Practical 6. Network meta-analysis — SOLUTIONS

Lecture 6 PDF version

Fixed effects NMA

The data from the smoking cessation studies discussed in the lecture are included in the file `smoke.Rdata` and can be loaded into the R workspace using the `load` command.

```
# Loads the data (assuming they are in the current folder)
load("smoke.Rdata")
```

We can also use other built-in R commands to inspect the object we have just loaded into our workspace to figure out what is stored in it, for example as in the following.

```
# List the objects present in the workspace
ls()
```

```
## [1] "current_repo" "smoke.list"
```

```
# What type of object is 'smoke.list'?
class(smoke.list)
```

```
## [1] "list"
```

```
# What's in the data list?
names(smoke.list)
```

```
## [1] "r"  "n"  "t"  "na" "NS" "NT"
```

The command `ls()` simply lists all the objects currently present in the workspace. In this case, we only have an object `smoke.list`, which has been created when using the `load` command above. We can check its "class" (in this case, unsurprisingly, a list) and show the names of the elements contained in it, using the command `names`.

We can ask R to tell us more about these variables; for instance, we can inspect each variable's "class" (e.g.its nature) using the following helpful R command.

```
lapply(smoke.list,class)
```

```
## $r
## [1] "matrix" "array"
##
## $n
## [1] "matrix" "array"
##
## $t
## [1] "matrix" "array"
##
## $na
## [1] "integer"
##
## $NS
## [1] "integer"
##
## $NT
## [1] "numeric"
```

The `R` function `lapply` can be used to apply a function to the elements of a list (such as `smoke.list`). In this case, we want `R` to tell us what class each of the elements of `smoke.list` belongs to, which is what the command returns — for instance, the object `r` inside the object `smoke.list` is a matrix, while the object `NS` is an integer. We can also visualise each, e.g. by using the following commands

```
# Shows the first few elements of the object r included inside the object smoke.list
head(smoke.list$r)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   79   77   NA   NA
## [2,]   18   21   NA   NA
## [3,]    8   19   NA   NA
## [4,]   75   NA  363   NA
## [5,]    2   NA    9   NA
## [6,]   58   NA  237   NA
```

```
# Shows the first few elements of the object n included inside the object smoke.list
head(smoke.list$n)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]  702  694   NA   NA
## [2,]  671  535   NA   NA
## [3,]  116  149   NA   NA
## [4,]  731   NA  714   NA
## [5,]  106   NA  205   NA
## [6,]  549   NA 1561   NA
```

More in details, the elements of `smoke.list` are:

- `NS`: the total number of studies included in our analysis (24);
- `NT`: the total number of interventions considered (4);
- `na`: a vector containing the number of arms included in each of the studies;
- `r`: a matrix with `NS` rows and `NT` columns, containing the number of subjects that in each study and under each treatment arms have been observed to quit smoking;
- `n`: a matrix with `NS` rows and `NT` columns, containing the total number of subjects observed in each study and under each treatment arms;
- `t`: a matrix with `NS` rows and 3 columns, identifying the label associated with the treatments included in each of the studies. Notice that there are 3 columns because all studies have at most 3 treatment involved (i.e. all studies compare either 2 or 3 treatments — cfr. the lecture slides). The treatments are labelled as 1 = No intervention; 2 = Self-help; 3 = Individual counselling; 4 = Group counselling.

Notice that some of the data will be associated with `NA` (i.e. "Not Available" or "missing"). In this case, these are not really "missing data", but rather indicate that that particular column is not relevant. For example, in study 1 there are only two arms (you can confirm this by asking `R` to tell what `na[1]` is). The matrix `r` has values 79 and 77 in the first two columns and then `NA` in the third and fourth column — this is because study 1 only had data on arm 1 and arm 2. Incidentally, you can check what these arms where by looking at the value of the first row in the matrix `t`

```
smoke.list$t[1,]
```

```
## t1 t2 t3
##  1  2 NA
```

which tells you that the first arm was treatment 1, the second arm was treatment 2 and the third arm was nothing.

You can do a similar check on row (i.e. study) 4, using the following commands.

```
# How many arms were used in study 4?
smoke.list$na[4]
```

```
## [1] 2
```

```
# What treatment arms were they?
smoke.list$t[4,]
```

```
## t1 t2 t3
##  1  3 NA
```

```
# Data on the number of subjects quitting smoke for study 4
smoke.list$r[4,]
```

```
## [1]  75  NA 363  NA
```

```
# Data on the total sample size in study 4
smoke.list$n[4,]
```

```
## [1] 731  NA 714  NA
```

As we can see, study 4 considered 2 arms (comparing interventions 1 and 3, i.e. No intervention vs Individual counselling) and the number of quitters out of the total number of subjects studied were 75/731 and 363/714, respectively

Now we are ready to prepare to run the model. First, we consider the "fixed-effect" specification, whose model code is included in the file `smokefix_model.txt`. One of the complications of this model code is in the use of the "nested index" notation. For example, the code

```
for(s in 1:NS) { # loop over studies
      for (a in 1:na[s])  { # loop over arms
         r[s,t[s,a]] ~ dbin(p[s,t[s,a]], n[s,t[s,a]])
         ...
```

can be interpreted in this way. Let us consider `s` = 1, i.e. the first study in our dataset. This consists of `na[1]` = 2 arms, which means we will have two observed data points in terms of number of subjects who quit smoking. This also means that the index `a` will run from 1 to `na[s]` = `na[1]` = 2. Moreover, `t[s,a]` is in this case `t[1,1]` = 1 and `t[s,a]` = `t[1,2]` = 2, which in turns means that the above code effectively means that we are using the following modelling assumptions.

```
r[1,1] ~ dbin(p[1,1], n[1,1])
r[1,2] ~ dbin(p[1,2], n[1,2])
```

The use of the nested index notation is a clever shortcut for the full specification of all the cases (for all the studies and for all arms specified in each study) and it is particularly helpful for "non-rectangular" data (i.e. when not all the rows have data on the same number of columns).

From a more substantial point of view, we are modelling the logit of the study- and arm-specific probability of quitting smoking using a linear term made by an overall study-specific mean (`mu[s]`) and an incremental term (`delta[s,t[s,a]]`), which accounts for the "treatment effect".

```
logit(p[s,t[s,a]]) <- mu[s] + delta[s,t[s,a]]
```

Again, we are using the nested index notation in exactly the same way as above. In addition, what characterises this model as a "fixed-effect" specification is the distributional assumption on the incremental effects `delta`. These are modelled as follows.

```
## log odds ratio compared to treatment in arm 1 of study s
delta[s,t[s,1]] <- 0
for (a in 2:na[s])  {
   delta[s,t[s,a]] <- d[t[s,a]] - d[t[s,1]]
}
```

Firstly, we set the "baseline" intervention for each study `s`. In particular, we arbitrarily assume that the first intervention (associated with the index identified by `t[s,1]`) is the reference one for study `s` and as such, we (again, arbitrarily) assign it an "extra" effect of 0. Obviously, this means that for the reference treatment, the study- and arm-specific probability of quitting smoking is simply `mu[s]`, because in that case we would be adding to the linear predictor a value for the corresponding `delta` equal to 0. Any other intervention (from 2 to `na[s]`) is modelled through the difference between its specific value `d[t[s,a]]` and the value associated with the study-specific reference intervention, `d[t[s,1]]`.

Let us make a specific example: consider for example study `s` = 21. The details are as below.

```
# How many arms?
smoke.list$na[21]
```

```
## [1] 3
```

```
# Which arms?
smoke.list$t[21,]
```

```
## t1 t2 t3
##  2  3  4
```

This means that this study compares interventions 2, 3 and 4 (Self-help, Individual and Group counselling) and that, arbitrarily, we assume that the one in the first column of `t[21,]` is the reference — in this case, that is Self-help. In line with the code above, we then set

```
delta[21,2] <- 0
delta[21,3] <- d[21,3] - d[21,2]
delta[21,4] <- d[21,4] - d[21,2]
```

because `t[21,1]` = 2, `t[21,2]` = 3 and `t[21,3]` = 4 (cfr. with the code above describing the general definition of the variables `delta[s,t[s,a]]`). Again, the nested index notation allows us to use a single double loop to model all the possible cases.

The log ORs `d` are then defined as follows.

```
d[1] <- 0  # log odds ratio compared to treatment 1 (e.g. placebo)
for (i in 2:NT) {
    d[i] ~ dnorm(0, 0.0001)
}
```

This again sets one reference category, which in this case is associated with intervention 1 (No intervention), by setting the corresponding log OR to 0. Then we assign a vague prior to all the other log ORs (the interventions labelled from 2 to `NT` = 4), using a Normal with mean equal to 0 and a very small precision.

The next bit of the model code constructs the ORs for all potential treatment comparisons.

```
## odds ratios for all treatment comparisons
for (c in 1:(NT-1)) {
  for (k in (c+1):NT)  {
    or[c,k] <- exp(d[c] - d[k])
    or[k,c] <- 1/or[c,k]
  }
}
```

Again, there is some clever coding to use loops and compactly write down *all* the possible pairwise comparisons. Notice that the variables `d` define the log ORs and thus in order to obtain the OR on the natural scale, we need to exponentiate the difference between any two values. The line `or[k,c] <- 1/or[c,k]` uses the fact that ORs for the comparison between two generic interventions k and c are simply the reciprocal of the ORs for the comparison between c and k.

Finally, the model code has some additional definitions for other useful variables.

```
## Log odds of quitting successfully under no intervention (from published data)
alpha ~ dnorm(-2.6, 6.925208) # = SD 0.38
## Absolute probability of quitting successfully under each intervention
for (i in 1:NT) {
    logit(pq[i]) <- alpha + d[i]
}

## Life years gained by quitting
L ~ dnorm(15, 0.0625) # SD=4
```

Firstly, we define a model for the absolute probability of quitting smoking under each intervention. We do on the logit scale, by defining the baseline value $\alpha \sim \text{Normal}(\mu_\alpha, \sigma_\alpha)$, to which we add the incremental effect of each treatment. Notice that because we set `d[1]` = 0, then `alpha` is equal to `logit(pq[1])`, which is the absolute "success rate" for No intervention. This is incremented by the value of the log OR for each active treatment (against the reference, i.e. No intervention). Notice also that we use an informative prior distribution to model the parameter α. We have information suggesting that a random smoker who is not undergoing any active treatment has an average chance of quitting smoking of about 7%, ranging between around 2% to 13.8%. We can map this information into a suitable prior on the logit scale by setting

$$\mu_\alpha = \text{logit}(0.07) = \log\left(\frac{0.07}{1-0.07}\right) = \log\left(\frac{0.07}{0.93}\right) \approx -2.6.$$

We can also use the fact that

$$\sigma_\alpha \approx \frac{\text{logit}(0.138) - \text{logit}(0.07)}{1.96} \approx 0.38$$

— this is reasonable if we assume some sort of symmetry in the interval estimate, whereby the upper extreme is 1.96 standard deviations away from the central point, which implies that

$$\text{logit}(0.07) + 1.96\sigma_\alpha \approx \text{logit}(0.138).$$

Of course, we need to include in the `OpenBUGS` model the precision, i.e. $1/(0.38)^2 = 6.925208$.

The model for the life years gained by quitting smoking is constructed in a similar way: our best estimate is a gain of between 7 and 22 extra years, with an average of 15, which we turn into a Normal distribution with mean 15 and standard deviation of 4 (i.e. precision of $1/16 = 0.0625$).

The model is then run from `R`.

```r
library(R2OpenBUGS)

### Initial values
inits <- list(list(mu=rep(0,24), d=c(NA,0,0,0)),
              list(mu=rep(-1,24), d=c(NA,1,1,1)))

### Pilot run with no burn-in, to illustrate convergence using traceplots
res0 <- bugs(model="smokefix_model.txt", data=smoke.list, inits=inits,
             parameters=c("d"),
             n.chains=2, n.burnin=0, n.iter=10000
)
```

Here, we first define the initial values creating a suitable list made by two "sub-lists" — this implies we are prepared to run the model using two parallel chains. We initialise the variables `mu` and `d` — notice that because the first element of the vector `d` is in fact fixed at 0, we cannot initialise it. We overcome this issue by creating a vector of initial values, the first of which is set to `NA`.

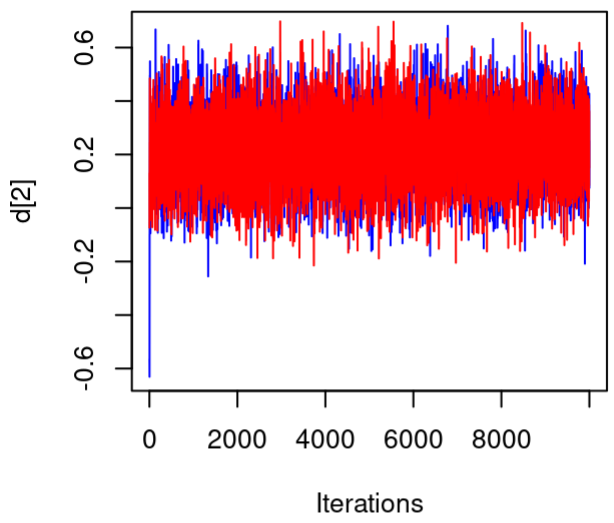
In addition, this time we run the model with no burn-in, to explore convergence in more details than we've done so far. The code below uses the results of the `OpenBUGS` model as stored in the `R` object `res0` to produce traceplots for the only variable monitored (the vector `d`). Notice in particular that we can use the object `res$sims.array`, which (as the name suggests) is an array of dimension `Number of iterations stored` × `Number of chains run` × `Number of parameters monitored`. In this case, the number of parameters is equal to 4, because there are 3 "active" elements in `d` (since `d[1]` is set to 0 and thus is not technically a random variable, in `OpenBUGS`), plus the model deviance, which `OpenBUGS` computes by default. You can check this by simply printing the summary statistics for your model

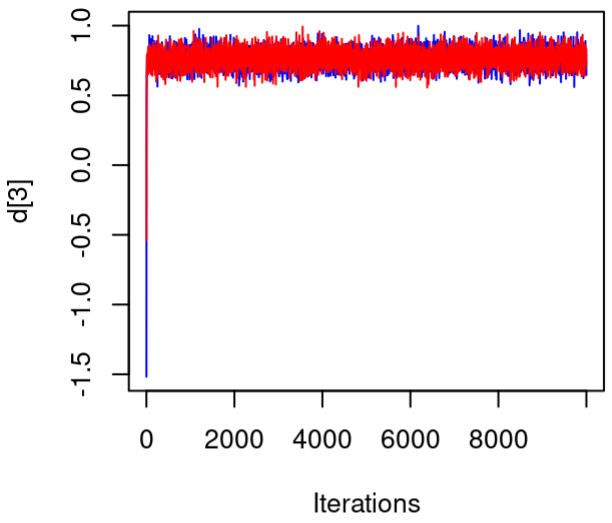
```r
library(R2OpenBUGS)
print(res0,digits=3)
```

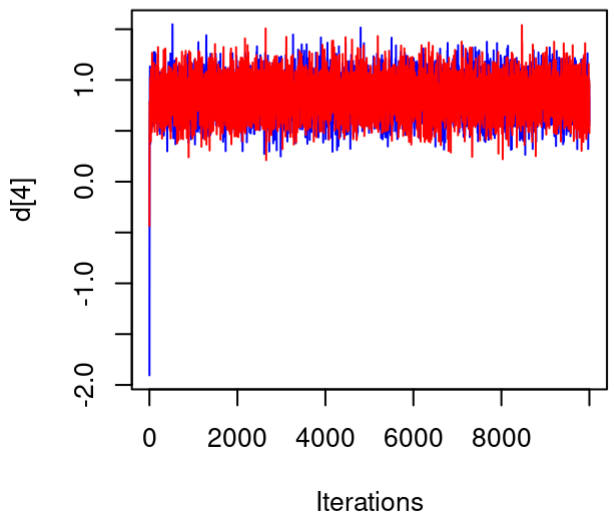
```
Inference for Bugs model at "smokefix_model.txt",
Current: 2 chains, each with 10000 iterations (first 0 discarded)
Cumulative: n.sims = 20000 iterations saved
            mean     sd    2.5%     25%     50%     75%   97.5%  Rhat n.eff
d[2]       0.225  0.126  -0.020   0.140   0.227   0.309   0.473 1.001 20000
d[3]       0.765  0.063   0.650   0.727   0.766   0.805   0.879 1.001 20000
d[4]       0.839  0.178   0.499   0.719   0.839   0.959   1.183 1.001 20000
deviance 494.927 16.902 482.300 489.500 494.100 499.300 510.700 1.031 20000

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 = Dbar-Dhat)
pD = 27.220 and DIC = 522.100
DIC is an estimate of expected predictive error (lower deviance is better).
```

The actual traceplot is produce using the built-in functions `plot` and `points` as below.

```r
# Splits the graphical output into a 2-by-2 panel (side-by-side graphs)
par(mfrow=c(2,2))
# First graph
plot(res0$sims.array[,1,1],t="l",xlab="Iterations",ylab="d[2]",col="blue")
points(res0$sims.array[,2,1],t="l",col="red")
# Second graph
plot(res0$sims.array[,1,2],t="l",xlab="Iterations",ylab="d[3]",col="blue")
points(res0$sims.array[,2,2],t="l",col="red")
# Third graph
plot(res0$sims.array[,1,3],t="l",xlab="Iterations",ylab="d[4]",col="blue")
points(res0$sims.array[,2,3],t="l",col="red")
```







As is possible to see, for all the three important parameters, convergence does not seem an issue and in fact the two chains seem to mix up almost immediately, despite being seen to start from rather different points (cfr. the red and blue lines). Notice that this strategy is not an absolute requirement! We can monitor all the relevant parameters and run the model for a large number of iterations in the first place. But, especially when there are many parameters, this course of action may be beneficial, because we are not stuck waiting for `OpenBUGS` to finish the simulations for a long time, before we can even assess how the model is working in terms of convergence.

At this point, we can monitor all the model parameters (including `L` and `pq`) and re-run the model to produce the relevant estimates.

```
res <- bugs(model="smokefix_model.txt", data=smoke.list, inits=inits,
            parameters=c("d","L","pq"),
            n.chains=2, n.burnin=1000, n.iter=5000)

# Show summary statistics
print(res,digits=3)
```

```
Inference for Bugs model at "smokefix_model.txt",
Current: 2 chains, each with 5000 iterations (first 1000 discarded)
Cumulative: n.sims = 8000 iterations saved
            mean    sd    2.5%     25%     50%     75%   97.5%  Rhat n.eff
d[2]       0.224 0.126  -0.023   0.139   0.225   0.309   0.470 1.001  8000
d[3]       0.764 0.058   0.649   0.725   0.765   0.804   0.877 1.004   420
d[4]       0.840 0.176   0.505   0.719   0.838   0.958   1.189 1.001  8000
L         14.895 4.009   6.947  12.250  14.940  17.550  22.750 1.001  7300
pq[1]      0.073 0.026   0.034   0.054   0.069   0.087   0.136 1.001  2700
pq[2]      0.090 0.034   0.040   0.065   0.085   0.108   0.171 1.002  2300
pq[3]      0.143 0.048   0.069   0.108   0.137   0.171   0.254 1.001  7500
pq[4]      0.154 0.055   0.070   0.114   0.146   0.186   0.284 1.001  4900
deviance 494.653 7.175 482.400 489.500 494.100 499.200 510.102 1.001  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 = Dbar-Dhat)
pD = 26.940 and DIC = 521.600
DIC is an estimate of expected predictive error (lower deviance is better).
```

Again, convergence is clearly reached (cfr. `Rhat` and `n.eff`). We could proceed with further analyses as well as with building the cost-effectiveness model, but we defer this to after we've run the random effects model (cfr. lecture slides for evidence of heterogeneity in individual studies; we can replicate the analysis monitoring the nodes `delta`, which are the study- and treatment-specific log ORs).

Random effects NMA

The model code is fairly similar to the one discussed above for the fixed effects NMA. The only difference, really, is in how the study- and treatment-specific log ORs `delta` are modelled. In this case, we consider a simple specification, characterised by the following code lines

```
delta[s,t[s,a]] ~ dnorm(md[s,t[s,a]],taud[s,t[s,a]])

# random effects means
md[s,t[s,a]] <- d[t[s,a]] - d[t[s,1]]

## random effects 1/variance constrained to be the same for every comparison
taud[s,t[s,a]] <- tau

# model for the standard deviation of the random effects
sd ~ dunif(0, 10)
# rescaling to the precision
tau <- 1/pow(sd, 2)
```

— notice that we can include more complexity, for instance by modelling the precision as dependent on the studies or the treatments, as well as by considering a more structured model accounting for correlation within trials with 3 arms or more (but we do not do this here!).

The model is run again using the following code.

```
res2 <- bugs(model="smokere_model.txt", data=smoke.list, inits=inits,
             parameters=c("d", "sd", "pq", "L"),
             n.chains=2, n.burnin=1000, n.iter=20000
)
print(res2,digits=3)
```

```
Inference for Bugs model at "smokere_model.txt",
Current: 2 chains, each with 20000 iterations (first 1000 discarded)
Cumulative: n.sims = 38000 iterations saved
            mean     sd    2.5%     25%     50%     75%   97.5%  Rhat n.eff
d[2]       0.519  0.386  -0.230   0.267   0.514   0.766   1.305 1.001  6300
d[3]       0.810  0.232   0.374   0.656   0.805   0.956   1.289 1.001  5900
d[4]       1.169  0.455   0.304   0.867   1.156   1.460   2.098 1.001 38000
sd         0.820  0.183   0.533   0.690   0.795   0.923   1.248 1.001 38000
pq[1]      0.073  0.026   0.034   0.054   0.069   0.088   0.136 1.001 38000
pq[2]      0.122  0.059   0.042   0.080   0.111   0.152   0.269 1.001  6500
pq[3]      0.152  0.057   0.065   0.110   0.143   0.184   0.286 1.001  8400
pq[4]      0.208  0.096   0.071   0.138   0.191   0.262   0.437 1.001 38000
L         14.989  3.993   7.109  12.310  15.000  17.670  22.810 1.001 38000
deviance 281.823 10.073 263.900 274.700 281.200 288.200 303.500 1.001 38000

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 = Dbar-Dhat)
pD = 44.870 and DIC = 326.700
DIC is an estimate of expected predictive error (lower deviance is better).
```

Again, the model seems to do well in terms of convergence, although autocorrelation is possibly more of a concern (check the values of `n.eff`, which are slightly smaller than the nominal sample size of 38000). This is not uncommon with hierarchical/random effect models.

In terms of comparison with the results of the fixed effects version, there is generally an increase in the value of the mean for the log ORs `d`, coupled with larger uncertainty. On the other hand, the absolute probabilities of quitting `pq` are rather stable. The estimate for `L` does not change — but this is not surprising, as this node is modelled independently on the other variables in the code and based on an informative prior, which is not updated by any data. So, simply changing parts of the model that are effectively disconnected by the one in which we model `L` is not changing our estimates for this node.

We can also complete the model to include the cost-effectiveness component. We do this by firstly defining a vector of unit costs associated with each intervention. Here we assume that No intervention does not involve any cost for the public payer, while Self-help, Individual and Group counselling do have some costs. We then define the measures of effectiveness as the overall life years gained (`L`) weighted by the absolute probability of quitting smoking (`pq`) for each intervention. We build the variables `e` and `c` in the loop over the 4 interventions. Notably, in this case, we do not model the costs (although a variant of this model that does account for this is presented in the the BCEA Book).

```
### Cost-effectiveness analysis
library(BCEA)
```

```
##
## Attaching package: 'BCEA'
```

```
## The following object is masked from 'package:graphics':
##
##     contour
```

```
unit.cost <- c(0,200,6000,600)
ints <- c("No contact","Self help","Individual counselling","Group counselling")
e <- c <- matrix(NA,res2$n.sims,4)
# MCMC sample from distribution of life-years gained by quitting
L <- res2$sims.list$L
# ...and from distributions of probability of quitting for each of 4 interventions
pq <- res2$sims.list$pq

for (t in 1:4) {
    e[,t] <- L*pq[,t]
    c[,t] <- unit.cost[t]
}
colnames(e) <- colnames(c) <- ints
```

Finally, we can run `BCEA` to post-process the model output and produce the relevant summaries, e.g. summaries or the cost-effectiveness plane.

```
m <- bcea(e,c,interventions=ints,Kmax=1000,ref=4)
summary(m)
```

```
## NB: k (wtp) is defined in the interval [0 - 1000]
```

```
##
```

```
##
## Cost-effectiveness analysis summary
##
## Reference intervention:  Group counselling
## Comparator intervention(s): No contact
##                           : Self help
##                           : Individual counselling
##
## Optimal decision: choose No contact for k < 274
##                           Self help for 274 <= k < 310
##                           Group counselling for k >= 310
##
##
## Analysis for willingness to pay parameter k = 1000
##
##                       Expected utility
## No contact                      1096.7
## Self help                       1629.9
## Individual counselling         -3727.5
## Group counselling               2523.5
##
##                                            EIB    CEAC     ICER
## Group counselling vs No contact            1426.85 0.89945   296.03
## Group counselling vs Self help              893.65 0.78947   309.20
## Group counselling vs Individual counselling 6251.05 1.00000 -6345.08
##
## Optimal intervention (max expected utility) for k = 1000: Group counselling
##
## EVPI 97.442
```

Notice that, because this model involves multiple comparisons, the default output for the `plot` function in `BCEA` is not entirely satisfactory. There are ways in which we can modify this default behaviour to improve the look of the pictures (see the help for `BCEA` as well as the BCEA Book).

```
plot(m)
```