$	$\delta^{AC} + \omega$	$\delta^{AC} - \delta^{AB}$
AC (indirect)		δ^{AC}	
BC (indirect)			$\delta^{AC} - \delta^{AB}$
Symmetrical alternative			
ABC (direct)	$\delta^{AB} + \omega$	$\delta^{AC} + \omega/2$	$\delta^{AC} - \delta^{AB} - \omega/2$
AC (indirect)		δ^{AC}	
BC (indirect)			$\delta^{AC} - \delta^{AB}$

I propose a small change that treats A and B symmetrically (last block of table 1). Here, instead of allocating ω in the direct studies either fully to the C versus A contrast or fully to the C versus B contrast, it is shared between them. This model can be conceptualized by regarding both A and B as different treatments in the direct studies, where B* is $\omega/2$ units higher than B and A* is $\omega/2$ units lower than A. This method is intermediate between the two alternative ways (splitting B versus A and splitting A versus B) to implement the method of Dias et al. (2010). The symmetrical method is the default but can be changed using the `nosymmetric` option.

Syntax

`network sidesplit` *trtcode1 trtcode2*|`all` [*if*] [*in*] [`, show nosymmetric tau` *mvmeta_options*]

`network sidesplit` *trtcode1 trtcode2* fits a single node-splitting model, while `network sidesplit all` fits all appropriate node-splitting models.

Options

`show` shows the `mvmeta` calculation(s) and results.

`nosymmetric` uses the node-splitting model as originally specified by Dias et al. (2010), rather than the symmetrizing modification described above.

`tau` specifies to additionally display tau, the standard deviation of the between-studies heterogeneity.

mvmeta_options are any of the options documented in `help mvmeta`.

3.7 The network forest command

`network forest` draws a forest plot of network meta-analysis data, extending the idea of Hawkins et al. (2009). For each contrast for which there is direct evidence (that is, which is estimated within one or more studies), the forest plot displays the following results:

1. "Studies": each study contributing direct evidence, grouped by design (that is, set of treatments in a study);
2. "Pooled within design": the pooled treatment effect in each design, estimated by the model most recently fit using `network meta inconsistency`; and
3. "Pooled overall": the overall treatment effect, estimated by the model most recently fit using `network meta consistency`.

Each of results 1, 2, and 3 is displayed as a point estimate and 95% confidence interval (or other confidence level determined by `set level` or the `level(#)` option). The marker representing each point estimate has size proportional to the inverse square of the standard error. Because pooled estimates (results 2 and 3) allow for between-studies heterogeneity, they may have wider confidence intervals and smaller markers than study-specific estimates (result 1).

Syntax

`network` `forest` [*if*] [*in*] [`,` `consistency(off)` `inconsistency(off)` `list` `clear` `colors(`*string*`)` `contrastoptions(`*string*`)` `trtcodes` `contrastpos(`*#*`)` `ncolumns(`*#*`)` `columns(`*string*`)` `force` `diamond` `group(design|type)` `eform` *graph_options*]

Options controlling the summary treatment contrasts

`consistency(off)` omits the "Pooled overall" summaries from the forest plot.

`inconsistency(off)` omits the "Pooled within design" summaries from the forest plot.

Options controlling output

`list` lists the data for the forest plot.

`clear` clears the current data from memory so that the data for the forest plot can be loaded into memory. The `forestplot` command can then be recalled by pressing *F9*.

Options controlling graph appearance

`colors(`*string*`)` specifies up to three colors for the "Studies" results, "Pooled within design" results, and "Pooled overall" results, respectively. The default is `colors(blue green red)`.

`contrastoptions(`*string*`)` are options for the text identifying the contrasts (for example, "C vs. B"). Any marker label options are possible, for example, `contrastoptions(mlabsize(large) mlabcolor(red))`.
See [G-3] ***marker_label_options***.

`trtcodes` specifies use of treatment codes rather than full treatment names.

`contrastpos(`*#*`)` specifies the value of the horizontal axis at which the text identifying the contrasts (for example, "C vs. B") is centered.

`ncolumns(`*#*`)` specifies the number of columns for the display. The default is automatically determined so that the number of rows per column is approximately 10 times the number of columns.

`columns(`*string*`)` specifies how to assign contrasts to columns. `columns(xtile)` assigns contrasts to columns in order using `xtile` and can lead to very unbalanced columns (that is, much more forest in one column than another). `columns(smart)` assigns contrasts to columns to optimize balance without keeping the logical order of the contrasts (so, for example, column 1 may contain "B vs. A" and "D vs. A" while column 2 contains "C vs. A"). The default is `columns(smart)`.

`force`, only relevant when `xlabel()` is specified with a numlist, forces confidence intervals to not be truncated within the specified range. By default, confidence intervals are truncated within the range implied by `xlabel()`; truncated confidence intervals are indicated by arrows.

`diamond` specifies that summaries ("Pooled by design" and "Pooled overall") be displayed as diamonds. This is useful for monochrome printing.

`group(design|type)` specifies that, within comparisons, the forest plot may be ordered by design (showing the summary for each design after the studies for that design) or by type (showing all the studies and then all the summaries). The default is `group(design)` if inconsistency results are shown and `group(type)` otherwise.

`eform` labels the horizontal axis using the exponential of the values.

graph_options are any of the options documented in [G-3] ***twoway_options***, such as `xlabel()`, `xline(0)`, or `legend(pos(3) col(1))`. The option most often needed is `msize(`*markersizestyle*`)` to change the marker sizes; the default is `msize(*0.2)`, so try, for example, `msize(*0.15)` or `msize(*0.3)`.

3.8 Utility commands

`network import` imports a dataset, either of pairwise comparisons using the syntax

```
network import [if] [in], studyvar(varname) treat(trtvar1 trtvar2)
    effect(varname) stderr(varname) [options]
```

where `treat(`*trtvar1 trtvar2*`)` specifies that the record compares *trtvar2* with *trtvar1*, `effect(`*varname*`)` specifies the point estimate for this comparison, and `stderr(`*varname*`)` specifies its standard error; or of comparisons with a reference treatment using the syntax

```
network import [if] [in], studyvar(varname) effect(effectstub)
    variance(varstub) ref(string) [options]
```

where variables *effectstub_** contain the point estimates of the comparisons with reference, variables *varstub_*_** contain their variances and covariances, and `ref()` is the reference treatment used. See `help network import` for the complete list of options and their descriptions.

`network convert` converts between the three formats described. The syntax is

```
network convert augmented|standard|pairs [, large(#) ref(trtcode)]
```

The options are only relevant when converting to augmented format (see `help convert` for details). `large(#)` specifies the value used for the variance of contrasts with the reference treatment in studies without the reference treatment; the default is `large(10000)`. `ref(`*trtcode*`)` specifies a new reference treatment.

`network query` displays the current network settings.

`network unset` removes the network settings.

`network table` tabulates network data. Data are reformatted and displayed using `tabdisp`. The syntax is

`network table` [*if*] [*in*], [`trtcodes` *tabdisp_options*]

where `trtcodes` specifies to use treatment codes rather than treatment names, and *tabdisp_options* are any of the options documented in `help tabdisp` except `cellvar()`. For example, `cellwidth(#)` may be useful to increase the column width to accommodate treatment names, and `stubwidth(#)` may be useful to increase the width of the study name column.

`network pattern` shows which treatments are used in which studies. This is done using the utility `misspattern`, which can display general patterns of missing data. The syntax is

`network pattern` [*if*] [*in*] [, `trtcodes` *misspattern_options*]

where `trtcodes` specifies to use treatment codes rather than treatment names, and *misspattern_options* are any of the options documented in `help misspattern`.

3.9 Requirements

Various parts of the `network` package require `mvmeta` version 3.1 or greater (White 2009, 2011, 2015), `metareg` (Harbord and Higgins 2008), and `networkplot` version 1.2 or greater (Chaimani et al. 2013; Chaimani and Salanti 2015).

4 Examples

4.1 Smoking network

I demonstrate the `network` package using the smoking data (Lu and Ades 2006), starting with the data in wide format, with `study` and `trt` as identifiers, `d` containing the number of events, and `n` containing the total number. In this version of the data, the treatments are coded 1–4 with labels "No contact", "Self help", "Individual counselling", and "Group counselling". `network setup` produces a dataset ready for `mvmeta`, coding the treatments A–D, using treatment A as reference and using the odds ratio as the measure of effect.

```
. use smoking
(Smoking data from Lu & Ades (2006))

. network setup d n, studyvar(study) trtvar(trt)
Treatments used
   A (reference):                    No contact
   B:                                Self help
   C:                                Individual counselling
   D:                                Group counselling
```

```
Measure                                 Log odds ratio
Studies
   ID variable:                         study
   Number used:                         24
   IDs with zero cells:                 9 20
   - count added to all their cells:    .5
   IDs with augmented reference arm:    2 21 22 23 24
   - observations added:                0.001
   - mean in augmented observations:    study-specific mean
Network information
   Components:                          1 (connected)
   D.f. for inconsistency:              7
   D.f. for heterogeneity:              16
Current data
   Data format:                         augmented
   Design variable:                     _design
   Estimate variables:                  _y*
   Variance variables:                  _S*
   Command to list the data:            list study _y* _S*, noo sepby(_design)
```

Next, we tabulate the data, graph the patterns (shown in figure 1), and draw a network map (shown in figure 2):

```
. network table
```

	Treatment and Statistic							
	− Group −		− Indivi −		− No con −		− Self h −	
study	d	n	d	n	d	n	d	n
1	10	138	23	140	9	140		
2	29	170	12	85			11	78
3					79	702	77	694
4					18	671	21	535
5					8	116	19	146
6			363	714	75	731		
7			9	205	2	106		
8			237	1561	58	549		
9			9.5	49	.5	34		
10			31	98	3	100		
11			26	95	1	31		
12			17	77	6	39		
13			134	1031	95	1107		
14			35	504	15	187		
15			73	675	78	584		
16			54	888	69	1177		
17			107	761	64	642		
18			8	90	5	62		
19			34	237	20	234		
20	9.5	21			.5	21		
21			16	43			20	49
22	32	127					7	66
23	20	74	12	76				
24	3	26	9	55				

```
. network pattern
```

Figure 1. Network pattern for smoking data

```
. network map
```

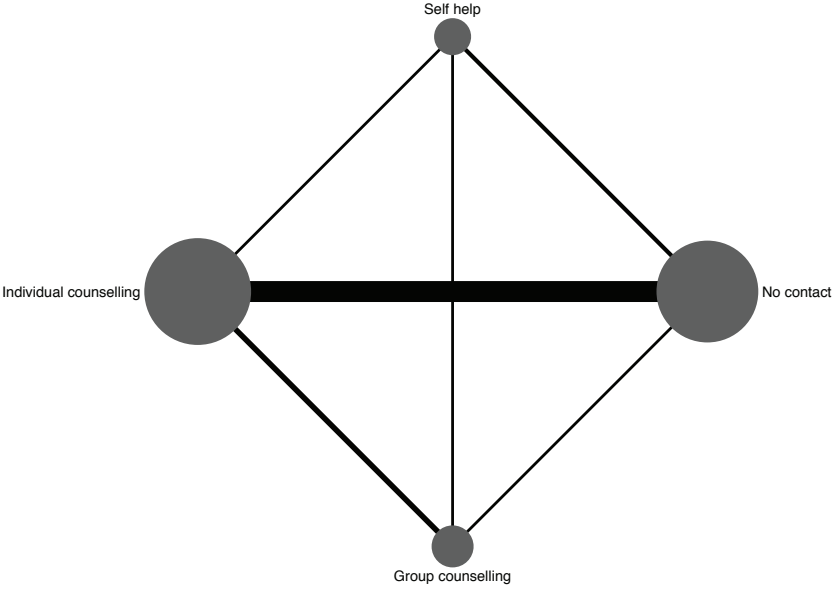

Figure 2. Network map for smoking data

Note that the treatment names are used when possible. They are abbreviated in the network table; this can be improved by using the `cellwidth()` option. The output shows that each pair of treatments is directly compared but that much of the data compares treatments A and C.

Next, we fit the consistency model:

```
. network meta consistency
Command is: mvmeta _y _S, bscovariance(exch 0.5) longparm suppress(uv mm)
> vars(_y_B _y_C _y_D)
Note: using method reml
Note: using variables _y_B _y_C _y_D
Note: 24 observations on 3 variables
Note: variance-covariance matrix is proportional to .5*I(3)+.5*J(3,3,1)

initial:       log likelihood = -60.906947
rescale:       log likelihood = -60.906947
rescale eq:    log likelihood = -60.694018
Iteration 0:   log likelihood = -60.694018
Iteration 1:   log likelihood = -59.279414
Iteration 2:   log likelihood = -59.252153
Iteration 3:   log likelihood = -59.252035
Iteration 4:   log likelihood = -59.252035

Multivariate meta-analysis
Variance-covariance matrix = proportional .5*I(3)+.5*J(3,3,1)
Method = reml                                   Number of dimensions    =     3
Restricted log likelihood = -59.252035          Number of observations  =    24
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
_y_B						
_cons	.3984173	.3310951	1.20	0.229	-.2505171	1.047352
_y_C						
_cons	.7023359	.1991037	3.53	0.000	.3120999	1.092572
_y_D						
_cons	.8658433	.3762801	2.30	0.021	.1283477	1.603339

```
Estimated between-studies SDs and correlation matrix:
             SD       _y_B       _y_C       _y_D
_y_B  .67445374          1          .          .
_y_C  .67445374         .5          1          .
_y_D  .67445374         .5         .5          1
mvmeta command stored as F9
```

Note that treatment codes, not names, are used. The `mvmeta` model used is displayed (and stored in *F9*) so that the user can modify it if desired. The estimated log odds-ratio for intervention B compared with A is 0.398, etc. We use these results to find the probabilities that each treatment is the best (that is, has the highest odds) under the consistency model and to plot the rankogram (Salanti, Ades, and Ioannidis 2011), shown in figure 3.

```
. network rank max, line cumulative xlabel(1/4) seed(37195)
> tabdispoptions(cellwidth(15))
Command is: mvmeta, noest pbest(max  in 1, zero id(study) line cumulative
> xlabel(1/4) seed(37195) tabdispoptions(cellwidth(15)) stripprefix(_y_)
> zeroname(A) rename(A = No contact, B = Self help, C = Individual counselling,
> D = Group counselling))
Option line specified -> option all assumed
Estimated probabilities (%) of each treatment being the best (and other ranks)
- assuming the maximum parameter is the best
- using 1000 draws
- allowing for parameter uncertainty
```

study and Rank	Treatment: No contact	Self help	Individual coun	Group counselli
1				
Best	0.0	6.0	31.4	62.6
2nd	0.4	18.7	54.9	26.0
3rd	13.6	62.2	13.7	10.5
Worst	86.0	13.1	0.0	0.9

```
mvmeta command is stored in F9
```

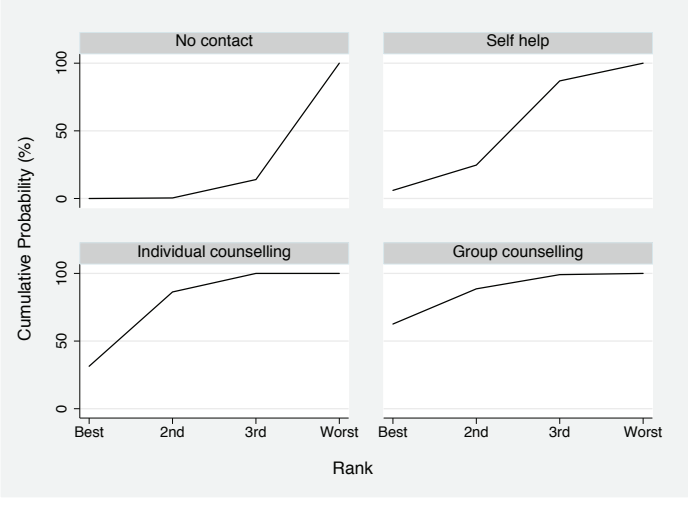

Figure 3. Rankogram for smoking data

The table and graph show, for example, that group counselling has a 62.6% probability of being the best treatment and about a 90% probability of being one of the two best treatments.

We now fit the inconsistency model:

```
. network meta inconsistency
Command is: mvmeta _y _S, bscovariance(exch 0.5) longparm suppress(uv mm)
> eq(_y_C: des_ACD des_BC des_BCD, _y_D: des_AD des_BCD des_BD des_CD)
> vars(_y_B _y_C _y_D)
Note: using method reml
Note: regressing _y_B on (nothing)
Note: regressing _y_C on des_ACD des_BC des_BCD
Note: regressing _y_D on des_AD des_BCD des_BD des_CD
Note: 24 observations on 3 variables
Note: variance-covariance matrix is proportional to .5*I(3)+.5*J(3,3,1)

initial:       log likelihood = -50.816796
rescale:       log likelihood = -50.816796
rescale eq:    log likelihood = -50.816796
Iteration 0:   log likelihood = -50.816796
Iteration 1:   log likelihood = -50.089407
Iteration 2:   log likelihood = -50.088702
Iteration 3:   log likelihood = -50.088702

Multivariate meta-analysis
Variance-covariance matrix = proportional .5*I(3)+.5*J(3,3,1)
Method = reml                                   Number of dimensions    =     3
Restricted log likelihood = -50.088702          Number of observations  =    24
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
_y_B						
_cons	.3299969	.4675425	0.71	0.480	-.5863696	1.246363
_y_C						
des_ACD	.3468324	.8821287	0.39	0.694	-1.382108	2.075773
des_BC	-.5261706	1.004485	-0.52	0.600	-2.494926	1.442584
des_BCD	-.3732418	1.013837	-0.37	0.713	-2.360325	1.613841
_cons	.7044606	.2347922	3.00	0.003	.2442763	1.164645
_y_D						
des_AD	3.393989	1.889991	1.80	0.073	-.3103244	7.098303
des_BCD	.4267799	1.3027	0.33	0.743	-2.126466	2.980026
des_BD	1.244879	1.323322	0.94	0.347	-1.348784	3.838543
des_CD	.8178211	1.123139	0.73	0.467	-1.383491	3.019133
_cons	.1285276	.8825026	0.15	0.884	-1.601146	1.858201

```
Estimated between-studies SDs and correlation matrix:
             SD       _y_B       _y_C       _y_D
_y_B  .74313772          1          .          .
_y_C  .74313772         .5          1          .
_y_D  .74313772         .5         .5          1

Testing for inconsistency:
 ( 1)  [_y_C]des_ACD = 0
 ( 2)  [_y_D]des_AD = 0
 ( 3)  [_y_C]des_BC = 0
 ( 4)  [_y_C]des_BCD = 0
 ( 5)  [_y_D]des_BCD = 0
 ( 6)  [_y_D]des_BD = 0
 ( 7)  [_y_D]des_CD = 0

           chi2(  7) =    5.11
         Prob > chi2 =    0.6464
mvmeta command stored as F9; test command stored as F8
```

The global test for inconsistency gives a p-value of 0.65, giving no evidence of inconsistency.

To see that the results are the same using the standard format, we convert the format and repeat the consistency analysis:

```
. network convert standard
Converting augmented to standard ...
. network meta consistency
Command is: mvmeta _y _S, bscovariance(exch 0.5) commonparm noconstant
> suppress(uv mm) eq(_y_1: _trtdiff1_B _trtdiff1_C _trtdiff1_D, _y_2:
> _trtdiff2_B _trtdiff2_C _trtdiff2_D) vars(_y_1 _y_2)
Note: using method reml
Note: regressing _y_1 on _trtdiff1_B _trtdiff1_C _trtdiff1_D
Note: regressing _y_2 on _trtdiff2_B _trtdiff2_C _trtdiff2_D
Note: 24 observations on 2 variables
Note: variance-covariance matrix is proportional to .5*I(2)+.5*J(2,2,1)
initial:       log likelihood = -34.471727
rescale:       log likelihood = -34.471727
rescale eq:    log likelihood = -34.259312
Iteration 0:   log likelihood = -34.259312
Iteration 1:   log likelihood = -32.844434
Iteration 2:   log likelihood = -32.817174
Iteration 3:   log likelihood = -32.817057
Iteration 4:   log likelihood = -32.817057
Multivariate meta-analysis
Variance-covariance matrix = proportional .5*I(2)+.5*J(2,2,1)
Method = reml                                  Number of dimensions    =     2
Restricted log likelihood = -32.817057         Number of observations  =    24
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
_y_1						
_trtdiff1_B	.3984487	.3311044	1.20	0.229	-.250504	1.047401
_trtdiff1_C	.7023556	.1991095	3.53	0.000	.3121081	1.092603
_trtdiff1_D	.8658902	.3762928	2.30	0.021	.1283699	1.603411

```
The above coefficients also apply to the following equations:
    _y_2: _trtdiff2_B _trtdiff2_C _trtdiff2_D
Estimated between-studies SDs and correlation matrix:
             SD       _y_1       _y_2
_y_1  .67446638          1          .
_y_2  .67446638         .5          1
mvmeta command stored as F9
```

Although the parameterization is different, the numerical results are almost identical. Differences arise in the fifth or sixth decimal place because of the tiny approximation introduced by augmentation.

Having fit both consistency and inconsistency models, we produce a forest plot (shown in figure 4):

```
. network forest, msize(*0.15) diamond name(smoke_forest, replace) eform
> xlabel(0.1 1 10 100)
group(design) assumed
```

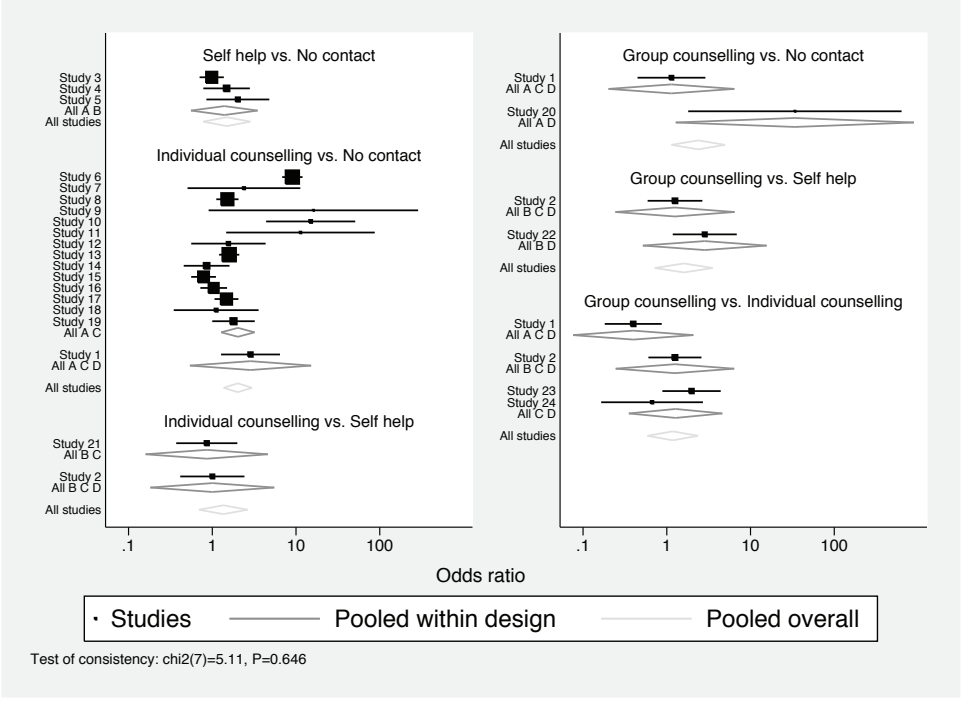

Figure 4. Forest plot for smoking network

The forest plot shows the individual study results, grouped by treatment contrast and design. It is clear that there is substantial heterogeneity between studies of C ("Individual counselling") versus A ("No contact"). Pooled results within design (from the inconsistency model) are shown as diamonds. Where there is only one study of a given design—for example, study 2 is the only "B C D" study at the bottom left—the point estimate pooled within design is the same as the single study's result, but the confidence interval is wider because the heterogeneity is assumed to be the same as in the other studies (here, estimated primarily from the "A C" studies). Overall pooled results are also shown as diamonds. The similarity of the "Pooled within design" and "Pooled overall" results again supports the consistency model.

We next explore inconsistency by side-splitting. We first split the side A–C, using the original method of Dias et al. (2010) for which the results depend on whether we split A–C or C–A (because A and C are contained in a three-arm study),

```
. network sidesplit A C, nosymmetric
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
direct	.7482566	.2088504	3.58	0.000	.3389174	1.157596
indirect	.317684	.5560699	0.57	0.568	-.7721931	1.407561
difference	.4305726	.5769152	0.75	0.455	-.7001604	1.561306

```
. network sidesplit C A, nosymmetric
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
direct	-.6840217	.21289	-3.21	0.001	-1.101278	-.2667649
indirect	-.937374	.636093	-1.47	0.141	-2.184093	.3093454
difference	.2533524	.6619598	0.38	0.702	-1.044065	1.55077

and using the new method for which the results are the same (apart from a change of sign) for splitting A–C or C–A:

```
. network sidesplit A C
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
direct	.7215198	.2132355	3.38	0.001	.303586	1.139454
indirect	.5736366	.6371481	0.90	0.368	-.6751508	1.822424
difference	.1478833	.6675922	0.22	0.825	-1.160573	1.45634

```
. network sidesplit C A
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
direct	-.7215198	.2132355	-3.38	0.001	-1.139454	-.3035859
indirect	-.5736366	.6371482	-0.90	0.368	-1.822424	.6751509
difference	-.1478833	.6675923	-0.22	0.825	-1.45634	1.160574

Finally, we split each node in turn—this can be slow because each split involves fitting a model:

```
. network sidesplit all
```

Side	Direct		Indirect		Difference		
	Coef.	Std. Err.	Coef.	Std. Err.	Coef.	Std. Err.	P>|z|
A B	.3266968	.4431874	.5101193	.5274799	-.1834225	.6881715	0.790
A C	.7215198	.2132355	.5736366	.6371481	.1478833	.6675922	0.825
A D	.8201395	.7576194	.8893868	.4420005	-.0692474	.8743754	0.937
B C	-.0756829	.5752542	.5163241	.4300301	-.592007	.7180019	0.410
B D	.6231378	.5693373	.2974687	.604145	.3256691	.8305243	0.695
C D	-.075668	.411104	.8722707	.70663	-.9479387	.8161917	0.245

These results again support consistency.

4.2 Thrombolytics network

The thrombolytics network (Lu and Ades 2006) includes eight treatments. It is more typical than the smoking network in that many possible pairs of treatments are not directly compared. We use these data, with just treatment codes, to demonstrate `network map`. We first set up the data:

```
. use thromb, clear
(Thrombolytics network meta-analysis from Lu & Ades (2006), corrected)

. network setup r n, studyvar(study) trtvar(treat)
Treatments used
   A (reference):                      A
   B:                                  B
   C:                                  C
   D:                                  D
   E:                                  E
   F:                                  F
   G:                                  G
   H:                                  H

Measure                                Log odds ratio

Studies
   ID variable:                        study
   Number used:                        28
   IDs with zero cells:                [none]
   IDs with augmented reference arm:   17 18 19 20 21 22 23 24 25 26 27 28
   - observations added:               0.001
   - mean in augmented observations:   study-specific mean

Network information
   Components:                         1 (connected)
   D.f. for inconsistency:             8
   D.f. for heterogeneity:             15

Current data
   Data format:                        augmented
   Design variable:                    _design
   Estimate variables:                 _y*
   Variance variables:                 _S*
   Command to list the data:           list study _y* _S*, noo sepby(_design)
```

The network map (figure 5) follows a standard format, but there are many line crossings that obscure the structure.

```
. network map
Graph command stored in F9
```

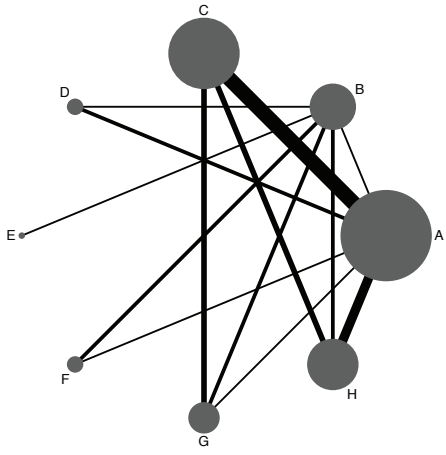

Figure 5. Network map for thrombolytics data: Default

We try improving the graph, using the `improve` option to repeatedly switch treatment locations to minimize the number of line crossings. We first base the map on the circle (see figure 6):

```
. network map, circle improve
Improving locations ...
loop 1 score 24= 21 19 17 16=.. 11== 7==....=... 4....=.=..... 3=......=.=
> ........==......==
loop 2 score 3=........=..=.....=........=.=...=..=......=.=........==......==
Stopping because loop 2 gave no improvement
Evaluating optimal locations ...
  1-3 and 2-8: score 1
  1-6 and 2-4: score 1
  1-7 and 2-8: score 1
Graph command stored in F9
```

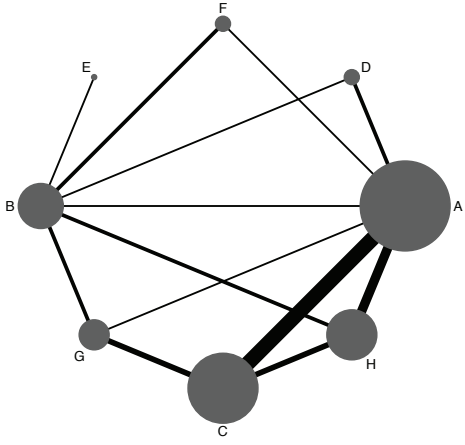

Figure 6. Network map for thrombolytics data: Circle with `improve` option

The numbers in the output count the line crossings (with co-incident lines scored as 10 crossings); "=" indicates a switch of treatment locations that doesn't improve the score (and is done); and "." indicates a switch of treatment locations that would make the score worse (and so is not done). The final output shows, for example, that lines 1–3 (A–C) and 2–8 (B–H) cross.

To get a map with no crossings, we next use the `improve` option starting with a triangular grid of side 5, which has 23 locations and hence 15 gaps (see figure 7):

```
. network map, triangular(5) improve
Improving locations ...
loop 1 score 117==. 106 103 58 55== 53.= 42....== 41... 38=...........=....=...
> 36.= 35.... 28.. 17.............. 15=.=... 5.========.===....=....= 4.==..=====
> .....===.....=.==.====.........==.........=..=........==.........=..=....=....
> .===============..=.....===============. 3..=...===============....=...=======
> =========.=...=.===============...==...===============....=== 1===============
> ....==.================....==..===============....==..===============......=.=
> ==============......=.================....=...===============....=...==========
> =====.....=..===============
loop 2 score 1=........=.=....=.......=.................=....=......=.....==.=.
> .==...=== 0
Stopping after achieving score of 0
Evaluating optimal locations ...
Graph command stored in F9
```

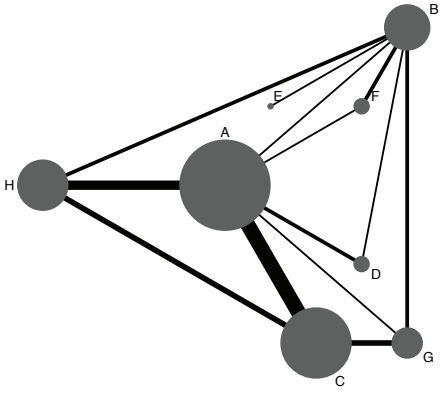

Figure 7. Network map for thrombolytics data: 5×5 triangular grid with `improve` option

That's good—there are no line crossings. Finally, we tidy the map by retrieving the treatment locations from matrix `_network_map_location`, moving treatments C, D, and E, and moving the label for treatment F (see figure 8):

```
. matrix loc2 = _network_map_location[1..8,.]
. * move C left to level of A
. matrix loc2[3,1]=loc2[1,1]
. * move D up to level of A
. matrix loc2[4,2]=loc2[1,2]
. * move E up to level of B
. matrix loc2[5,2]=loc2[2,2]
. * move F label to 10 o´clock
. matrix loc2[6,3]=10
. network map, loc(loc2)
Graph command stored in F9
```

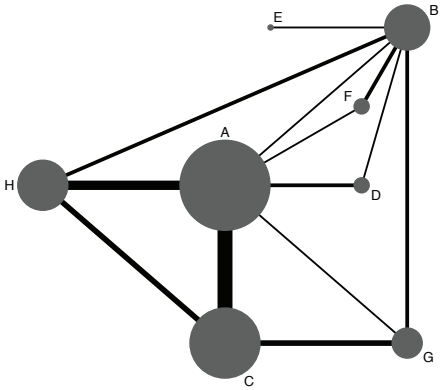

Figure 8. Network map for thrombolytics data: Figure 7 after moving treatments C, D, and E and the label for treatment F

Finally, we fit the consistency and inconsistency models (results not shown) and draw a forest plot (shown in figure 9). The most striking feature of this plot is that the two H versus B studies disagree strongly with the results of the consistency model, even though the overall test of inconsistency has a p-value of 0.38. A similar result is found using `network sidesplit all`.

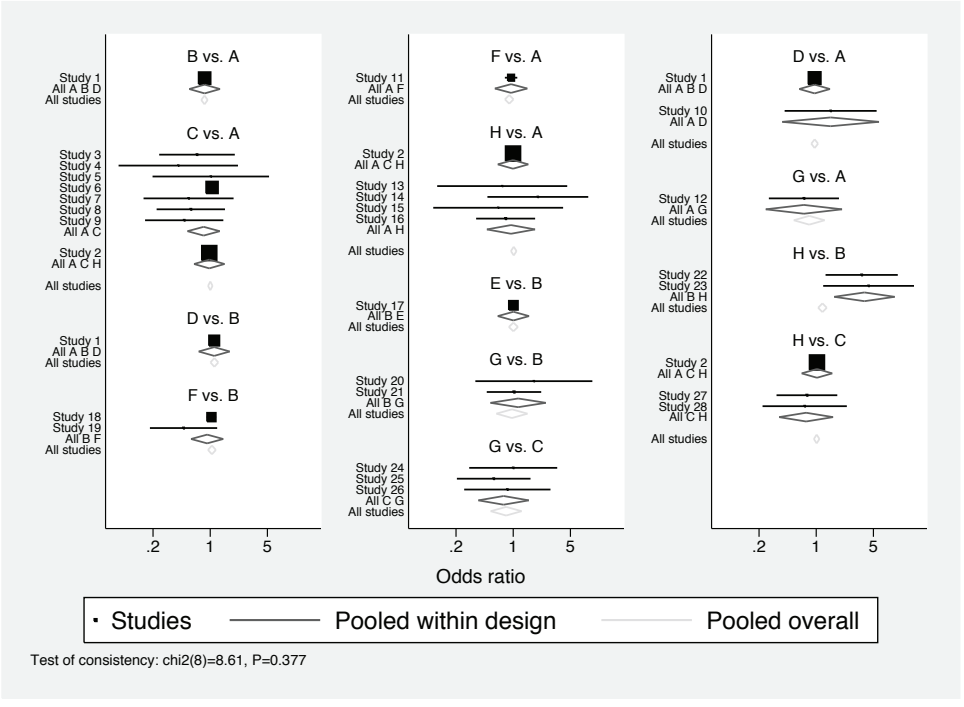

Figure 9. Forest plot for thrombolytics data

4.3 Quantitative data

I finally show how quantitative data may be used, using fictitious data on three treatments in long format:

```
. list, noobs clean
    study   trt   sbpmean   sbpsd   count
        1     P       150      15     100
        1     A       160      14     100
        1     B       162      16     100
        2     P       170      22      73
        2     A       175      20      77
        3     A       160      19      25
        3     B       154      19      25
```

Because the treatments are already encoded (P meaning placebo), we set them up using the `nocodes` option, and we set placebo as the reference:

```
. network setup sbpmean sbpsd count, studyvar(study) trtvar(trt) nocodes ref(P)
Treatments used
   A:                                  A
   B:                                  B
   P (reference):                      P
Measure                                Mean difference
   Standard deviation pooling:         off
Studies
   ID variable:                        study
   Number used:                        3
   IDs with augmented reference arm:   3
   - observations added:               0.001
   - mean in augmented observations:   study-specific mean
   - SD in augmented observations:     study-specific within-arms SD
Network information
   Components:                         1 (connected)
   D.f. for inconsistency:             2
   D.f. for heterogeneity:             0
Current data
   Data format:                        augmented
   Design variable:                    _design
   Estimate variables:                 _y*
   Variance variables:                 _S*
   Command to list the data:           list study _y* _S*, noo sepby(_design)
```

The data can now be tabulated and listed:

```
. network table
```

	Treatment and Statistic								
	A			B			P		
study	sbpmean	sbpsd	count	sbpmean	sbpsd	count	sbpmean	sbpsd	count
1	160	14	100	162	16	100	150	15	100
2	175	20	77				170	22	73
3	160	19	25	154	19	25			

```
. list study _*
```

	study	_design	_y_A	_S_A_A	_y_B	_S_B_B	_S_A_B
1.	1	A B P	10	4.21	12	4.81	2.25
2.	2	A P	5	11.824942	.	.	.
3.	3	A B	3	361014.42	-3	361014.42	360999.98

Further analyses would proceed as above, except that there are no degrees of freedom for heterogeneity, so a random-effects inconsistency model cannot be fit.

5 Discussion

The main limitation of the analysis methods presented here is that they are two-stage methods that rely on a normal approximation to the distribution of the estimated study-specific treatment effects. This approximation can be problematic with count data, especially with small counts. The main alternative is a Bayesian analysis that avoids this approximation. Comparisons of frequentist and Bayesian results suggest that frequentist results can be somewhat biased toward the null.

A second limitation of the analysis methods presented here is that the models are restricted to those that can be fit with `mvmeta`, whereas Bayesian models can allow for further levels of hierarchical modeling.

The key advantage of the methods presented here is their relative simplicity and speed, and hence the opportunity for the user to use a variety of analyses; for example, it would be easy to repeat the analyses omitting one or more studies. The data formats provided are sufficiently flexible to interface with other user-written software, and in particular, with the routines of Chaimani et al. (2013), which are available from http://www.mtm.uoi.gr.

The `network` suite is work in progress, and I would be delighted to hear suggestions for improvements and new features, or even for others to write new subcommands. Possible future features include the ability to create WinBUGS code and fit the model in a Bayesian way, a feature already available in R (van Valkenhoef et al. 2012b,a); the random inconsistency model (Jackson et al. 2014); and methods to explore inconsistency (White et al. 2012).

6 Acknowledgments

This work was supported by the Medical Research Council (Unit Programme number U105260558). I thank Anna Chaimani for tailoring `networkplot` to work with `network`, and I thank everyone who has made comments on this software for helping me to improve it.

7 References

Chaimani, A., J. P. T. Higgins, D. Mavridis, P. Spyridonos, and G. Salanti. 2013. Graphical tools for network meta-analysis in Stata. *PLOS ONE* 8: e76654.

Chaimani, A., and G. Salanti. 2015. Visualizing assumptions and results in network meta-analysis: The network graphs package. *Stata Journal* 15: 905–950.

Dias, S., N. J. Welton, D. M. Caldwell, and A. E. Ades. 2010. Checking consistency in mixed treatment comparison meta-analysis. *Statistics in Medicine* 29: 932–944.

Greenland, S., J. J. Schlesselman, and M. H. Criqui. 1986. The fallacy of employing standardized regression coefficients and correlations as measures of effect. *American Journal of Epidemiology* 123: 203–208.

Harbord, R. M., and J. P. T. Higgins. 2008. Meta-regression in Stata. *Stata Journal* 8: 493–519.

Hasselblad, V. 1998. Meta-analysis of multitreatment studies. *Medical Decision Making* 18: 37–43.

Hawkins, N., D. A. Scott, B. S. Woods, and N. Thatcher. 2009. No study left behind: A network meta-analysis in non–small-cell lung cancer demonstrating the importance of considering all relevant data. *Value in Health* 12: 996–1003.

Higgins, J. P. T., D. Jackson, J. K. Barrett, G. Lu, A. E. Ades, and I. R. White. 2012. Consistency and inconsistency in network meta-analysis: Concepts and models for multi-arm studies. *Research Synthesis Methods* 3: 98–110.

Higgins, J. P. T., S. G. Thompson, and D. J. Spiegelhalter. 2009. A re-evaluation of random-effects meta-analysis. *Journal of the Royal Statistical Society, Series A* 172: 137–159.

Higgins, J. P. T., and A. Whitehead. 1996. Borrowing strength from external trials in a meta-analysis. *Statistics in Medicine* 15: 2733–2749.

Jackson, D., J. K. Barrett, S. Rice, I. R. White, and J. P. T. Higgins. 2014. A design-by-treatment interaction model for network meta-analysis with random inconsistency effects. *Statistics in Medicine* 33: 3639–3654.

Lu, G., and A. E. Ades. 2004. Combination of direct and indirect evidence in mixed treatment comparisons. *Statistics in Medicine* 23: 3105–3124.

———. 2006. Assessing evidence inconsistency in mixed treatment comparisons. *Journal of the American Statistical Association* 101: 447–459.

———. 2009. Modeling between-trial variance structure in mixed treatment comparisons. *Biostatistics* 10: 792–805.

Lu, G., J. E. Brazier, and A. E. Ades. 2013. Mapping from disease-specific to generic health-related quality-of-life scales: A common factor model. *Value in Health* 16: 177–184.

Lu, G., D. Kounali, and A. E. Ades. 2014. Simultaneous multioutcome synthesis and mapping of treatment effects to a common scale. *Value in Health* 17: 280–287.

Lumley, T. 2002. Network meta-analysis for indirect treatment comparisons. *Statistics in Medicine* 21: 2313–2324.

Miladinovic, B., A. Chaimani, I. Hozo, and B. Djulbegovic. 2014. Indirect treatment comparison. *Stata Journal* 14: 76–86.

Mills, E. J., K. Thorlund, and J. P. A. Ioannidis. 2013. Demystifying trial networks and network meta-analysis. *British Medical Journal* 346: f2914.

Rücker, G., and G. Schwarzer. Forthcoming. Automated drawing of network plots in network meta-analysis. *Research Synthesis Methods*.

Salanti, G., A. E. Ades, and J. P. Ioannidis. 2011. Graphical methods and numerical summaries for presenting results from multiple-treatment meta-analysis: An overview and tutorial. *Journal of Clinical Epidemiology* 64: 163–171.

Salanti, G., J. P. T. Higgins, A. E. Ades, and J. P. A. Ioannidis. 2008. Evaluation of networks of randomized trials. *Statistical Methods in Medical Research* 17: 279–301.

van Valkenhoef, G., G. Lu, B. de Brock, H. Hillege, A. E. Ades, and N. J. Welton. 2012a. Automating network meta-analysis. *Research Synthesis Methods* 3: 285–299.

van Valkenhoef, G., T. Tervonen, B. de Brock, and H. Hillege. 2012b. Algorithmic parameterization of mixed treatment comparisons. *Statistics and Computing* 22: 1099–1111.

White, I. R. 2009. Multivariate random-effects meta-analysis. *Stata Journal* 9: 40–56.

———. 2011. Multivariate random-effects meta-regression: Updates to mvmeta. *Stata Journal* 11: 255–270.

———. 2015. Software Updates: st0156_2: Multivariate random-effects meta-analysis. *Stata Journal* 15: 1185–1186.

White, I. R., J. K. Barrett, D. Jackson, and J. P. T. Higgins. 2012. Consistency and inconsistency in network meta-analysis: Model estimation using multivariate meta-regression. *Research Synthesis Methods* 3: 111–125.

White, I. R., and J. Thomas. 2005. Standardized mean differences in individually randomized and cluster-randomized trials, with applications to meta-analysis. *Clinical Trials* 2: 141–151.

About the author

Ian White is a program leader at the MRC Biostatistics Unit in Cambridge, UK. His research interests focus on handling missing data, noncompliance and measurement error in the analysis of clinical trials, observational studies, and meta-analysis. He is the author of `mvmeta`.