MBNMAdose for dose-response Model-Based (Network) Meta-Analysis

Hugo Pedder

2019-07-03

Introduction

This vignette demonstrates how to use `MBNMAdose` to perform Model-Based Network Meta-Analysis (MBNMA) of studies with multiple doses of different agents by accounting for the dose-response relationship. This can connect disconnected networks via the dose-response relationship and the placebo response, improve precision of estimated effects and allow interpolation/extrapolation of predicted response based on the dose-response relationship.

Modelling the dose-response relationship also avoids the "lumping" of different doses of an agent which is often done in Network Meta-Analysis (NMA) and can introduce additional heterogeneity or inconsistency. All models and analyses are implemented in a Baysian framework, following an extension of the standrd NMA methodology presented by (Lu and Ades 2004) and are run in JAGS (JAGS Computer Program 2017). For full details of dose-response MBNMA methodology see Mawdsley et al. (2016). Throughout this vignette we refer to a **treatment** as a specific **dose** or a specific **agent**

This package has been developed alongside `MBNMAtime`, a package that allows users to perform time-course MBNMA to incorporate multiple time points within different studies. However, *they should not be loaded into R at the same time* as there are a number of functions with shared names that perform similar tasks yet are specific to dealing with either time-course *or* dose-response data.

Workflow within the package

Functions within `MBNMAdose` follow a clear pattern of use:

1. Load your data into the correct format using `mbnma.network()`
2. Analyse your data using `mbnma.run()`, or any of the available wrapper dose-response functions
3. Test for consistency at the treatment-level using functions like `NMA.nodesplit()` and `NMA.run()`
4. Examine model results using forest plots and treatment rankings
5. Use your model to predict responses using `predict()`

At each of these stages there are a number of informative plots that can be generated to help understand the data and to make decisions regaring model fitting.

Datasets Included in the Package

Triptans for migraine pain relief

`HF2PPITT` is from a systematic review of interventions for pain relief in migraine (Langford et al. 2016a). The outcome is binary, and represents (as aggregate data) the proportion of participants who were headache-free at 2 hours. Data are from patients who had had at least one migraine attack, who were not lost to follow-up, and who did not violate the trial protocol. The dataset includes 70 Randomised-Controlled Trials (RCTs), comparing 7 triptans with placebo. Doses are standardised as relative to a "common" dose, and in total there are 23 different treatments (combination of dose and agent). `HF2PPITT` is a data frame in long format (one row per arm and study), with the variables `studyID`, `AuthorYear`, `N`, `r`, `dose` and `agent`.

studyID	AuthorYear	N	r	dose	agent
1	Tfelt-Hansen P 2006	22	6	0	placebo

studyID	AuthorYear	N	r	dose	agent
1	Tfelt-Hansen P 2006	30	14	1	sumatriptan
2	Goadsby PJ 2007	467	213	1	almotriptan
2	Goadsby PJ 2007	472	229	1	zolmitriptan
3	Tuchman M2006	160	15	0	placebo
3	Tuchman M2006	174	48	1	zolmitriptan

Interventions for Serum Uric Acid (SUA) reduction in gout

`GoutSUA_2wkCFB` is from a systematic review of interventions for lowering Serum Uric Acid (SUA) concentration in patients with gout *[not published prevously]*. The outcome is continuous, and aggregate data responses correspond to the mean change from baseline in SUA in mg/dL at 2 weeks follow-up. The dataset includes 10 Randomised-Controlled Trials (RCTs), comparing 5 different agents, and placebo. Data for one agent (RDEA) arises from an RCT that is not placebo-controlled, and so is not connected to the network directly. In total there were 19 different treatments (combination of dose and agent). `GoutSUA_2wkCFB` is a data frame in long format (one row per arm and study), with the variables `studyID`, `y`, `se`, `agent` and `dose`.

	studyID	y	se	agent	dose
4	1102	-0.53	0.25	RDEA	100
5	1102	-1.37	0.18	RDEA	200
6	1102	-1.73	0.25	RDEA	400
53	2001	-6.82	0.06	Febu	240
54	2001	0.15	0.04	Plac	0
92	2003	-3.43	0.03	Allo	300

Interventions for pain relief in osteoarthritis

`osteopain_2wkabs` is from a systematic review of interventions for pain relief in osteoarthritis, used previously in Pedder et al. (2019). The outcome is continuous, and aggregate data responses correspond to the mean WOMAC pain score at 2 weeks follow-up. The dataset includes 18 Randomised-Controlled Trials (RCTs), comparing 8 different agents with placebo. In total there were 26 different treatments (combination of dose and agent). The active treatments can also be grouped into 3 different classes, within which they have similar mechanisms of action. `osteopain_2wkabs` is a data frame in long format (one row per arm and study), with the variables `studyID`, `agent`, `dose`, `class`, `y`, `se`, and `N`.

	studyID	agent	dose	class	y	se	N
13	1	Placebo	0	Placebo	6.26	0.23	60
14	1	Etoricoxib	10	Cox2Inhib	5.08	0.16	114
15	1	Etoricoxib	30	Cox2Inhib	4.42	0.17	102
16	1	Etoricoxib	5	Cox2Inhib	5.34	0.16	117
17	1	Etoricoxib	60	Cox2Inhib	3.62	0.17	112
18	1	Etoricoxib	90	Cox2Inhib	4.08	0.17	112

Alogliptin for lowering blood glucose concentration in type II diabetes

`alog_pcfb` is from a systematic review of Randomised-Controlled Trials (RCTs) comparing different doses of alogliptin with placebo (Langford et al. 2016b). The systematic review was simply performed and was intended to provide data to illustrate a statistical methodology rather than for clinical inference. Alogliptin is a treatment aimed at reducing blood glucose concentration in type II diabetes. The outcome is continuous, and aggregate data responses correspond to the mean change in HbA1c from baseline to follow-up in studies of at least 12 weeks follow-up. The dataset includes 14 Randomised-Controlled Trials (RCTs), comparing 5 different doses of alogliptin with placebo, leading to 6 different treatments (combination of dose and agent) within the network. `alog_pcfb` is a data frame in long format (one row per arm and study), with the variables `studyID`, `agent`, `dose`, `y`, `se`, and `N`.

studyID	agent	dose	y	se	N
NCT01263470	alogliptin	0.00	0.06	0.05	75
NCT01263470	alogliptin	6.25	-0.51	0.08	79
NCT01263470	alogliptin	12.50	-0.70	0.06	84
NCT01263470	alogliptin	25.00	-0.76	0.06	79
NCT01263470	alogliptin	50.00	-0.82	0.05	79
NCT00286455	alogliptin	0.00	-0.13	0.08	63

Inspecting the data

Before embarking on an analysis, the first step is to have a look at the raw data. Two features (network connectivity and dose-response relationship) are particularly important for MBNMA. For this we want to get our dataset into the right format for the package. We can do this using `mbnma.network()`.

```r
# Using the triptans dataset
network <- mbnma.network(HF2PPITT)
#> Values for `agent` with dose = 0 have been recoded to `Placebo`
#> agent is being recoded to enforce sequential numbering and allow inclusion of `Placebo`
print(network)
#> description :
#> [1] "Network"
#> 
#> data.ab :
#>   studyID agent dose treatment narm arm   r   N          AuthorYear
#> 1       1     1    0         1    2   1   6  22 Tfelt-Hansen P 2006
#> 2       1     3    1         2    2   2  14  30 Tfelt-Hansen P 2006
#> 3       2     5    1         3    2   1 213 467     Goadsby PJ 2007
#> 4       2     6    1         4    2   2 229 472     Goadsby PJ 2007
#> 5       3     1    0         1    2   1  15 160       Tuchman M2006
#> 6       3     6    1         4    2   2  48 174       Tuchman M2006
#>  [ reached 'max' / getOption("max.print") -- omitted 176 rows ]
#> 
#> agents :
#> [1] "Placebo"      "eletriptan"   "sumatriptan"  "frovatriptan"
#> [5] "almotriptan"  "zolmitriptan" "naratriptan"  "rizatriptan" 
#> 
#> treatments :
#>  [1] "Placebo_0"         "eletriptan_0.5"   "eletriptan_1"    
#>  [4] "eletriptan_2"      "sumatriptan_0.5"  "sumatriptan_1"   
#>  [7] "sumatriptan_1.7"  "sumatriptan_2"    "frovatriptan_1"  
#> [10] "frovatriptan_2"   "almotriptan_0.5"  "almotriptan_1"   
#> [13] "almotriptan_2"    "zolmitriptan_0.4" "zolmitriptan_1"  
#> [16] "zolmitriptan_2"   "zolmitriptan_4"   "zolmitriptan_10" 
#> [19] "naratriptan_1"    "naratriptan_2"    "rizatriptan_0.25"
#> [22] "rizatriptan_0.5"  "rizatriptan_1"
```

This takes a dataset with the columns:

- `studyID` Study identifiers
- `agent` Agent identifiers (can be character, factor or numeric)

- `dose` Numeric data indicating the dose of the given agent within the study arm
- `class` An optional column indicating a particular class code. Agents with the same name/identifier must also have the same class code.

Depending on the type of data (and the likelihood) the following columns are required:

- Normal likelihood
  - `y` Numeric data indicating the mean response for a given study arm
  - `se` Numeric data incicating the standard error for a given study arm
- Binomial likelihood
  - `r` Numeric data indicating the number of responders in a given study arm
  - `N` Numeric data indicating the total number of participants in a given study arm
- Poisson likelihood
  - `r` Numeric data indicating the number of events in a given study arm
  - `E` Numeric data indicating the total exposure time in a given study arm
- `se` Numeric data incicating the standard error for a given observation
- `treatment` Treatment identifiers (can be numeric, factor or character)

Additional columns can be included in the dataset. These will simply be added to the `mbnma.network` object, though will not affect the classification of the data.

It then performs the following checks on the data:

- The dataset has the required column names
- There are no missing values
- All doses are positive
- All SE, r, N and E are positive
- Class labels are consistent within each agent
- Studies have at least two arms
- Studies do not only compare the same agent at the same dose

Finally it converts the data frame into an object of `class("mbnma.network")`, which contains indices for study arms, numeric variables for treatments, agents and classes, and stores a vector of treatment, agent and class names as an element within the object. By convention, agents are numbered alphabetically, though if the original data for agents is provided as a factor then the factor codes will be used. This then contains all the necessary information for subsequent `MBNMAdose` functions.

Network connectivity

Looking at how the evidence in the network is connected and identifying which studies compare which treatments/agents helps to understand which effects can be estimated and what information will be helping to inform those estimates. In particular, the complexity of dose-response relationships that can be estimated is dependent on the number of doses of each agent available in the dataset.

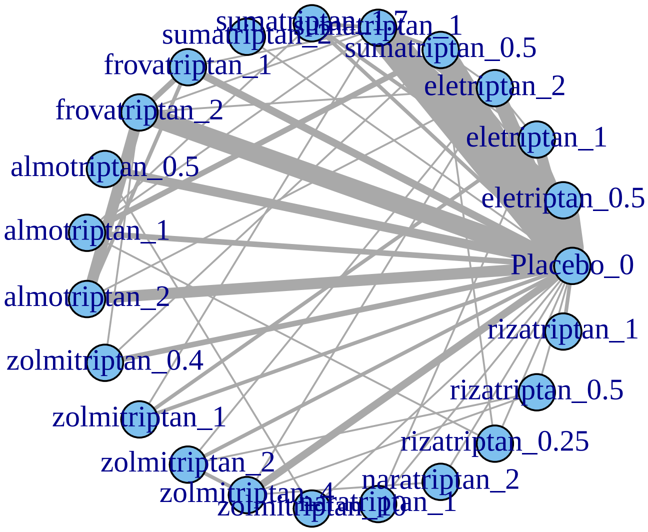
Network plots can be plotted which shows which treatments/agents have been compared in head-to-head trials. Typically the thickness of connecting lines ("edges") is proportional to the number of studies that make a particular comparison and the size of treatment nodes ("vertices") is proportional to the total number of patients in the network who were randomised to a given treatment/agent (provided `N` is included as a variable in the original dataset for `mbnma.network()`).

In `MBNMAdose` these plots are generated using `igraph`, and can be plotted by calling `plot()`. The generated plots are objects of `class("igraph")` meaning that, in addition to the options specified in `plot()`, various `igraph` functions can subsequently be used to make more detailed edits to them.

Within these network plots, vertices are automatically alligned in a circle (as the default) and can be tidied by shifting the label distance away from the nodes.

```
# Prepare data using the triptans dataset
tripnet <- mbnma.network(HF2PPITT)
```

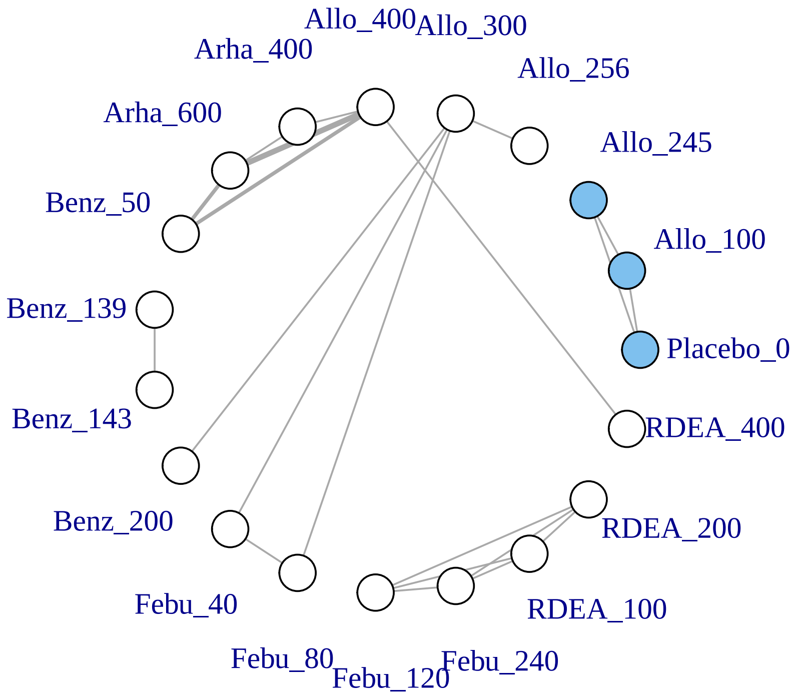
```
#> Values for `agent` with dose = 0 have been recoded to `Placebo`
#> agent is being recoded to enforce sequential numbering and allow inclusion of `Placebo`

# Draw network plot
plot(tripnet)
```



If some vertices are not connected to the network reference treatment through any pathway of head-to-head evidence, a warning will be given. The nodes that are coloured white represent these disconnected vertices.

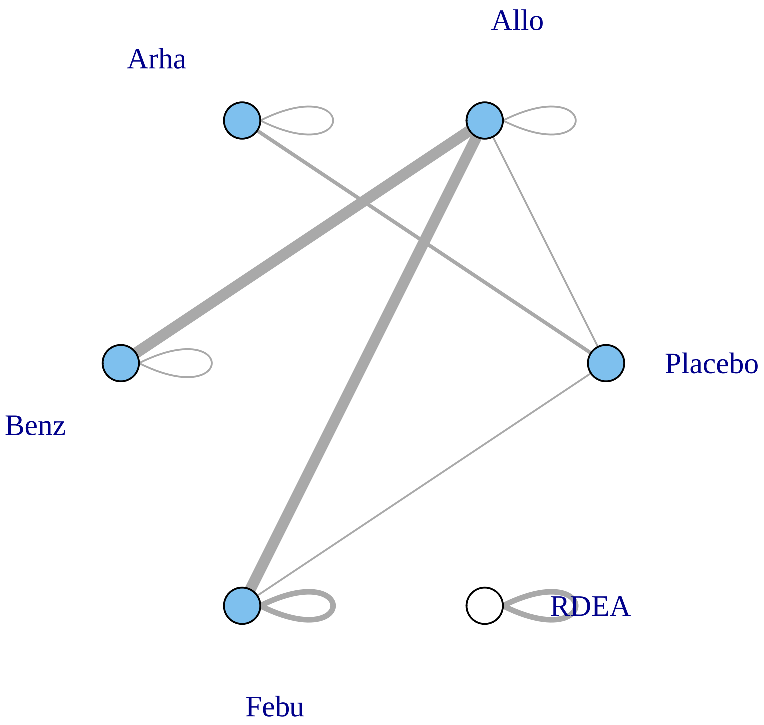
```
# Prepare data using the gout dataset
goutnet <- mbnma.network(GoutSUA_2wkCFB)
```

```
plot(goutnet, label.distance = 5)
#> Warning in check.network(g): The following treatments/agents are not connected to the network refere
#> Allo_256
#> Allo_300
#> Allo_400
#> Arha_400
#> Arha_600
#> Benz_50
#> Benz_139
#> Benz_143
#> Benz_200
#> Febu_40
#> Febu_80
#> Febu_120
#> Febu_240
#> RDEA_100
#> RDEA_200
#> RDEA_400
```



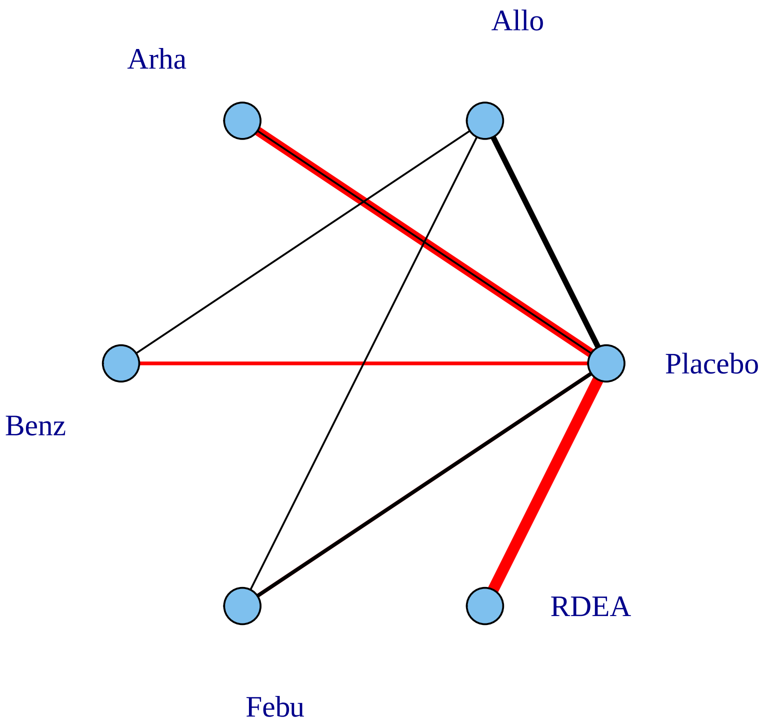
However, whilst at the treatment-level (specific dose of a specific agent), many of these vertices are disconnected, at the agent level they are connected (via different doses of the same agent), meaning that *via the dose-response relationship* it is possible to estimate results.

```
# Plot at the agent-level
plot(goutnet, level="agent", label.distance = 6)
#> Warning in check.network(g): The following treatments/agents are not connected to the network refere
#> RDEA
```



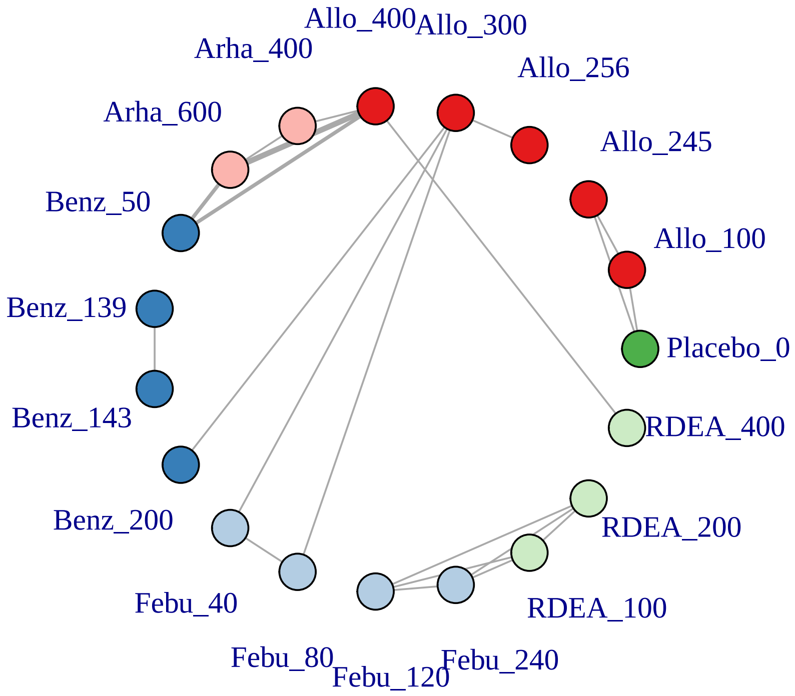
One agent (RDEA) is still not connected to the network, but `MBNMAdose` allows agents to connect via a placebo response *even if they do not include placebo in a head-to-head trial*. The connectivity of this will depend on the number of doses compared within an agent, and on the complexity of the dose-response relationship (Mawdsley et al. 2016).

```
# Plot connections to placebo via a two-parameter dose-response function (e.g. Emax)
plot(goutnet, level="agent", doselink = 2, remove.loops = TRUE, label.distance = 6)
#> Dose-response connections to placebo plotted based on a dose-response
#>                    function with 1 degrees of freedom
```



It is also possible to plot a network at the treatment level but to colour the doses by the agent that they belong to.

```
# Colour vertices by agent
plot(goutnet, v.color = "agent", label.distance = 5)
#> Warning in check.network(g): The following treatments/agents are not connected to the network refere
#> Allo_256
#> Allo_300
#> Allo_400
#> Arha_400
#> Arha_600
#> Benz_50
#> Benz_139
#> Benz_143
#> Benz_200
#> Febu_40
#> Febu_80
#> Febu_120
#> Febu_240
#> RDEA_100
#> RDEA_200
#> RDEA_400
```



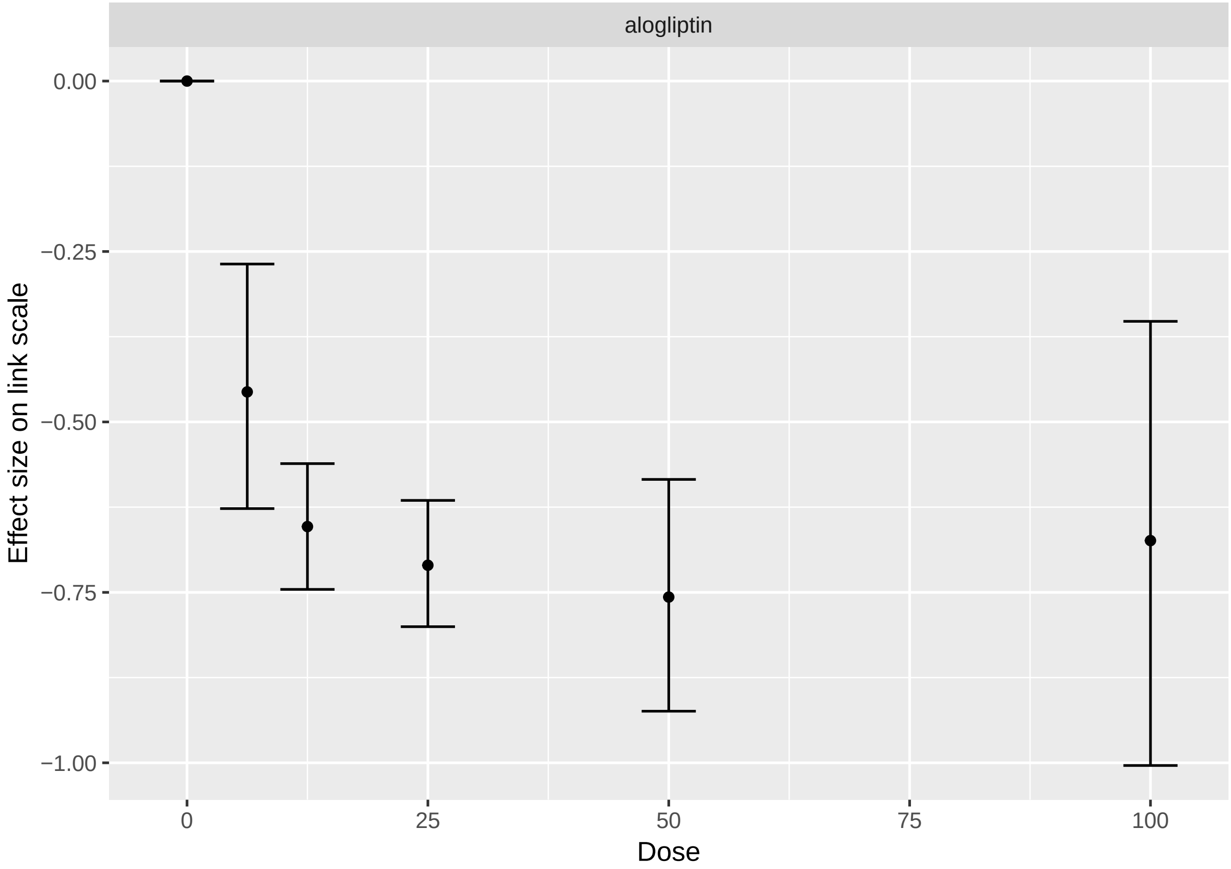
Several further options exist to allow for inclusion of disconnected treatments, such as assuming some sort of common effect among agents within the same class. This is discussed in more detail later in the vignette.

Examining the dose-response relationship

In order to consider which functional forms may be appropriate for modelling the dose-response relationship, it is useful to look at results from a "split" network meta-analysis (NMA), in which each dose of an agent is considered as separate and unrelated (i.e. we are not assuming any dose-response relationship). The `NMA.run()` function performs a simple NMA, and by default it drops studies that are disconnected at the treatment-level (since estimates for these will be very uncertain if included).

```
# Run a random effect split NMA using the alogliptin dataset
alognet <- mbnma.network(alog_pcfb)
nma.alog <- nma.run(alognet, method="random")
```

```
print(nma.alog)
#> $jagsresult
#> Inference for Bugs model at "C:\Users\hp17602\AppData\Local\Temp\RtmpwvhQlZ\file4c824106817", fit us
#>  3 chains, each with 10000 iterations (first 5000 discarded), n.thin = 5
#>  n.sims = 3000 iterations saved
#>           mu.vect sd.vect   2.5%    25%    50%    75%  97.5%
#> d[1]        0.000   0.000  0.000  0.000  0.000  0.000  0.000
#> d[2]       -0.453   0.089 -0.627 -0.511 -0.456 -0.396 -0.268
#> d[3]       -0.654   0.047 -0.746 -0.685 -0.653 -0.623 -0.561
#> d[4]       -0.709   0.046 -0.800 -0.739 -0.710 -0.678 -0.615
#> d[5]       -0.757   0.086 -0.924 -0.815 -0.757 -0.702 -0.584
#> d[6]        -0.677   0.168   -1.004   -0.788   -0.674   -0.566   -0.352
#> sd           0.124   0.028    0.075    0.104    0.122    0.141    0.184
#> totresdev   46.668  10.044   29.151   39.401   46.049   53.077   68.468
#> deviance  -124.646  10.044 -142.163 -131.913 -125.265 -118.237 -102.846
#>            Rhat n.eff
#> d[1]      1.000     1
#> d[2]      1.004   640
#> d[3]      1.003   690
#> d[4]      1.001  3000
#> d[5]      1.002  1300
#> d[6]      1.001  3000
#> sd        1.001  3000
#> totresdev 1.001  3000
#> deviance  1.001  3000
#>
#> For each parameter, n.eff is a crude measure of effective sample size,
#> and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
#>
#> DIC info (using the rule, pD = var(deviance)/2)
#> pD = 50.5 and DIC = -74.2
#> DIC is an estimate of expected predictive error (lower deviance is better).
#>
#> $trt.labs
#> [1] "Placebo_0"       "alogliptin_6.25" "alogliptin_12.5" "alogliptin_25"
#> [5] "alogliptin_50"   "alogliptin_100"
#>
#> attr(,"class")
#> [1] "nma"

# Draw plot of NMA estimates plotted by dose
plot(nma.alog)
```



In the alogliptin dataset there appears to be a dose-response relationship, and it also appears to be non-linear.

One additional use of `NMA.run()` is that is can be used after fitting an MBNMA to ensure that a fitting a dose-response function is not leading to poorer model fit than when conducting a conventional NMA. Comparing the total residual deviance between NMA and MBNMA models is useful to identify if introducing a dose-response relationship is leading to poorer model fit. However, it is important to note that if treatments are disconnected in the NMA and have been dropped (`drop.discon=TRUE`), there will be fewer observations present in the dataset, which will subsequently lead to lower pD and lower residual deviance, meaning that model fit statistics from NMA and MBNMA may not be directly comparable.

Analysis using `mbnma.run()`

MBNMA is performed in `MBNMAdose` by applying `mbnma.run()`. A `"mbnma.network"` object must be provided as the data for `mbnma.run()`. The key arguments within `mbnma.run()` involve specifying the functional form used to model the dose-response, and the dose-response parameters that comprise that functional form.

Dose-response functions

Several different functional forms are implemented within `MBNMAdose`, that allow a variety of parameterizations and dose-response shapes. These are provided to the `fun` argument in `mbnma.run()`:

- `"linear"`
- `"exponential"`
- `"emax"` - Emax without a Hill parameter
- `"emax.hill"` - Emax with a Hill parameter
- `"nonparam.up"` - Non-parametric monotonically increasing dose-response
- `"nonparam.down"` - Non-parametric monotonically decreasing dose-response

- "user" - A function that can be explictly specified by the user within user.fun (see ?mbnma.run())

Dose-response parameters

In mbnma.run() it is possible to specify up to three different dose-response parameters, depending on the dose-response function used. These are named beta.1, beta.2, and beta.3, and their interpretation varies depending on the dose-response function used (see ?mbnma.run()).

For simplification and interpretability, both in the way in which dose-response parameters are defined and in how they are reported in the output, there are wrapper functions for mbnma.run() for each of the provided time-course functions. For example, mbnma.emax() is equivalent to mbnma.run(fun="emax"), but with a different naming of time-course parameters (emax instead of beta.1 and ed50 instead of beta.2) . A number of these will be shown in other examples in this vignette.

Dose-response parameters can be assigned different specifications which define the key parameters estimated by the model and are dependent on the assumptions made within the model. Three different specifications are available for each parameter:

- "rel" indicates that relative effects should be pooled for this dose-response parameter. This preserves randomisation within included studies and is likely to vary less between studies (only due to effect modification).
- "common" indicates that a single absolute value for this dose-response parameter should be estimated across the whole network *that does not vary by agent*. This is particularly useful for parameters expected to be constant (e.g. Hill parameters in mbnma.emax.hill()).
- "random" indicates that a single absolute value should be estimated separately for each agent, but that all the agent values vary randomly around a single mean absolute network effect. It is similar to "common" but makes slightly less strong assumptions.
- numeric() Assigned a numeric value - this is similar to assigning "common", but the single absolute value is assigned as a numeric value by the user, rather than estimated from the data.

In mbnma.run(), an additional argument, method, indicates what method to use for pooling relative effects and can take either the values "common", implying that all studies estimate the same true effect (akin to a "fixed effects" meta-analysis), or "random", implying that all studies estimate a separate true effect, but that each of these true effects vary randomly around a true mean effect. This approach allows for modelling of between-study heterogeneity.

If relative effects ("rel") are modelled on more than one dose-response parameter then by default, a correlation will be assumed between the dose-response parameters, which will typically improve estimation (provided the parameters are correlated…they usually are). This can be prevented by setting cor=FALSE.

Output

mbnma.run() returns an object of class c("mbnma", "rjags"). summary() provides posterior medians and 95% credible intervals for different parameters in the model, with some explanation of the way in which the model has been defined. print() can also be used to give summary statistics of the posterior distributions for monitored nodes in the JAGS model. Estimates are automatically reported for parameters of interest depending on the model specification (unless otherwise specified in parameters.to.save)

Nodes that are automatically monitored (if present in the model) have the following interpretation. They will have an additional suffix that relates to the name/number of the time-course paramter to which they correspond (e.g. d.et50 or beta.1):

- d The pooled effect for each agent for a given dose-response parameter. Will be estimated by the model if dose-respones parameters (beta.1, beta.2, beta.3) are set to "rel".
- sd (without a suffix) - the between-study SD (heterogeneity) for relative effects, reported if method="random".
- D The class effect for each class for a given dose-response parameter. Will be estimated by the model if specified in class.effect for a dose-response parameter set to "rel".

- sd.D The within-class SD for different agents within the same class. Will be estimated by the model if any dose-response parameter in class.effect is set to "random".
- beta The absolute value of a given dose-response parameter across the whole network (does not vary by agent/class). Will be estimated by the model if dose-response parameters (beta.1, beta.2, beta.3) are set to "common" or "random".
- sd (with a suffix) - the between-study SD (heterogeneity) for absolute dose-response parameters, reported if beta.1, beta.2 or beta.3 are set to "random"
- totresdev The residual deviance of the model
- deviance The deviance of the model

Model fit statistics for pD (effective number of parameters) and DIC (Deviance Information Criterion) are also reported, with an explanation as to how they have been calculated.

Examples

An example MBNMA of the triptans dataset using an Emax dose-response function and common treatment effects that pool relative effects on both Emax and ED50 parameters follows:

```
# Run an Emax dose-response MBNMA
mbnma <- mbnma.run(tripnet, fun="emax",
                   beta.1="rel", beta.2="rel", method="common")
#> `likelihood` not given by user - set to `binomial` based on data provided
#> `link` not given by user - set to `logit` based on assigned value for `likelihood`
#> Results for ED50 (`beta.2`) modelled on the exponential scale
```

```
summary(mbnma)
#> ========================================
#> Dose-response MBNMA
#> ========================================
#>
#> Dose-response function: emax
#>
#> #### beta.1 dose-response parameter results ####
#> Pooling: relative effects
#>
#> Parameter    Median (95%CrI)
#> ---------------------------------
#> d.1[2]   2.55 (2.29, 2.91)
#> d.1[3]   1.72 (1.5, 1.97)
#> d.1[4]   1.82 (1.32, 2.95)
#> d.1[5]   1.84 (1.41, 2.75)
#> d.1[6]   2.02 (1.63, 2.76)
#> d.1[7]   1.03 (0.503, 1.84)
#> d.1[8]   2.3 (1.86, 3.12)
#>
#>
#>
#> #### beta.2 dose-response parameter results ####
#> Parameter modelled on exponential scale to ensure it takes positive values on the natural scale
#> Pooling: relative effects
#>
#> Parameter    Median (95%CrI)
#> ---------------------------------
#> d.2[2]   -0.691 (-1.09, -0.324)
#> d.2[3]   -0.74 (-1.26, -0.207)
#> d.2[4]   -0.637 (-1.72, 0.457)
#> d.2[5]   -0.222 (-0.895, 0.55)
#> d.2[6]   -0.44 (-1.14, 0.242)
#> d.2[7]   -0.167 (-1.47, 0.835)
#> d.2[8]   -0.717 (-1.36, 0.00266)
#>
#>
#>
#> #### Pooling method ####
#>
#> Method: Common (fixed) effects estimated for relative effects
#>
#>
#>
#>
#> #### Model Fit Statistics ####
#>
#> Effective number of parameters:
#> pD (pV) calculated using the rule, pD = var(deviance)/2 = 76
#> Deviance = 1170
#> Residual deviance = 266
#> Deviance Information Criterion (DIC) = 1246
```

```
# An alternative would be to use an Emax wrapper for mbnma.run() which would give the same result but w
mbnma.emax(tripnet, emax="rel", ed50="rel", method="common")
```

In this case the `d.1`/`d.emax` parameters correspond to the effects of each agent for the dose-response parameter `beta.1`/`emax`, which corresponds (for this dose-reponse function) to the maximum response that can be achieved for a particular agent. The `d.2`/`d.ed50` parameters correspond to the effects for `beta.2`/`ed50`, which (in this case) corresponds to the dose at which 50% of the maximum response is achieved (results are given on the log scale for thsi parameter as it is constraned to be >0).

Instead of pooling relative effects and estimating a separate relative effect for each agent, a simpler dose-response model (that makes more assumptions) could be to estimate a single parameter across the whole network for ED50, but that still estimates relative effects for Emax. In this case we can run with random effects on Emax:

```
# Emax model with single parameter estimated for Emax
emax <- mbnma.emax(tripnet, emax="rel", ed50="common", method="random")
summary(emax)
```

In this case the `d.1`/`d.emax` parameters correspond to the effect of each agent for the dose-response parameter `beta.1`/`emax`, as previously. But now we have a `beta.ed50` parameter in the output (instead of `d.ed50`), which corresponds to the absolute value of ED50 (on the log scale) across the whole network. As we have modelled random relative effects, we also have estimated a parameter for `sd`, the between-study standard deviation.

However, although the total residual deviance (`totresdev`) is lower in this model, indicating a better fit, the effective number of parameters (pD) is much greater, and overall the DIC is higher, suggesting that the first model is a better compromise of fit and complexity.

Additional arguments for `mbnma.run()`

Several additional arguments can be given to `mbnma.run()` that require further explanation.

Class effects

Shared effects between agents within the network can be modelled using class effects. This requires assuming that different agents have some sort of common class effect, perhaps due to similar mechanisms of action. Advantages of this is that class effects can be used to connect agents that might otherwise be disconnected from the network, and they can also provide additional information on agents that might otherwise have insufficient data available to estimate a desired dose-response. The drawback is that this requires making additional assumptions regarding similarity of efficacy.

Class effects can only be applied to dose-response parameters which are modelled using relative effects (`"rel"`). In `mbnma.run()` they are supplied as a list, in which each element is named following the name of the corresponding dose-response parameter as defined in the function. The names will therefore differ when using wrapper functions for `mbnma.run()`. The class effect for each dose-response parameter can be either `"common"`, in which the effects for each agent within the same class are constrained to a common class effect, or `"random"`, in which the effects for each agent within the same class are assumed to be randomly distributed around a shared class mean.

When working with class effects in `MBNMAdose` a variable named `class` must be included in the original data frame provided to `mbnma.network()`.

```
# Using the osteoarthritis dataset (contains info on classes)
painnet <- mbnma.network(osteopain_2wkabs)
```

```
# Emax MBNMA with a single absolute parameter for Emax and random relative effects on ED50
# Common class effects on ED50
emax <- mbnma.emax(painnet, emax="common", ed50="rel", method="random",
                   class.effect=list(ed50="common"))
#> `likelihood` not given by user - set to `normal` based on data provided
#> `link` not given by user - set to `identity` based on assigned value for `likelihood`
#> Results for ED50 (`beta.2`) modelled on the exponential scale
#> Warning in write.cor(model, beta.1 = beta.1, beta.2 = beta.2, beta.3 =
#> beta.3, : Class effects cannot be modelled with correlation between time-
#> course relative effects - correlation will be ignored
```

```
summary(emax)
#> ========================================
#> Dose-response MBNMA
#> ========================================
#>
#> Dose-response function: emax
#>
#> #### emax dose-response parameter results ####
#> Pooling: absolute single parameter
#>
#> Parameter    Median (95%CrI)
#> ---------------------------------
#> beta.emax        -1.02 (-1.23, -0.826)
#>
#>
#>
#> #### Pooling method ####
#>
#> Method: Random effects estimated for relative effects
#>
#> Parameter                    Median (95%CrI)
#> -----------------------------------------------------------------------
#> Between-study SD for relative effects        0.421 (0.331, 0.546)
#>
```

```
#> #### Class effects ####
#>
#> Class effect on ed50: common
#>
#> Parameter    Median (95%CrI)
#> ---------------------------------
#> D.ed50[2]    -6.91 (-61.6, 2.08)
#> D.ed50[3]    -14.6 (-67.6, 5.43)
#> D.ed50[4]    7.43 (-24, 66.1)
#>
#> #### Model Fit Statistics ####
#>
#> Effective number of parameters:
#> pD (pV) calculated using the rule, pD = var(deviance)/2 = 83
#> Deviance = -90
#> Residual deviance = 77
#> Deviance Information Criterion (DIC) = -6
```

Mean class effects are given in the output as `D.ed50`/`D.1` parameters. These can be interpreted as the effect of each class for Emax parameters (`beta.1`). Note the number of `D.ed50` parameters is therefore equal to the number of classes defined in the dataset.

If we had specified that the class effects were `"random"`, each treatment effect for Emax (`beta.1`) would be assumed to be randomly distributed around its class mean with SD given in the output as `sd.D.ed50`/`sd.D.1`.

Correlation between dose-response parameters

`mbnma.run()` automatically models correlation between relative effects dose-response parameters (unless `cor=FALSE`). The correlation is modelled using a vague Wishart prior, but this can be made more informative by indicating the relative magnitude of scales of the parameters that are modelled using relative effects.

`var.scale` can be used for this - it takes a numeric vector the same length as the number of relative effect dose-response parameters, and the relative magnitude of the numbers indicates the relative magnitude of the scales. Each element of `var.scale` corresponds to the relevant dose-response parameter (i.e. `var.scale[1]` will correspond to `beta.1`)

For example, with the triptans dataset we might expect that values for Emax might be 4 times larger for ED50 (on the log scale):

```
# Define relative magnitudes of Emax and ED50
rel.size <- c(4,1)

mbnma.emax(tripnet, emax="rel", ed50="rel", method="random",
           var.scale=rel.size)
```

User-specified time-course function

If users want to write their own dose-response function rather than using one of the ones specified in `mbnma.run()` they can do this by specifying `fun = "user"` in the arguments. A string can then be provided to `user.fun`, which specifies a new time course in terms of `beta` parameters and `dose`. This allows a huge amount of addtional flexibility when defining the dose-response function.

The string assigned to `user.fun` needs to fulfill a few criteria to be valid: * `dose` must always be included in the function * At least one `beta` time-course parameter must be specified, up to a maximum of three. These are always named `beta.1`, `beta.2` and `beta.3`, and must be included sequentially (i.e. don't include `beta.3` if `beta.1` is not included) * Indices used by JAGS should *not* be added to `user.fun` (e.g. use `dose` rather than `dose[i,k]`) * Any mathematical/logical operators that can be implemented in JAGS can be added to the function

```r
# An example specifying a quadratic dose-response function
quad <- "(beta.1 * dose) + (beta.2 * (dose^2))"

mbnma.run(tripnet, fun="user", user.fun=quad,
          beta.1 = "rel", beta.2 = "rel")
```

Priors

Default vague priors for the model are as follows:

$$
\begin{aligned}
&d_{\phi,a} \sim N(0, 10000)\\
&beta_{\phi} \sim N(0, 10000)\\
&\sigma \sim N(0, 400) \text{ limited to } x \in [0, \infty]\\
&\sigma_{\phi} \sim N(0, 400) \text{ limited to } x \in [0, \infty]\\
&D_{\phi,c} \sim N(0, 1000)\\
&\sigma_{\phi} \sim N(0, 400) \text{ limited to } x \in [0, \infty]
\end{aligned}
$$

…where ϕ is an identifier for the dose-response parameter (e.g. 1 for Emax and 2 for ED50), a is an agent identifier and c is a class identifier

Users may wish to change these, perhaps in order to use more/less informative priors, but also because the default prior distributions in some models may lead to errors when compiling/updating models.

If the model fails during compilation/updating (i.e. due to a problem in JAGS), `mbnma.run()` will generate an error and return a list of arguments that `mbnma.run()` used to generate the model. Within this (as within a model that has run successfully), the priors used by the model (in JAGS syntax) are stored within `"model.arg"`:

```r
print(mbnma$model.arg$priors)
```

In this way a model can first be run with vague priors and then rerun with different priors, perhaps to allow successful computation, perhaps to provide more informative priors, or perhaps to run a sensitivity analysis with different priors. Increasing the precison of prior distributions only a little can also often improve convergence considerably.

To change priors within a model, a list of replacements can be provided to `priors` in `mbnma.run()`. The name of each element is the name of the parameter to change (without indices) and the value of the element is the JAGS distribution to use for the prior. This can include censoring or truncation if desired. Only the priors to be changed need to be specified - priors for parameters that aren't specified will take default values.

For example, if we want to use tighter priors for the half-normal SD parameters we could increase the precision:

```r
# Define replacement prior
new.priors <- list(
  sd = "dnorm(0, 1) T(0,)"
  )

# Run an MBNMA model with new priors
emax <- mbnma.emax(alognet, emax="rel", ed50="rel", method="random",
                   priors=new.priors)
```

Mathematical errors can be more likely for certain types of models. For dose-response relationships which include a dose-response parameter as an exponent (e.g. an exponential dose-response function), default prior distributions may result in numerical values that are too extreme for computation. One way to fix this problem is to use tighter priors for `d`'s. However, a contributing factor to this issue is that doses within a dataset can be very variable in magnitude, thus leading to excessively high/low exponents. A solution for this is is to standardise doses within an agent by some common value (i.e. the most commonly prescribed dose or the maximum dose investigated). This leads to all doses being relative to the standard dose, but can improve computation.

```
# Exponential model currently returns an error on gout dataset
exponetial <- mbnma.exponential(goutnet, lambda="rel")
#> Compiling model graph
#>    Resolving undeclared variables
#>    Allocating nodes
#> Graph information:
#>    Observed stochastic nodes: 27
#>    Unobserved stochastic nodes: 15
#>    Total graph size: 348
#>
#> Initializing model
#>
#>
  |
  |                                                    |   0%
  |
  |++                                                  |   4%
#> Error in update.jags(object, n.iter, ...): Error in node y[2,2]
#> Failure to calculate log density
```

```
# Standardise gout dataset doses relative to maximum investigated dose
std.gout <- GoutSUA_2wkCFB
std.gout <- std.gout %>% group_by(agent) %>% mutate(dose=dose/max(dose))
std.gout$dose[std.gout$agent=="Plac"] <- 0
std.gout <- mbnma.network(std.gout)

exponential <- mbnma.exponential(std.gout, lambda="rel")
```

pD (effective number of parameters)

The default value in for `pd` in `mbnma.run()` is `"pv"`, which uses the value automatically calculated in the `R2jags` package as `pv = var(deviance)/2`. Whilst this is easy to calculate, it is numerically less stable than `pD` and may perform more poorly in certain conditions (Gelman, Hwang, and Vehtari 2014).

A commonly-used approach for calculating pD is the plug-in method (`pd="plugin"`) (Spiegelhalter et al. 2002). However, this can sometimes result in negative non-sensical values due to skewed posterior distributions for deviance contributions that can arise when fitting non-linear models.

Another approach that is more reliable than the plug-in method when modelling non-linear effects is using the Kullback-Leibler divergence (`pd="pd.kl"`) (Plummer 2008). The disadvantage of this approach is that it requires running additional MCMC iterations, so can be slightly slower to calculate.

Finally, pD can also be calculated using an optimism adjustment (`pd="popt"`) which allows for calculation of the penalized expected deviance (Plummer 2008). This adjustment allows for the fact that data used to estimate the model is the same as that used to assess its parsimony. It also requires running additional MCMC iterations.

Arguments to be sent to JAGS

In addition to the arguments specific to `mbnma.run()` it is also possible to use any arguments to be sent to

R2jags::jags(). Most of these are likely to relate to improving the performance of MCMC simulations in JAGS. Some of the key arguments that may be of interest are:

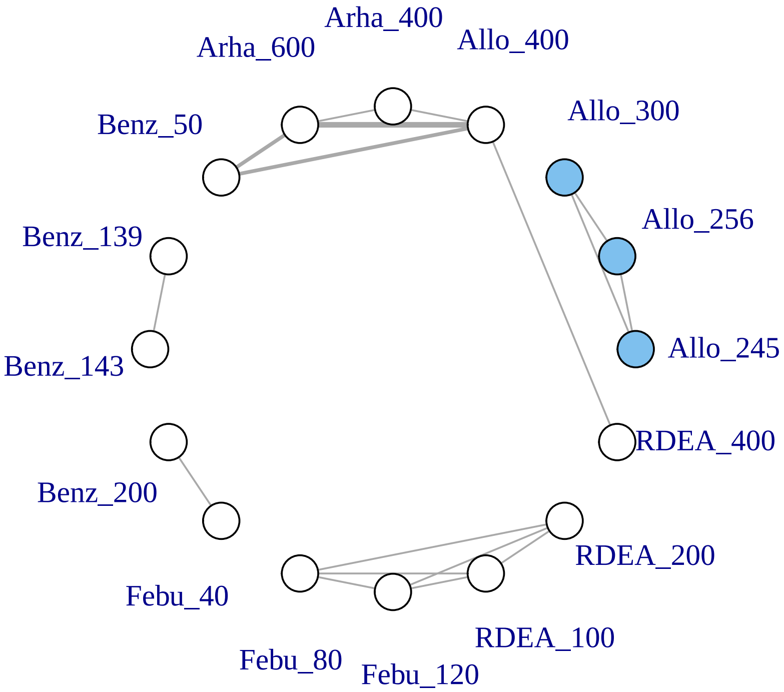
- n.chains The number of Markov chains to run (default is 3)
- n.iter The total number of iterations per MCMC chain
- n.burnin The number of iterations that are discarded to ensure iterations are only saved once chains have converged
- n.thin The thinning rate which ensures that results are only saved for 1 in every n.thin iterations per chain. This can be increased to reduce autocorrelation

Connecting networks via the dose-response relationship

One of the strengths of dose-response MBNMA is that it allows treatments to be connected in a network that might otherwise be disconnected, by linking up different doses of the same agent via the dose-respones relationship. To illustrate this we can generate a version of the gout dataset which excludes placebo (to artificially disconnect the network):

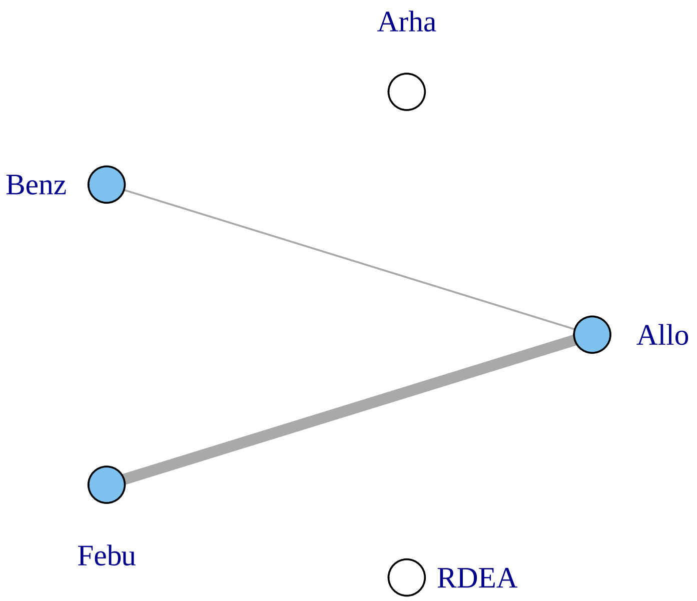
```
# Generate dataset without placebo
noplac.gout <- GoutSUA_2wkCFB[!GoutSUA_2wkCFB$studyID %in% c(2001, 3102),] # Drop two-arm placebo studi
noplac.gout <- noplac.gout[noplac.gout$agent!="Plac",] # Drop placebo arm from multi-arm studies

# Create mbnma.network object
noplac.net <- mbnma.network(noplac.gout)
```

```
# Plot network
plot(noplac.net, label.distance=5)
#> Warning in check.network(g): The following treatments/agents are not connected to the network refere
#> Allo_400
#> Arha_400
#> Arha_600
#> Benz_50
#> Benz_139
#> Benz_143
#> Benz_200
#> Febu_40
#> Febu_80
#> Febu_120
#> RDEA_100
#> RDEA_200
#> RDEA_400
```



This results in a very disconnected network, and if we were to conduct a conventional “split” NMA (whereby different doses of an agent are considered to be independent), we would only be able to estimate relative effects for a very small number of treatments. However, if we assume a dose-response relationship then these different doses can be connected via this relationship, and we can connect up more treatments and agents in the network.

```
# Network plot at the agent level illustrates how doses can connect using MBNMA
plot(noplac.net, level="agent", remove.loops = TRUE, label.distance = 4)
#> Warning in check.network(g): The following treatments/agents are not connected to the network refere
#> Arha
#> RDEA
```

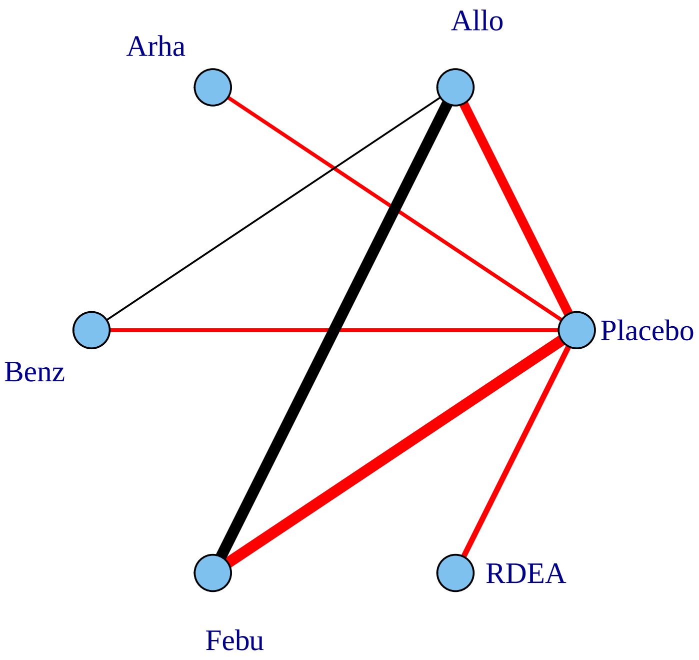


There are still two agents that do not connect to the network because they involve comparisons of different doses of the same agent. However, multiple doses of an agent within a study allow us to estimate the dose-response relationship and tell us something about the placebo (dose = 0) response - the number of different doses of an agent within a study will determine the degrees of freedom with which we are able to estimate a given dose-response function. Although the placebo response is not estimated directly in the MBNMA framework (it is modelled as a nuisance parameter), it allows us to connect the dose-response function estimated for an agent in one study, with that in another.

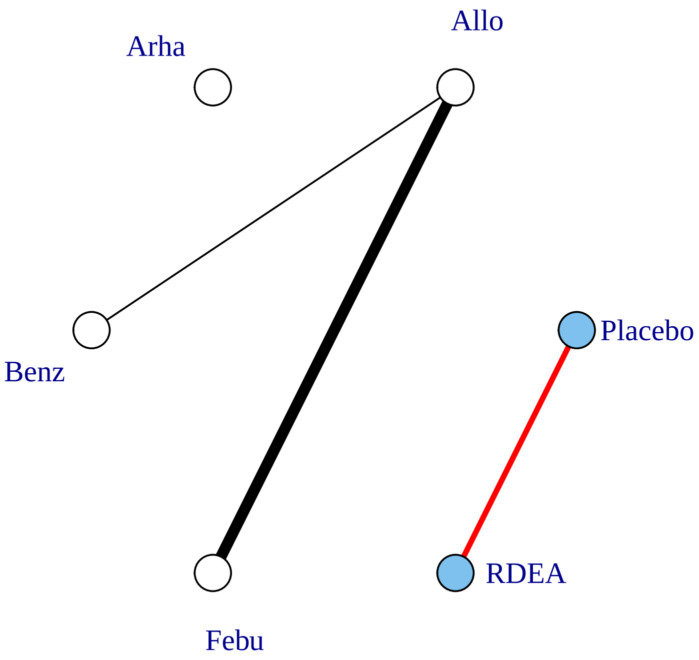
To visualise this, we can use the `doselink` argument in `plot(mbnma.network)`. The integer given to this argument indicates the minimum number of doses from which a dose-response function could be estimated, and is equivalent to the number of parameters in the desired dose-response function plus one. For example for an exponential function, we would require at least two doses on a dose-response curve (including placebo), since this would allow one degree of freedom with which to estimate the one-parameter dose-response function. By modifying the `doselink` argument we can determine the complexity of a dose-response function that we might expect to be able to estimate whilst still connecting all agents within the network.

If placebo is not included in the original dataset then this argument will also add a node for placebo to illustrate the connection.

```
# Network plot assuming connectivity via two doses
# Allows estimation of a single-parameter dose-response function
plot(noplac.net, level="agent", remove.loops = TRUE, label.distance = 4,
     doselink=2)
#> Dose-response connections to placebo plotted based on a dose-response
#>                    function with 1 degrees of freedom
```



```
# Network plot assuming connectivity via three doses
# Allows estimation of a two-parameter dose-response function
plot(noplac.net, level="agent", remove.loops = TRUE, label.distance = 4,
     doselink=3)
#> Warning in check.network(g): The following treatments/agents are not connected to the network refere
#> Allo
#> Arha
#> Benz
#> Febu
#> Dose-response connections to placebo plotted based on a dose-response
#>                     function with 2 degrees of freedom
```



In this way we can fully connect up treatments in an otherwise disconnected network, though unless informative prior information is used this will be limited by the number of doses of agents within included studies.

Non-parametric dose-response functions

Two non-parametric monotonic dose-response functions, `"nonparam.up"` and `"nonparam.down"` can be specified in `mbnma.run()`. They impose restrictions on the prior distributions of treatment effects that ensure that each increasing dose of an agent has an effect that is either the same or greater (most positive for `"nonparam.up"` and more negative for `"nonparam.down"`) than the previous dose, following the method of Owen et al. (2015). By making this assumption, this model is slightly more informative, and can lead to some slight gains in precision if relative effects are otherwise imprecisely estimated. However, because a functional form for the dose-response is not modelled, it cannot be used to connect networks that are disconnected at the treatment-level, but connected at the agent-level.

In the case of MBNMA, it may be useful to compare the fit of a non-parametric model to that of a parametric dose-response function, to ensure that fitting a parametric dose-response function does not lead to significantly poorer model fit.

When fitting a non-parametric dose-response model, there is no need to specify arguments for dose-response parameters (`beta.1`, `beta.2`, `beta.3`), as these are ignored in this modelling approach. `method` can still be used to specify either `"common"` or `"random"` effects. It is important to correctly choose either `"nonparam.up"` or `"nonparam.down"` depending on the expected direction of effect, or this can lead to computation errors.

```
nonparam <- mbnma.run(tripnet, fun="nonparam.up", method="random")
#> `likelihood` not given by user - set to `binomial` based on data provided
#> `link` not given by user - set to `logit` based on assigned value for `likelihood`
#> Modelling non-parametric dose-response - arguments for dose-response parameters `beta.1`, `beta.2`,
print(nonparam)
#> Inference for Bugs model at "C:\Users\hp17602\AppData\Local\Temp\RtmpwvhQlZ\file4c837223a68", fit us
#>  3 chains, each with 2000 iterations (first 1000 discarded)
#>  n.sims = 3000 iterations saved
#>            mu.vect sd.vect     2.5%      25%      50%      75%    97.5%
#> d.1[1,1]     0.000   0.000    0.000    0.000    0.000    0.000    0.000
#> d.1[1,2]     1.198   0.191    0.820    1.076    1.196    1.321    1.587
#> d.1[2,2]     1.767   0.124    1.519    1.683    1.771    1.849    2.017
#> d.1[3,2]     2.063   0.150    1.779    1.962    2.059    2.161    2.373
#> d.1[1,3]     1.108   0.215    0.677    0.967    1.105    1.256    1.517
#> d.1[2,3]     1.132   0.099    0.947    1.061    1.128    1.201    1.331
#> d.1[3,3]     1.117   0.252    0.631    0.953    1.118    1.285    1.595
#> d.1[4,3]     1.475   0.096    1.289    1.410    1.474    1.536    1.674
#> d.1[1,4]     1.299   0.224    0.867    1.149    1.295    1.441    1.741
#> d.1[2,4]     1.503   0.429    0.610    1.228    1.514    1.785    2.321
#> d.1[1,5]     0.688   0.308    0.096    0.477    0.683    0.895    1.314
#> d.1[2,5]     1.055   0.134    0.801    0.968    1.055    1.142    1.329
#> d.1[3,5]     1.453   0.232    1.001    1.299    1.447    1.606    1.921
#> d.1[1,6]     0.995   0.407    0.217    0.705    0.990    1.275    1.797
#> d.1[2,6]     1.261   0.135    1.008    1.166    1.255    1.352    1.538
#> d.1[3,6]     1.557   0.247    1.072    1.386    1.555    1.722    2.041
#> d.1[4,6]     1.823   0.358    1.097    1.585    1.834    2.061    2.519
#> d.1[5,6]     2.955   0.702    1.617    2.466    2.945    3.430    4.341
#> d.1[1,7]     0.630   0.238    0.186    0.462    0.623    0.790    1.112
#> d.1[2,7]     0.841   0.400    0.080    0.582    0.836    1.115    1.633
#> d.1[1,8]     0.565   0.353    0.036    0.285    0.523    0.781    1.364
#> d.1[2,8]     1.341   0.178    0.992    1.217    1.340    1.457    1.700
#> d.1[3,8]     1.642   0.107    1.437    1.569    1.642    1.711    1.866
#> sd           0.284   0.047    0.180    0.256    0.287    0.315    0.371
#> totresdev  185.861  19.398  150.049  172.655  184.975  197.880  227.914
#> deviance  1088.967  19.398 1053.155 1075.761 1088.081 1100.986 1131.020
#>            Rhat n.eff
#> d.1[1,1]  1.000     1
#> d.1[1,2]  1.014   340
#> d.1[2,2]  1.021   120
#> d.1[3,2]  1.006   410
#> d.1[1,3]  1.020   120
#> d.1[2,3]  1.083    29
#> d.1[3,3]  1.023   110
#> d.1[4,3]  1.080    30
#> d.1[1,4]  1.010   250
#> d.1[2,4]  1.013   160
#> d.1[1,5]  1.020  1300
#> d.1[2,5]  1.012   210
#> d.1[3,5]  1.007   320
#> d.1[1,6]  1.005  2200
#> d.1[2,6]  1.030    80
#> d.1[3,6]  1.024    96
#> d.1[4,6]  1.014   210
#> d.1[5,6]  1.070    35
#> d.1[1,7]  1.019   110
#> d.1[2,7]  1.005   470
#> d.1[1,8]  1.004  1200
#> d.1[2,8]  1.011   210
#> d.1[3,8]  1.055    43
#> sd        1.055   150
#> totresdev 1.003  3000
#> deviance  1.004  2700
#>
#> For each parameter, n.eff is a crude measure of effective sample size,
#> and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
#>
#> DIC info (using the rule, pD = var(deviance)/2)
#> pD = 188.1 and DIC = 1277.1
#> DIC is an estimate of expected predictive error (lower deviance is better).
```

In the output from non-parametric models, `d` parameters represent the relative effect for each treatment (specific dose of a specific agent) versus the reference treatment, similar to in a standard Network Meta-Analysis. The first index of `d` represents the dose identifier, and the second index represents the agent identifier. Information on the specific values of the doses is not included in the model, as only the ordering of them (lowest to highest) is important.

Note that post-estimation functions and plots (e.g. ranking, prediction) cannot be performed on non-parameteric models at the moment, as these models are more intended as a confirmation of MBNMA model fit rather than a tool for inference.

Post-Estimation

For looking at post-estimation in MBNMA we will demonstrate using results from an Emax MBNMA on the triptans dataset unless specified otherwise:

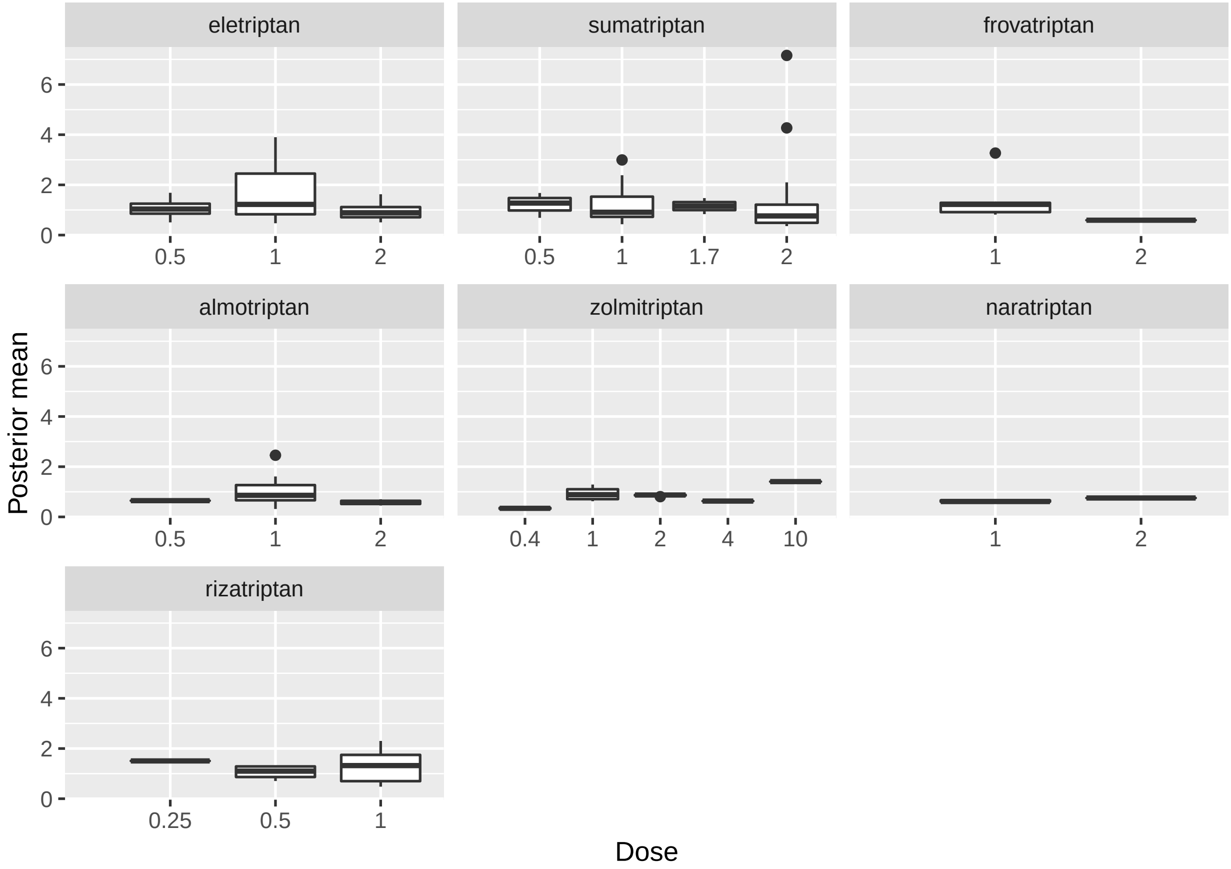
```
tripnet <- mbnma.network(HF2PPITT)
#> Values for `agent` with dose = 0 have been recoded to `Placebo`
#> agent is being recoded to enforce sequential numbering and allow inclusion of `Placebo`
trip.emax <- mbnma.emax(tripnet, emax="rel", ed50="rel")
#> `likelihood` not given by user - set to `binomial` based on data provided
#> `link` not given by user - set to `logit` based on assigned value for `likelihood`
#> Results for ED50 (`beta.2`) modelled on the exponential scale
```

Deviance plots

To assess how well a model fits the data, it can be useful to look at a plot of the contributions of each data point to the residual deviance. This can be done using `devplot()`. As individual deviance contributions are not automatically monitored in `parameters.to.save`, this might require the model to be automatically run for additional iterations.

Results can be plotted either as a scatter plot (`plot.type="scatter"`) or a series of boxplots (`plot.type="box"`).

```
# Plot boxplots of residual deviance contributions (scatterplot is the default)
devplot(trip.emax, plot.type = "box")
#> `resdev` not monitored in mbnma$parameters.to.save.
#> additional iterations will be run in order to obtain results for `resdev`
```



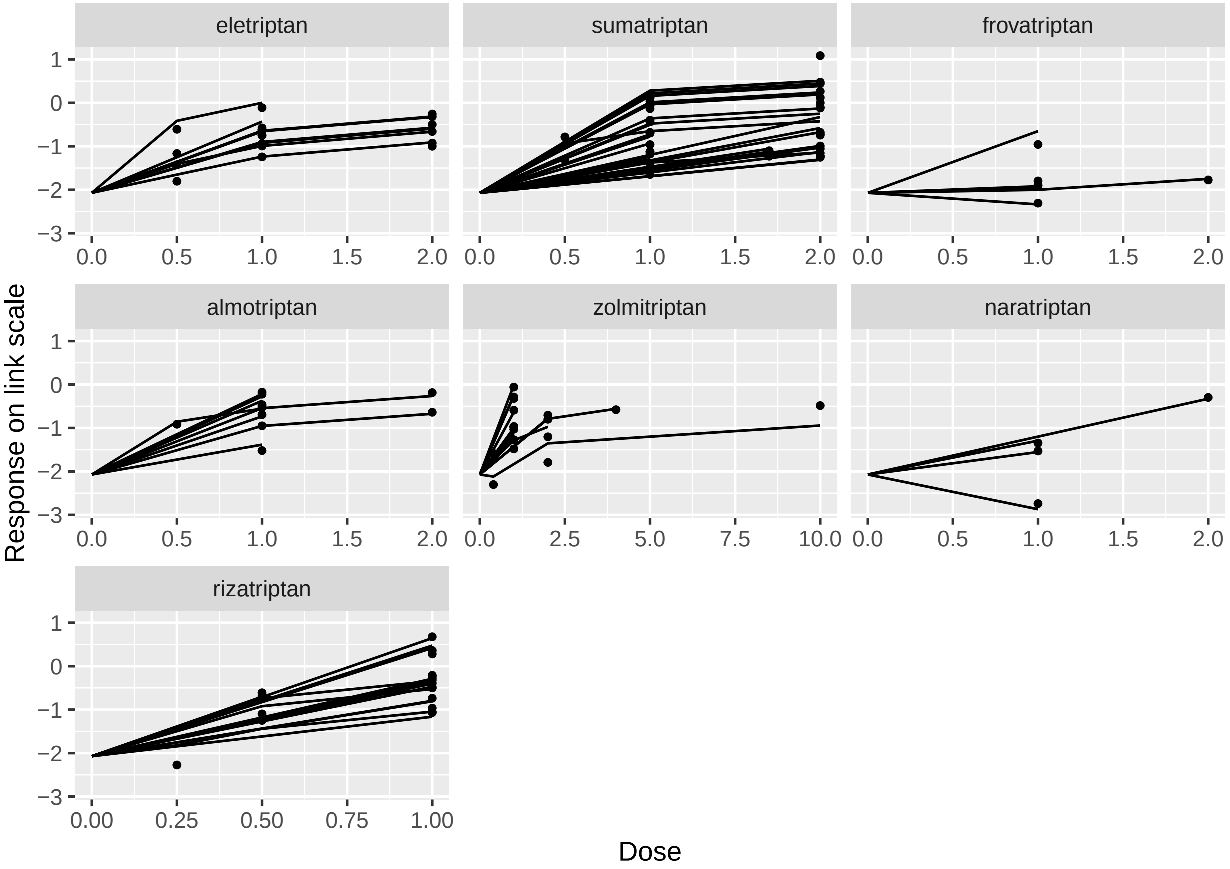
From these plots we can see that whilst the model fit does not seem to be systematically non-linear (which would suggest an alternative dose-response function may be a better fit), residual deviance is high at a dose of 1 for eletriptan, and at 2 for sumatriptan. This may indicate that fitting random effects may allow for additional variability in response which may improve the model fit.

If saved to an object, the output of `devplot()` contains the results for individual deviance contributions, and this can be used to identify any extreme outliers.

Fitted values

Another approach for assessing model fit can be to plot the fitted values, using `fitplot()`. As with `devplot()`, this may require running additional model iterations to monitor `theta`.

```
# Plot fitted and observed values with treatment labels
fitplot(trip.emax)
#> `theta` not monitored in mbnma$parameters.to.save.
#> additional iterations will be run in order to obtain results
```

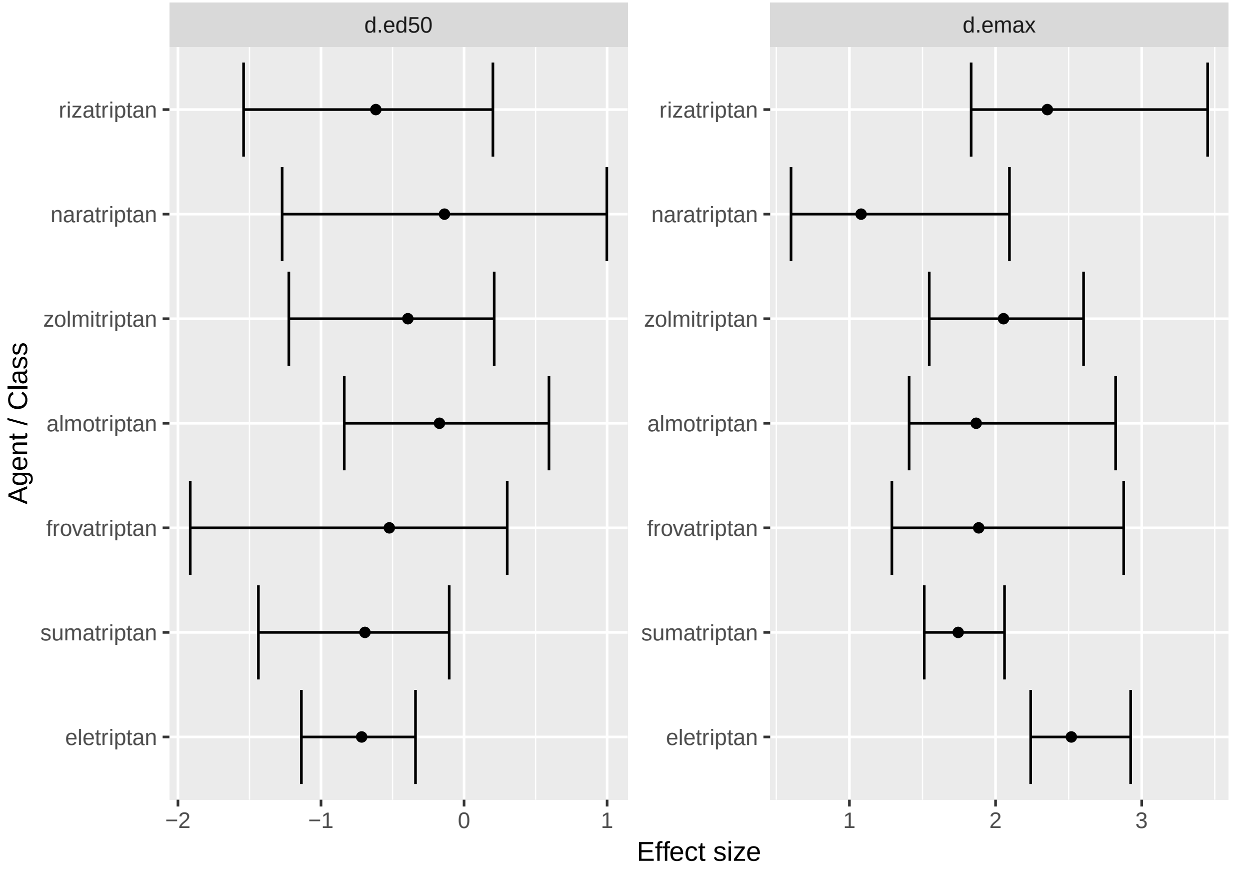


Fitted values are plotted as connecting lines and observed values in the original dataset are plotted as points. These plots can be used to identify if the model fits the data well for different agents and at different doses along the dose-response function.

Forest plots

Forest plots can be easily generated from MBNMA models using the `plot()` method on an `"mbnma"` object. By default this will plot a separate panel for each dose-response parameter in the model. Forest plots can only be generated for parameters which are modelled using relative effects and that vary by agent/class.

```
plot(trip.emax)
```



Ranking

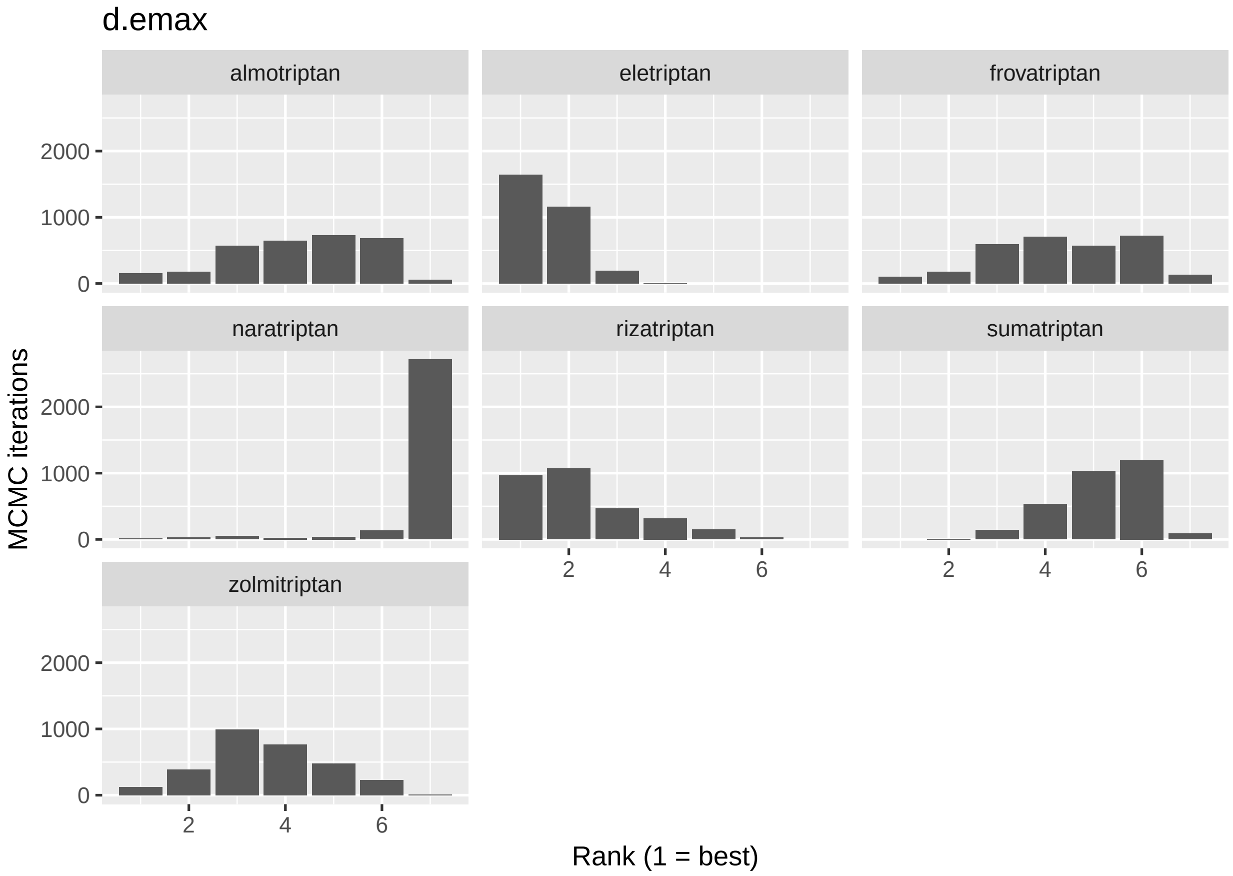
Rankings can be calculated for different dose-response parameters from MBNMA models by using `rank()` on an `"mbnma"` object. Any parameter monitored in an MBNMA model that varies by agent/class can be ranked. A vector of these is assigned to `params`. `direction` indicates whether negative responses should be ranked as "better" (`-1`) or "worse" (`1`).

```
ranks <- rank(trip.emax, direction=1)
summary(ranks)
#> $d.ed50
#>     rank.param     mean       sd 2.5% 25% 50% 75% 97.5%
#> 1   eletriptan 5.349667 1.250302    3   5   5   6     7
#> 2  sumatriptan 5.073667 1.572598    2   4   5   6     7
#> 3 frovatriptan 4.309000 1.919917    1   3   4   6     7
#> 4  almotriptan 2.488667 1.596680    1   1   2   3     7
#> 5 zolmitriptan 3.592667 1.724839    1   2   3   5     7
#> 6  naratriptan 2.498333 1.843673    1   1   2   3     7
#> 7  rizatriptan 4.688000 1.764188    1   3   5   6     7
#>
#> $d.emax
#>     rank.param     mean        sd 2.5% 25% 50% 75% 97.5%
#> 1   eletriptan 1.520333 0.6253401    1   1   1   2     3
#> 2  sumatriptan 5.186000 0.9253951    3   5   5   6     7
#> 3 frovatriptan 4.388667 1.4597883    1   3   4   6     7
#> 4  almotriptan 4.289667 1.4513656    1   3   4   5     6
#> 5 zolmitriptan 3.609333 1.2529576    1   3   3   4     6
#> 6  naratriptan 6.771667 0.8655135    3    7    7    7      7
#> 7  rizatriptan 2.234333 1.2068858    1    1    2    3      5
```

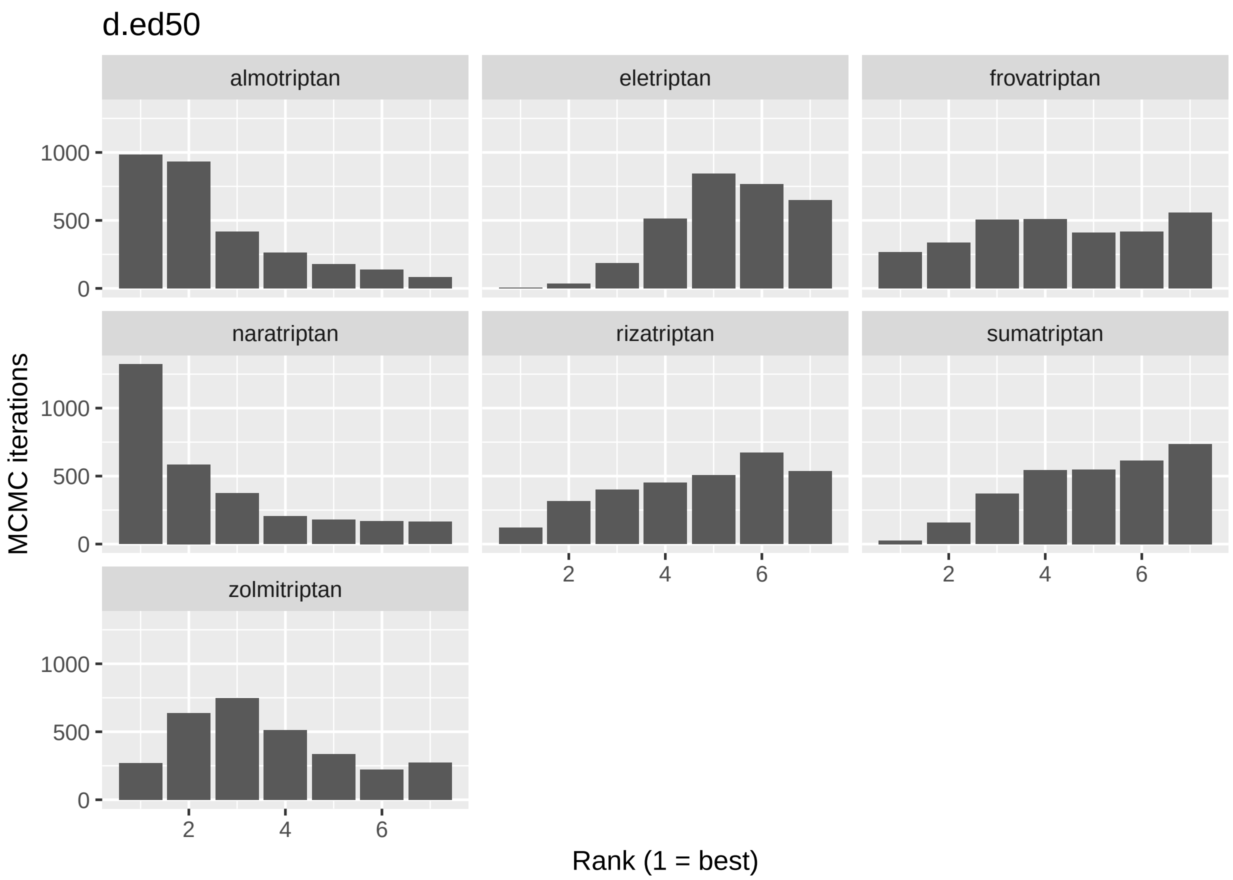
The output is an object of `class("mbnma.rank")`, containing a list for each ranked parameter in `params`, which consists of a summary table of rankings and raw information on agent/class (depending on argument given to `level`) ranking and probabilities. The summary median ranks with 95% credible intervals can be simply displayed using `summary()`.

Histograms for ranking results can also be plotted using the `plot()` method, which takes the raw MCMC ranking results stored in `mbnma.rank` and plots the number of MCMC iterations the parameter value for each treatment was ranked a particular position.

```
# Ranking histograms for Emax
plot(ranks, params = "d.emax")
```



```
# Ranking histograms for ED50
plot(ranks, params = "d.ed50")
```



Prediction

After performing an MBNMA, responses can be predicted from the model parameter estimates using `predict()` on an `"mbnma"` object. A number of important arguments should be specified for prediction.

`E0` This is the response at dose = 0 (equivalent to the placebo response). Since relative effects are the parameters estimated in MBNMA, the placebo response is not explicitly modelled and therefore must be provided by the user in some way. The simplest approach is to provide either a single numeric value for `E0` (deterministic approach), or a string representing a distribution for `E0` that can take any Random Number Generator (RNG) distribution for which a function exists in R (stochastic approach). Values should be given on the natural scale. For example for a binomial outcome:

- Deterministic: `E0 <- 0.2`
- Stochastic: `E0 <- "rbeta(n, shape1=2, shape2=10)"`

Another approach is to estimate `E0` from a set of studies. These would ideally be studies of untreated/placebo-treated patients that closely resemble the population for which predictions are desired, and the studies may be observational. However, synthesising results from the placebo arms of trials in the original network is also possible. For this, `E0` is assigned a data frame of studies in the long format (one row per study arm) with the variables `studyID`, and a selection of `y`, `se`, `r`, `N` and `E` (depending on the likelihood used in the MBNMA model). `synth` can be set to `"fixed"` or `"random"` to indicate whether this synthesis should be fixed or random effects.

```
E0 <- HF2PPITT[HF2PPITT$dose==0,]
```

Additionally, it's also necessary to specify the doses at which to predict responses. By default, `predict()` uses the maximum dose within the dataset for each agent, and predicts doses at a series of cut points. The number of cut points can be specified using `n.doses`, and the maximum dose to use for prediction for each agent can also be specified using `max.doses` (a named list of numeric values where element names correspond to agent names).

An alternative approach is to predict responses at specific doses for specific agents using the argument `exact.doses`. As with `max.doses`, this is a named list in which element names correspond to agent names, but each element is a numeric vector in which each value within the vector is a dose at which to predict a response for the given agent.

```
# Predict 20 doses for each agent, with a stochastic distribution for E0
doses <- list("Placebo"=0,
                  "eletriptan"=3,
                  "sumatriptan"=3,
                  "almotriptan"=3,
                  "zolmitriptan"=3,
                  "naratriptan"=3,
                  "rizatriptan"=3)

pred <- predict(trip.emax, E0="rbeta(n, shape1=2, shape2=10)",
                      max.doses=doses, n.dose=20)



# Predict exact doses for two agents, and estimate E0 from the data
E0.data <- HF2PPITT[HF2PPITT$dose==0,]
doses <- list("eletriptan"=c(0,1,3),
                  "sumatriptan"=c(0,3))
```

```
pred <- predict(trip.emax, E0=E0.data,
                      exact.doses=doses)
#> `link` not given by user - set to `logit` based on assigned value for `likelihood`
#> Values for `agent` with dose = 0 have been recoded to `Placebo`
#> agent is being recoded to enforce sequential numbering and allow inclusion of `Placebo`
```

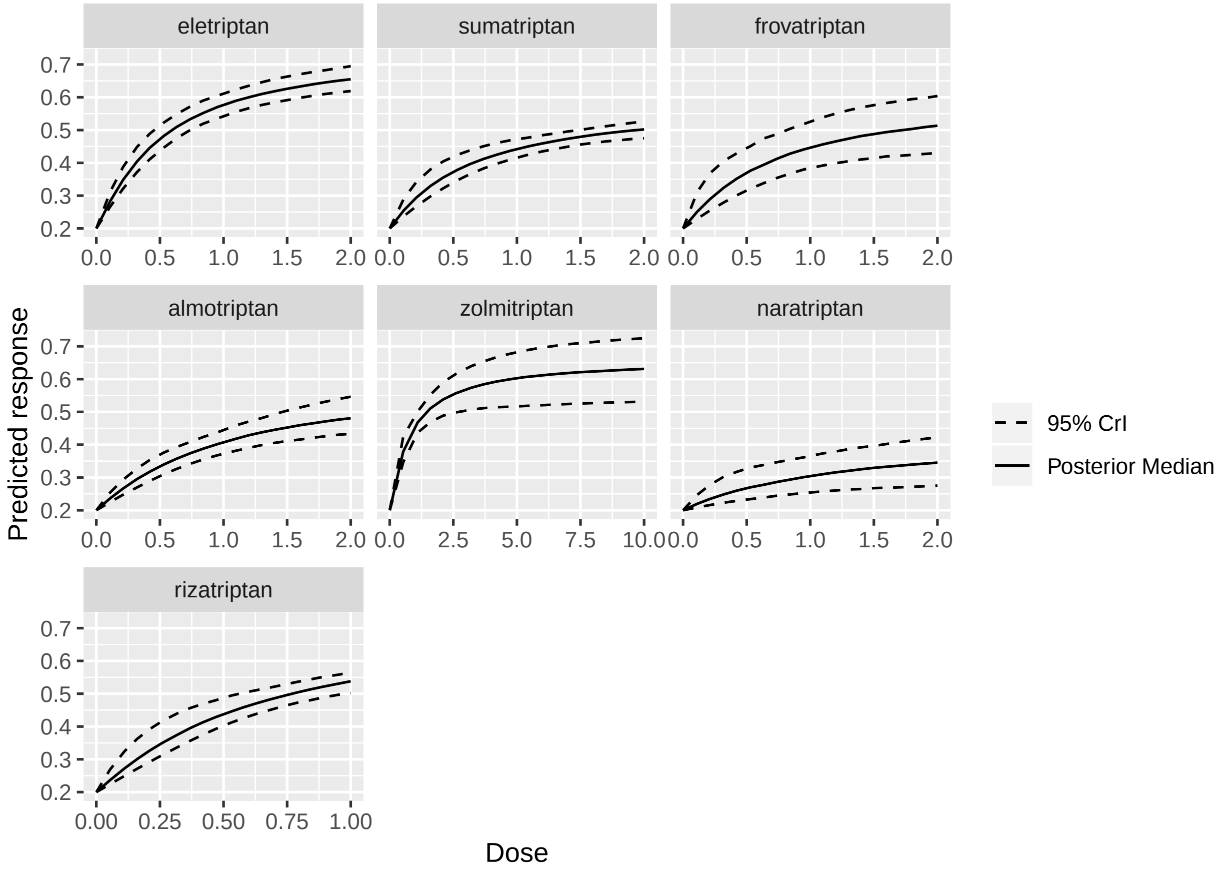
An object of class `"mbnma.predict"` is returned, which is a list of summary tables and MCMC prediction matrices for each treatment (combination of dose and agent). The `summary()` method can be used to print mean posterior predictions at each time point for each treatment.

```
summary(pred)
#>         agent dose      mean          sd      2.5%       25%       50%
#> 1  eletriptan    0 0.1240604 0.003308866 0.1175943 0.1218852 0.1240392
#> 2  eletriptan    1 0.4361478 0.019118415 0.3997727 0.4229120 0.4358769
#> 3  eletriptan    3 0.5549116 0.027408683 0.5054910 0.5345451 0.5542398
#> 4 sumatriptan    0 0.1240604 0.003308866 0.1175943 0.1218852 0.1240392
#> 5 sumatriptan    3 0.3867294 0.017660379 0.3512009 0.3747711 0.3865771
#>         75%     97.5%
#> 1 0.1263792 0.1304690
#> 2 0.4492309 0.4744319
#> 3 0.5729344 0.6109603
#> 4 0.1263792 0.1304690
#> 5 0.3984916 0.4220573
```

Plotting predicted responses

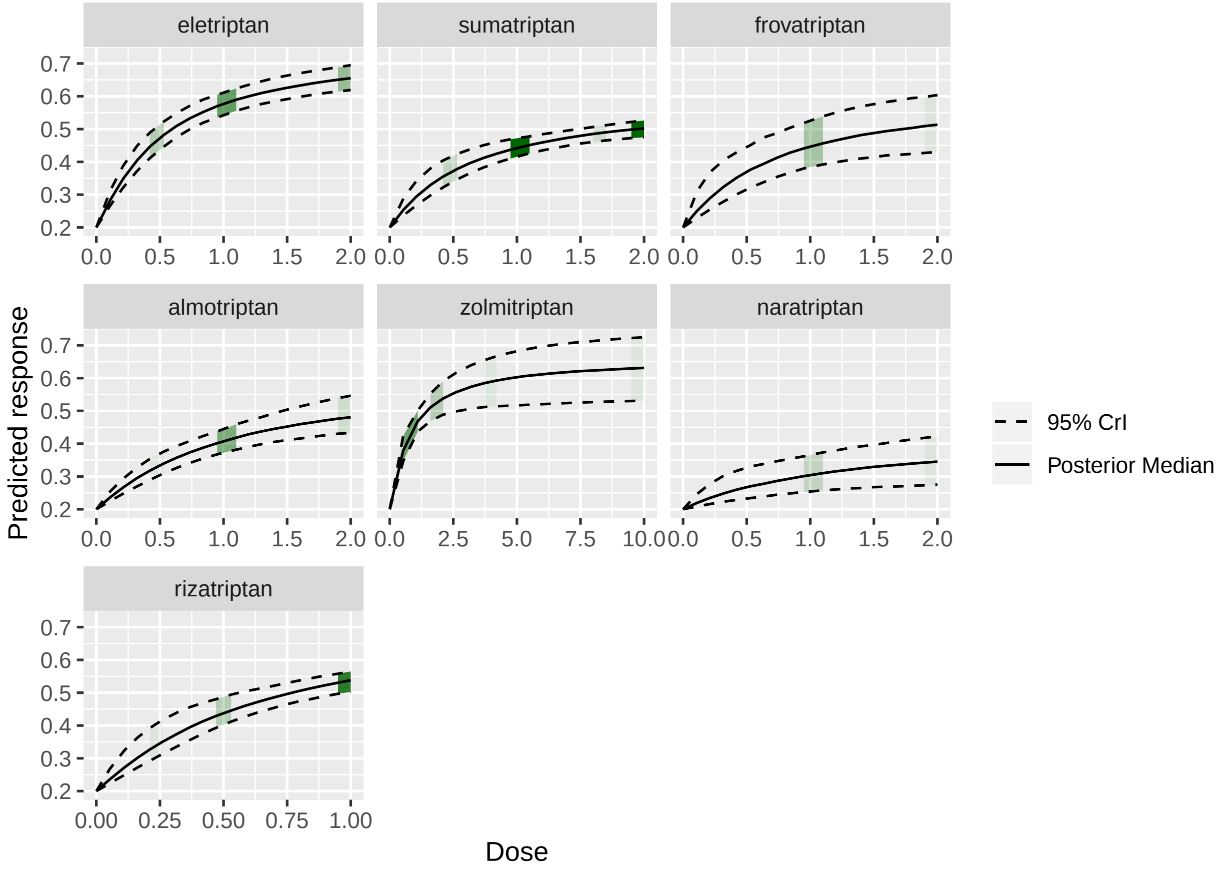
Predicted responses can also be plotted using the `plot()` method on an object of `class("mbnma.predict")`. The predicted responses are joined by a line to form the dose-response curve for each agent predicted, with 95% credible intervals (CrI). Therefore, when plotting the response it is important to predict a sufficient number of doses (using `n.doses`) to get a smooth curve.

```
# Predict responses using default doses up to the maximum of each agent in the dataset
pred <- predict(trip.emax, E0=0.2, n.dose=20)

plot(pred)
```



Shaded counts of the number of studies in the original dataset that investigate each dose of an agent can be plotted over the 95% CrI for each treatment by setting `disp.obs = TRUE`, though this requires that the original `"mbnma.network"` object used to estimate the MBNMA be provided via `network`.

```
plot(pred, disp.obs = TRUE, network=tripnet)
#> 66 placebo arms in the dataset are not shown within the plots
```



This can be used to identify any extrapolation/interpretation of the dose-response that might be ocurring for a particular agent. As you can see, more observations typically leads to tighter 95% CrI for the predicted response at a particular point along the dose-response curve.

We can also plot the results of a "split" Network Meta-Analysis (NMA) in which all doses of an agent are assumed to be independent. As with `disp.obs` we also need to provide the original `mbnma.network` object to be able to estimate this, and we can also specify if we want to perform a common or random effects NNMA using `method`. Treatments that are only connected to the network via the dose-response relationship (rather than by a direct head-to-head comparison) will not be included.

```r
plot(pred, overlay.split = TRUE, network=tripnet, method="common")
```

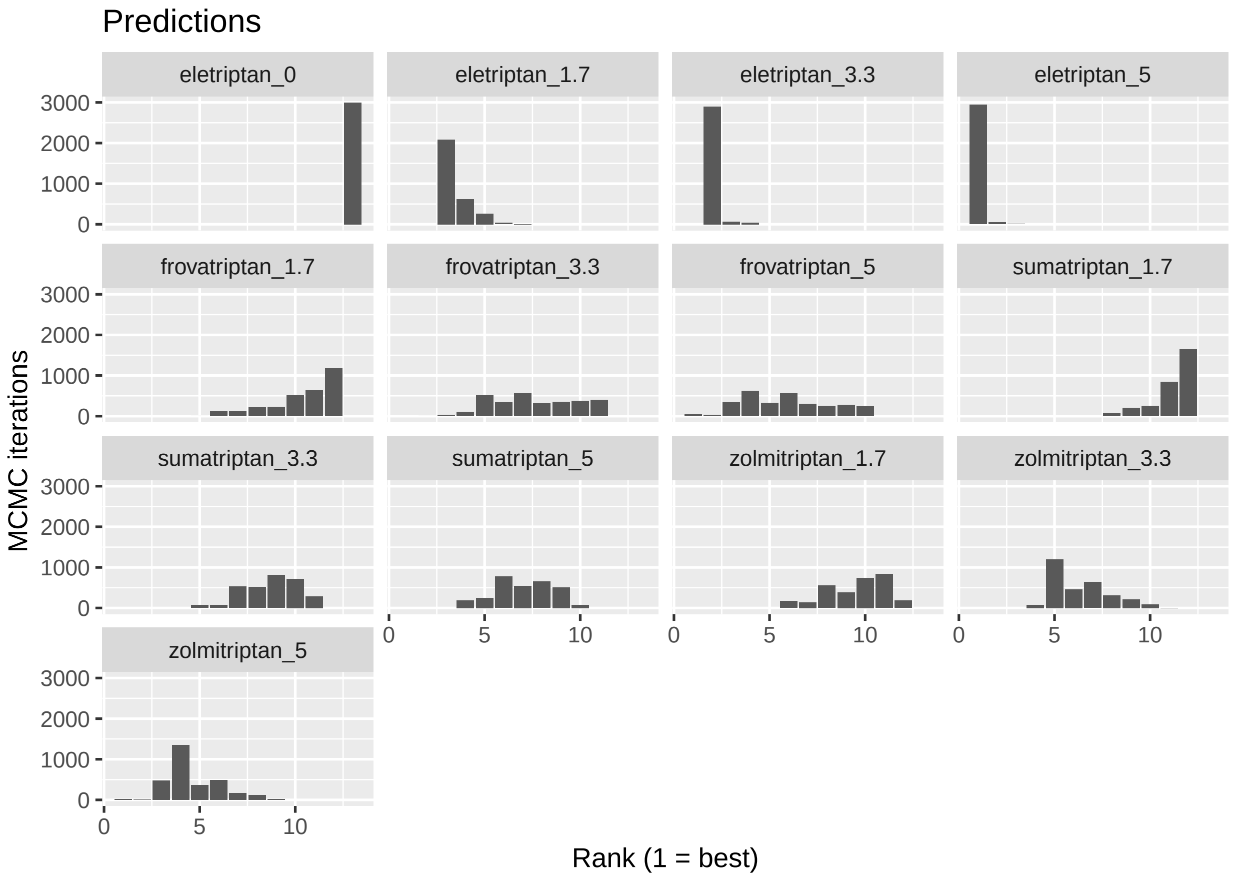
By plotting these, as well as observing how responses can be extrapolated/interpolated, we can also see which doses are likely to be providing most information to the dose-response relationship. The tighter 95% CrI on the predicted responses from the MBNMA also show that modelling the dose-response function also gives some additional precision even at doses for which there is information available.

Ranking predicted responses

Predicted responses from an object of `class("mbnma.predict")` can also be ranked using the `rank()` method. As when applied to an object of `class("mbnma")`, this method will rank parameters (in this case predictions) in order from either highest to lowest (`direction=1`) or lowest to highest (`direction=-1`), and return an object of `class("mbnma.rank")`.

If there have been predictions at dose = 0 for several agents only one of these will be included in the rankings, in order to avoid duplication (since the predicted response at dose = 0 is the same for all agents).

```r
pred <- predict(trip.emax, E0=0.2, n.doses=4,
                max.doses = list("eletriptan"=5, "sumatriptan"=5,
                                  "frovatriptan"=5, "zolmitriptan"=5))

ranks <- rank(pred)
plot(ranks)
```



Consistency Testing

When performing a MBNMA by pooling relative treatment effects, the modelling approach assumes consistency between direct and indirect evidence within a network. This is an incredibly useful assumption as it allows us to improve precision on existing direct estimates, or to estimate relative effects between treatments that have not been compared in head-to-head trials, by making use of indirect evidence.

However, if this assumption does not hold, this is extremely problematic for inference, so it is important to be able to test it. A number of different approaches exist to allow for this in standard Network Meta-Analysis (NMA) (Dias et al. 2013), but within dose-respones MBNMA there is added complexity because the consistency assumption can be conceptualised either for each treatment comparison (combination of dose and agent), or for each agent, where consistency is required for the agent-level parameters governing the dose-response relationship.

Testing for consistency at the agent-level is challenging as there is unlikley to be the required data available to be able to do this - included studies in the dataset must have multiple doses of multiple agents, so that sufficient information is available to estimate dose-response parameters *within that study.* However, testing for consistency at the treatment-level is possible, and this follows standard NMA methods which will be described below. In practice, testing for consistency at the treatment-level should suffice, as any inconsistency identified at the treatment level will also translate to inconsistency at the agent level and vice versa *[manuscript in progress]*.

Consistency also depends on the functional form assumed for the dose-response relationship, and so is inextricably linked to model fit of the dose-response relationship. A thorough assessment of the validity of the fitted model is therefore important to be confident that the resulting treatment effect estimates provide a firm basis for decision making.

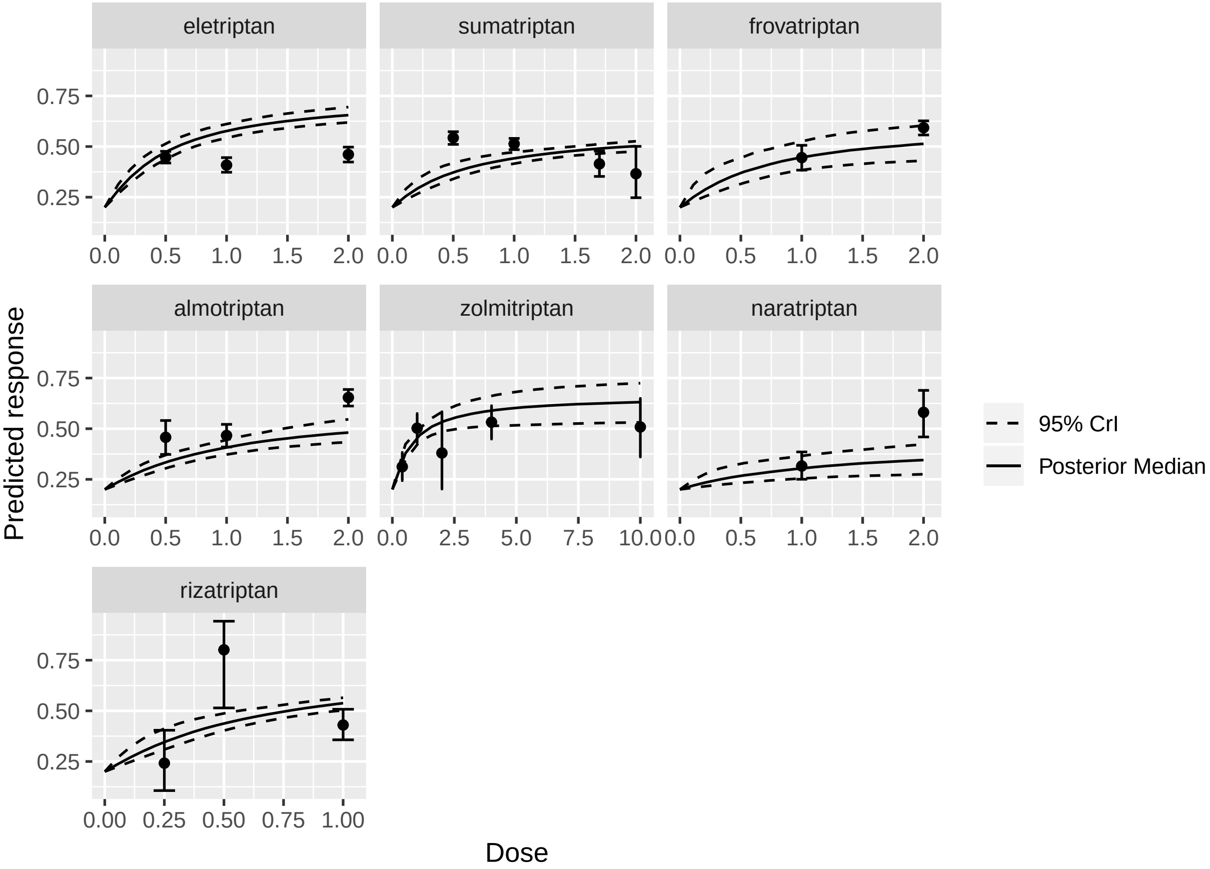
When meta-analysing dose-response studies, the potential for inconsistency testing may actually be reasonably rare, as most (if not all) trials will be multi-arm placebo-controlled. Since each study is internally consistent (the relative effects within the trial will always adhere to consistency relationships), there will be no closed loops of treatments that are informed by independent sources of evidence.

Unrelated Mean Effects (UME) model at the treatment level

To check for consistency at the treatment level using UME we fit an NMA that does not assume consistency relationships, and that only models the direct relative effects between each arm in a study and the study reference treatment. If the consistency assumption holds true then the results from the UME model and the NMA will be very similar. However, if there is a discrepancy between direct and indirect evidence in the network, then the consistency assumption may not be valid, and the UME results are likely differ in several ways:

- The UME model may provide a better fit to the data, as measured by residual deviance
- The between-study SD for different parameters may be lower in the UME model
- Individual relative effects may differ in magnitude or (more severely) in direction for different treatment comparisons between UME and NMA models.

```r
# Using the alogliptin dataset
alognet <- mbnma.network(alog_pcfb)
nma <- nma.run(alognet, method="random")
ume <- nma.run(alognet, method="random", UME = TRUE)
```



Model	Residual Deviance	Betwen-study SD
NMA	47.28	0.12 (0.073, 0.18)
UME	45.53	0.14 (0.089, 0.22)

As both the residual deviances and between-study SDs are similar, this would suggest that there is no evidence of inconsistency in this network, though comparing results for individual comparisons from the models can also be useful.

Node-splitting at the treatment level

Another approach for consistency checking is node-splitting. This splits contributions for a particular treatment comparison into direct and indirect evidence, and the two can then be compared to test their similarity (Valkenhoef et al. 2016). `NMA.nodesplit()` takes similar arguments to `NMA.run()`, and returns an object of `class("NMA.nodesplit")`.

In addition to these, the argument `comparisons` can be used to indicate which treatment comparisons to perform a nodesplit on. If left as `NULL` (the default) node-splits will automatically be performed in all closed loops of treatments in which comparisons are informed by independent sources of evidence. This is somewhat similar to the function `gemtc::mtc.nodesplit.comparisons()`, but uses a fixed network reference treatment and therefore identifies fewer loops in which to test for inconsistency.

As several NMA models will need to be run for each treatment comparison to split, this function can take some time to run.

```r
# Using the triptans dataset
tripnet <- mbnma.network(HF2PPITT)

# Nodesplit using a random effects model
nodesplit <- nma.nodesplit(tripnet, method="random")
```

```r
print(nodesplit)
#> ========================================
#> Node-splitting analysis of inconsistency
#> ========================================
#>  comparison  p.value         Median (95% CrI)
#> 12v7     0.1636
#> -> direct                0.288 (-0.282, 0.857)
#> -> indirect              0.904 (0.324, 1.49)
#> -> NMA                   0.213 (-0.208, 0.66)
#> 
#> 4v3      0.3939
#> -> direct                0.122 (-0.536, 0.763)
#> -> indirect              0.35 (0.0402, 0.668)
#> -> NMA                   0.216 (-0.125, 0.547)
#> 
#> 5v2      0.2407
#> -> direct                0.184 (-0.466, 0.781)
#> -> indirect              0.559 (0.272, 0.852)
#> -> NMA                   0.498 (0.251, 0.765)
#> 
#> 3v2      0.2611
#> -> direct                -0.403 (-1.04, 0.213)
#> -> indirect              -0.000421 (-0.33, 0.328)
#> -> NMA                   -0.0896 (-0.374, 0.207)
```

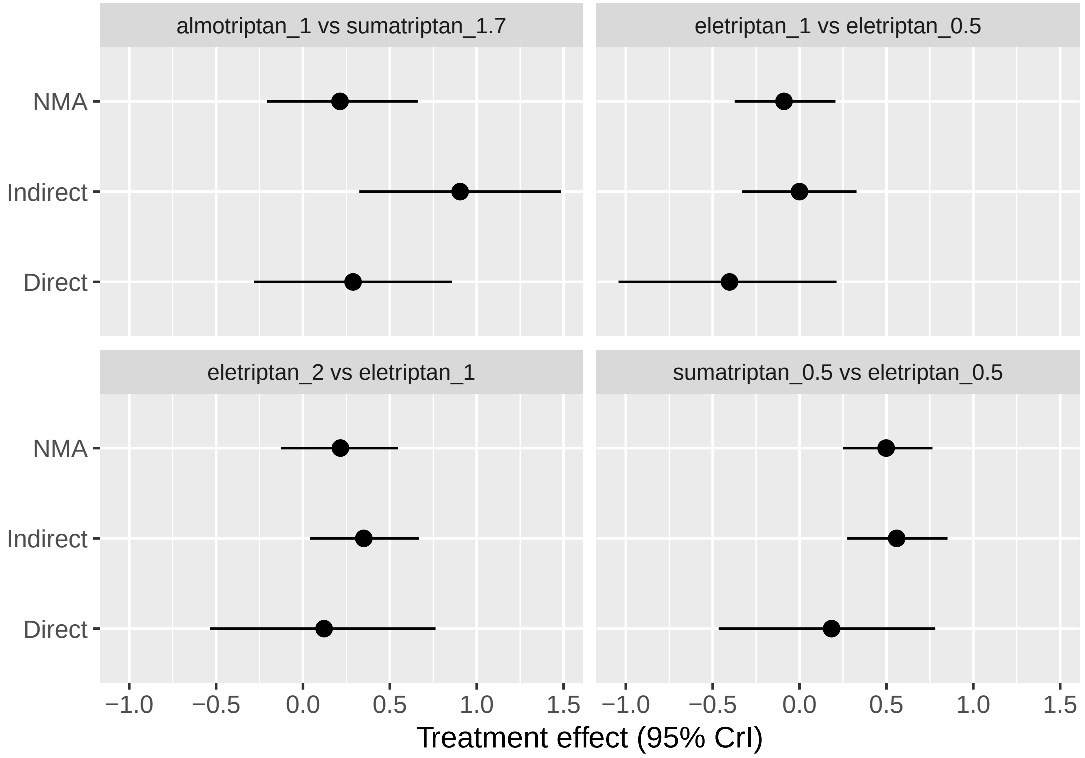
Performing the `print()` method on an object of `class("NMA.nodesplit")` prints a summary of the node-split results to the console, whilst the `summary()` method will return a data frame of posterior summaries for direct and indirect estimates for each split treatment comparison.

The nodesplit object itself is a list with results for each treatment comparison that has been split. There is a lot of information within the results, but the most useful (and easily interpretable) elements are:

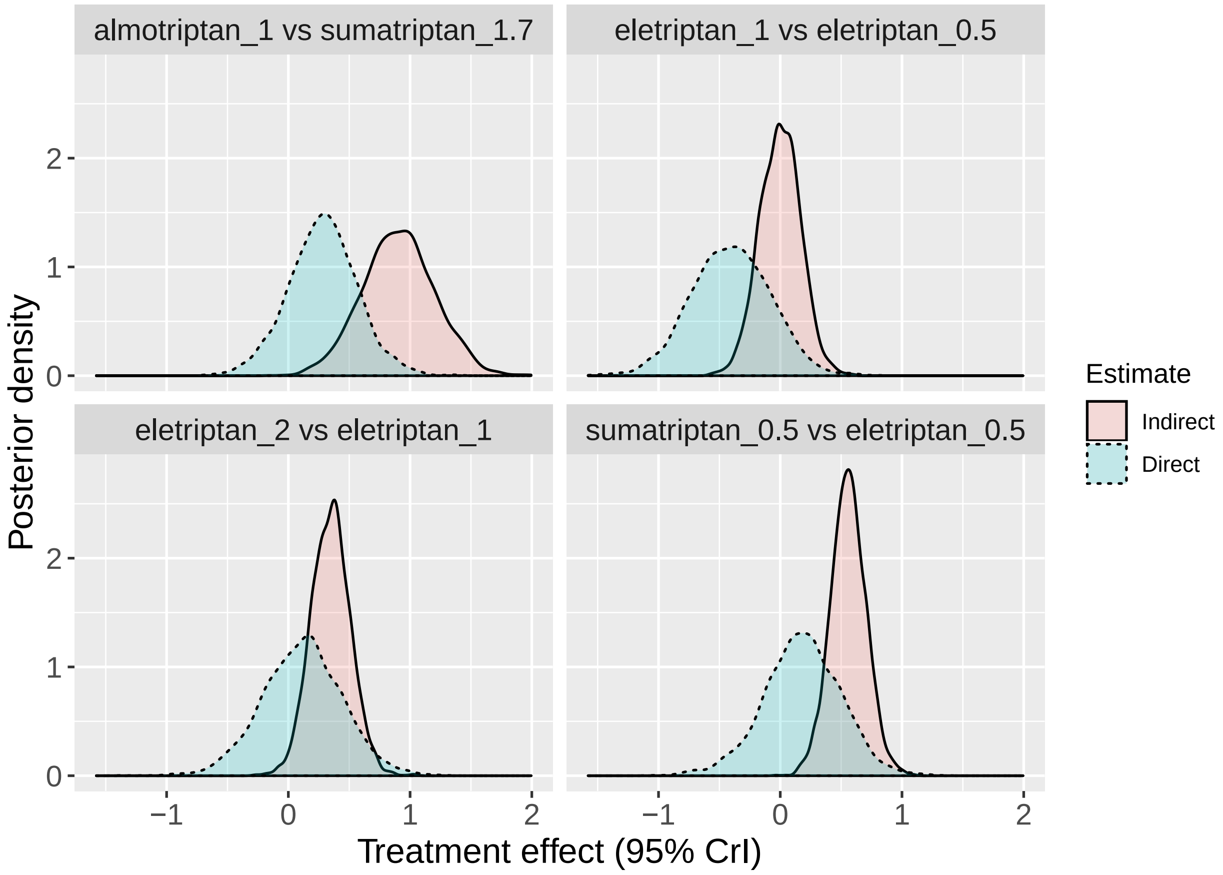
- `p.values` the Bayesian p-value for the posterior overlap between direct and indirect estimates
- `quantiles` the median and 95%CrI of the posterior distributions for direct and indirect evidence, and for the difference between them.
- `forest.plot` a forest plot that shows the median and 95% CrI for direct and indirect estimates
- `density.plot` a plot that shows the posterior distributions for direct and indirect estimates

It is possible to generate different plots of each nodesplit comparison using `plot()`:

```r
# Plot forest plots of direct, indirect and pooled (NMA) results for each comparison
plot(nodesplit, plot.type="forest")
```



```
# Plot posterior densities of direct and indirect results for each nodesplit comparisons
plot(nodesplit, plot.type="density")
```



Summary

MBNMAdose provides a complete set of functions that allow for performing dose-response MBNMA, model checking, prediction, and plotting of a number of informative graphics. By modelling a dose-response relationship within the network meta-analysis framework, this method can help connect networks of evidence that might otherwise be disconnected, allow extrapolation and interpolation of dose-response, and improve precision on predictions and relative effects between agents.

The package allows a range of dose-response functions (as well as the possibility to incorporate user-defined functions) and facilitates model specification in a way which allows users to make additional modelling assumptions to help identify parameters.

References

Dias, S., N. J. Welton, A. J. Sutton, D. M. Caldwell, G. Lu, and A. E. Ades. 2013. "Evidence Synthesis for Decision Making 4: Inconsistency in Networks of Evidence Based on Randomized Controlled Trials." Journal Article. *Med Decis Making* 33 (5): 641–56. https://doi.org/10.1177/0272989X12455847.

Gelman, Andrew, Jessica Hwang, and Aki Vehtari. 2014. "Understanding Predictive Information Criteria for Bayesian Models." Journal Article. *Statistics and Computing* 24 (6): 997–1016. https://doi.org/10.1007/s11222-013-9416-2.

JAGS Computer Program. 2017. http://mcmc-jags.sourceforge.net.

Langford, O., J. K. Aronson, G. van Valkenhoef, and R. J. Stevens. 2016a. "Methods for Meta-Analysis of Pharmacodynamic Dose-Response Data with Application to Multi-Arm Studies of Alogliptin." Journal Article. *Stat Methods Med Res*. https://doi.org/10.1177/0962280216637093.

———. 2016b. “Methods for Meta-Analysis of Pharmacodynamic Dose-Response Data with Application to Multi-Arm Studies of Alogliptin.” Journal Article. *Stat Methods Med Res*. https://doi.org/10.1177/0962280216637093.

Lu, G., and A. E. Ades. 2004. “Combination of Direct and Indirect Evidence in Mixed Treatment Comparisons.” Journal Article. *Stat Med* 23 (20): 3105–24. https://doi.org/10.1002/sim.1875.

Mawdsley, D., M. Bennetts, S. Dias, M. Boucher, and N. J. Welton. 2016. “Model-Based Network Meta-Analysis: A Framework for Evidence Synthesis of Clinical Trial Data.” Journal Article. *CPT Pharmacometrics Syst Pharmacol* 5 (8): 393–401. https://doi.org/10.1002/psp4.12091.

Owen, R. K., D. G. Tincello, and R. A. Keith. 2015. “Network Meta-Analysis: Development of a Three-Level Hierarchical Modeling Approach Incorporating Dose-Related Constraints.” Journal Article. *Value Health* 18 (1): 116–26. https://doi.org/10.1016/j.jval.2014.10.006.

Pedder, H., S. Dias, M. Bennetts, M. Boucher, and N. J. Welton. 2019. “Modelling Time-Course Relationships with Multiple Treatments: Model-Based Network Meta-Analysis for Continuous Summary Outcomes.” Journal Article. *Res Synth Methods* 10 (2): 267–86.

Plummer, M. 2008. “Penalized Loss Functions for Bayesian Model Comparison.” Journal Article. *Biostatistics* 9 (3): 523–39. https://doi.org/10.1093/biostatistics/kxm049.

Spiegelhalter, D. J., N. G. Best, B. P. Carlin, and A. van der Linde. 2002. “Bayesian Measures of Model Complexity and Fit.” Journal Article. *J R Statistic Soc B* 64 (4): 583–639.

Valkenhoef, G. van, S. Dias, A. E. Ades, and N. J. Welton. 2016. “Automated Generation of Node-Splitting Models for Assessment of Inconsistency in Network Meta-Analysis.” Journal Article. *Res Synth Methods* 7 (1): 80–93. https://doi.org/10.1002/jrsm.1167.