Practical 7. PSA to structural uncertainty — SOLUTIONS

Lecture 7 PDF version

The first thing we do is loading and analysing the "base-case" model, which is stored in the object statins_base.Rdata and the "robust" version of the model, stored in statins_HC.Rdata.

```
library(BCEA)
library(R2OpenBUGS)
load("statins_base.Rdata")
load("statins_HC.Rdata")
```

We can use the R function print to visualise the output for the two models, for example as in the following.

```
print(statins_base)
```

The output to this call is a long list of summary statistics — it is also possible to visualise an excerpt by using code such as the following

```
head(statins_base$summary[,c("mean","sd","2.5%","97.5%","Rhat","n.eff")],n=15)
```

```
                 mean       sd      2.5%     97.5%     Rhat n.eff
cost.hosp[1] 238.7010 137.0428  91.68029  482.0066 1.001533   980
cost.hosp[2] 315.6340 168.7579 124.20521  668.4667 1.002554  1000
cost.hosp[3] 523.0695 451.0165 144.02705 1357.9918 1.002820  1000
cost.hosp[4] 424.9861 232.1917 170.08732  877.9484 1.001743   980
cost.hosp[5] 305.1978 172.3249 120.79156  656.3935 1.002747   550
cost.hosp[6] 301.1282 163.4569 121.85935  618.0017 1.001342  1000
cost.stat[1] 480.8821 289.1621 137.50194 1232.7586 1.004167   360
cost.stat[2] 350.0194 201.6103 103.42871  871.5393 1.000252  1000
cost.stat[3] 166.6851 125.6502  34.24849  498.3156 1.000219  1000
cost.stat[4] 305.4061 261.7113  47.69648 1008.5598 1.004566  1000
cost.stat[5] 346.9371 209.6277 103.21293  880.0139 1.006177   400
cost.stat[6] 165.0717 129.7352  35.28509  526.9310 1.000074  1000
cost.tot[1]  719.5831 324.8498 309.13357 1526.9058 1.005610   270
cost.tot[2]  665.6534 265.4862 331.33786 1357.5038 1.001748  1000
cost.tot[3]  689.7546 467.3877 242.54283 1629.7467 1.002190  1000
```

which produces the first n=15 rows (i.e. parameters) for the whole summary table. In particular, we only select the columns headed as "mean", "sd", etc. (we do so to exclude additional quantiles that are automatically stored in the object statins_base$summary). We should make sure that the models have all converged and that autocorrelation is not an issue (by e.g. analysing the $\hat{R}$ and n_{eff} statistics).

We can already check the DIC associated with each of the two models, to get some ideas of which one will be given most weight in the structural PSA. We can do so by using the following command.

```
# Displays the DIC for the two models
c(statins_base$DIC,statins_HC$DIC)
```

```
[1] 2233.875 2225.988
```

As is easy to see, the "robust" model is associated with a relatively lower DIC (by over 10 points).

We can now move on and use the results from the two Bayesian models as inputs to BCEA. The objects statins_base$sims.list and statins_HC$sims.list contain the simulated values for all the model parameters monitored in list format. We can follow the script and use the code

```r
# Defines the intervention labels
interventions <- c("Atorvastatin","Fluvastatin","Lovastatin","Pravastatin",
                   "Rosuvastatin","Simvastatin")
# BCEA object with the economic analysis of the "base case" model
m1 <- bcea(statins_base$sims.list$effect,statins_base$sims.list$cost.tot,
           ref=1,interventions=interventions)
# BCEA object with the economic analysis of the Half-Cauchy model
m2 <- bcea(statins_HC$sims.list$effect,statins_HC$sims.list$cost.tot,
           ref=1,interventions=interventions)
```

to first define a vector of intervention labels and then apply the function bcea to the suitable variables of effects and costs in the two models.

The two objects m1 and m2 can be post-processed as any BCEA objects (e.g. using plot or print methods). But in addition to this, we can also combine them to perform the PSA to structural assumptions. To do so, we need to manipulate the original objects in a suitable way. Firstly, we need to create a list of models, which we can simply do using the following command.

```r
# Combines the BUGS models
models <- list(statins_base,statins_HC)
```

the newly created object contains the information from the two BUGS models. We can also create suitable lists in which we store the relevant variables of effectiveness and costs from the two models, for example using code such as the following.

```r
# Creates the variables of effectiveness and costs
effects <- list(statins_base$sims.list$effect, statins_HC$sims.list$effect)
costs <- list(statins_base$sims.list$cost.tot, statins_HC$sims.list$cost.tot)
```

Finally, we can feed these inputs to the BCEA function struct.psa as in the following.

```r
# Finally uses BCEA to perform the structural PSA to consider the base and HC models
m3 <- struct.psa(models,effects,costs,ref=1,interventions=interventions)
```

struct.psa takes as basic arguments three lists, containing the BUGS models, the list of effects and the list of costs simulations and combines them to compute the model weights (based on the DICs).

The object m3 is a list, which contains 3 elements:

```r
lapply(m3,function(x) list(class(x),names(x)))
```

```
$he
$he[[1]]
[1] "bcea" "list"

$he[[2]]
 [1] "n_sim"         "n_comparators" "n_comparisons" "delta_e"
 [5] "delta_c"       "ICER"          "Kmax"          "k"
 [9] "ceac"          "ib"            "eib"           "kstar"
[13] "best"          "U"             "vi"            "Ustar"
[17] "ol"            "evi"           "ref"           "comp"
[21] "step"          "interventions" "e"             "c"


$w
$w[[1]]
[1] "numeric"

$w[[2]]
NULL


$DIC
$DIC[[1]]
[1] "numeric"

$DIC[[2]]
NULL
```

The R function `lapply` applies iteratively the function `class` and `names` to each element of the object `m3`.

As is possible to see, the first one, named `he`, is an object in the class `BCEA` and so it contains the usual elements that such objects do (e.g. `n.sim`, `n.comparators`, etc). The elements `w` and `DIC` are numeric vectors and include the weights and the value of the DIC associated with each individual models. We can visualise for example the weights by using the command,

```
print(m3$w)
```

```
[1] 0.01901127 0.98098873
```

which indicates that the second model (the "robust" Half-Cauchy) is given a weight of over 98%. This is consistent with the fact that its DIC is lower than the one for the base case model, which in turn indicates better fit.

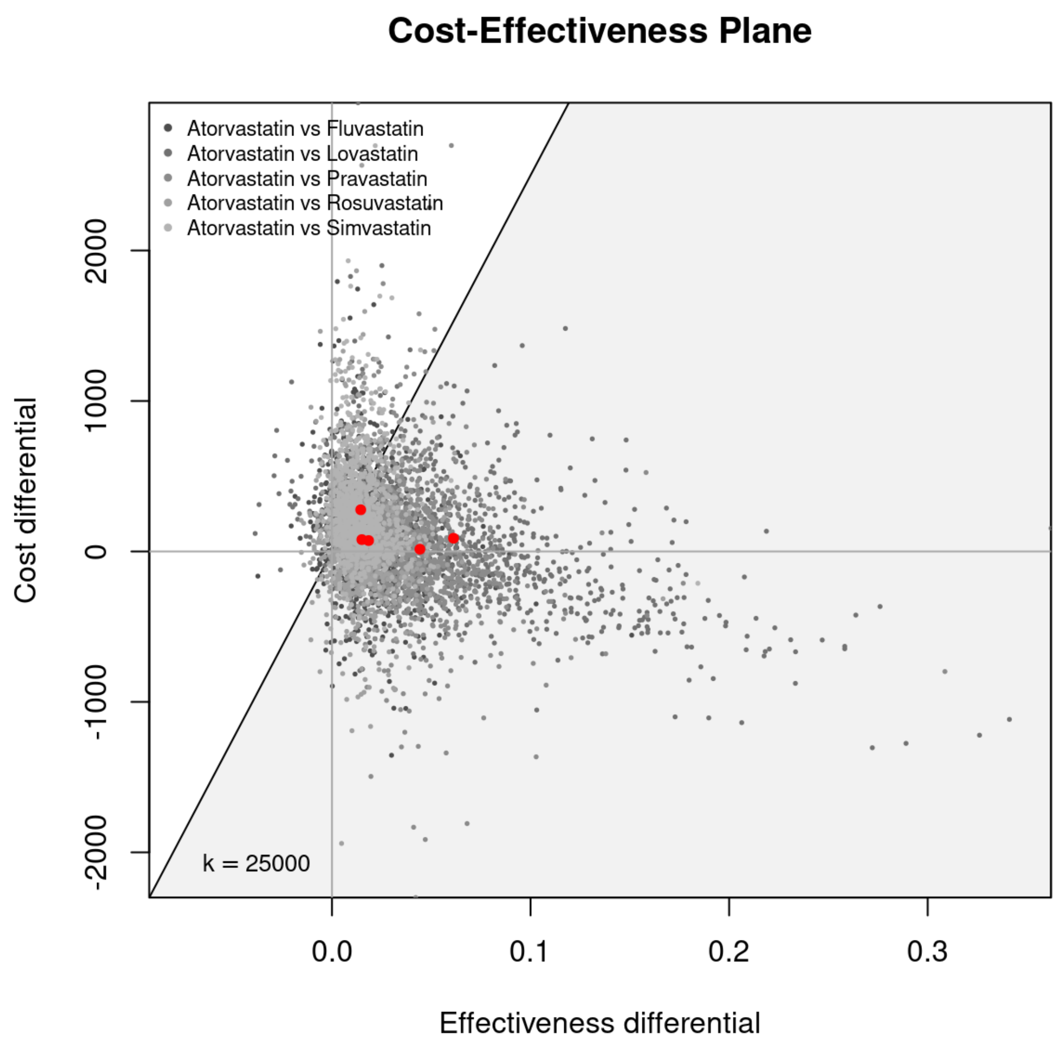
We can also use all the methods implemented for `BCEA` objects to analyse and visualise the output of the model averaging. For example, we can summarise the cost-effectiveness analysis by typing

```
summary(m3$he)
```

```

Cost-effectiveness analysis summary 

Reference intervention:  Atorvastatin
Comparator intervention(s): Fluvastatin
                          : Lovastatin
                          : Pravastatin
                          : Rosuvastatin
                          : Simvastatin

Optimal decision: choose Simvastatin for k <  and  for k >=


Analysis for willingness to pay parameter k = 25000

             Expected utility
Atorvastatin           22823
Fluvastatin            22435
Lovastatin             21381
Pravastatin            21731
Rosuvastatin           22526
Simvastatin            22738

                                  EIB  CEAC     ICER
Atorvastatin vs Fluvastatin   388.695 0.768  3919.60
Atorvastatin vs Lovastatin   1442.154 0.843  1432.44
Atorvastatin vs Pravastatin  1092.152 0.958   336.72
Atorvastatin vs Rosuvastatin  297.304 0.738  5227.07
Atorvastatin vs Simvastatin    85.002 0.594 19129.49

Optimal intervention (max expected utility) for k = 25000: Atorvastatin

EVPI 193.22
```

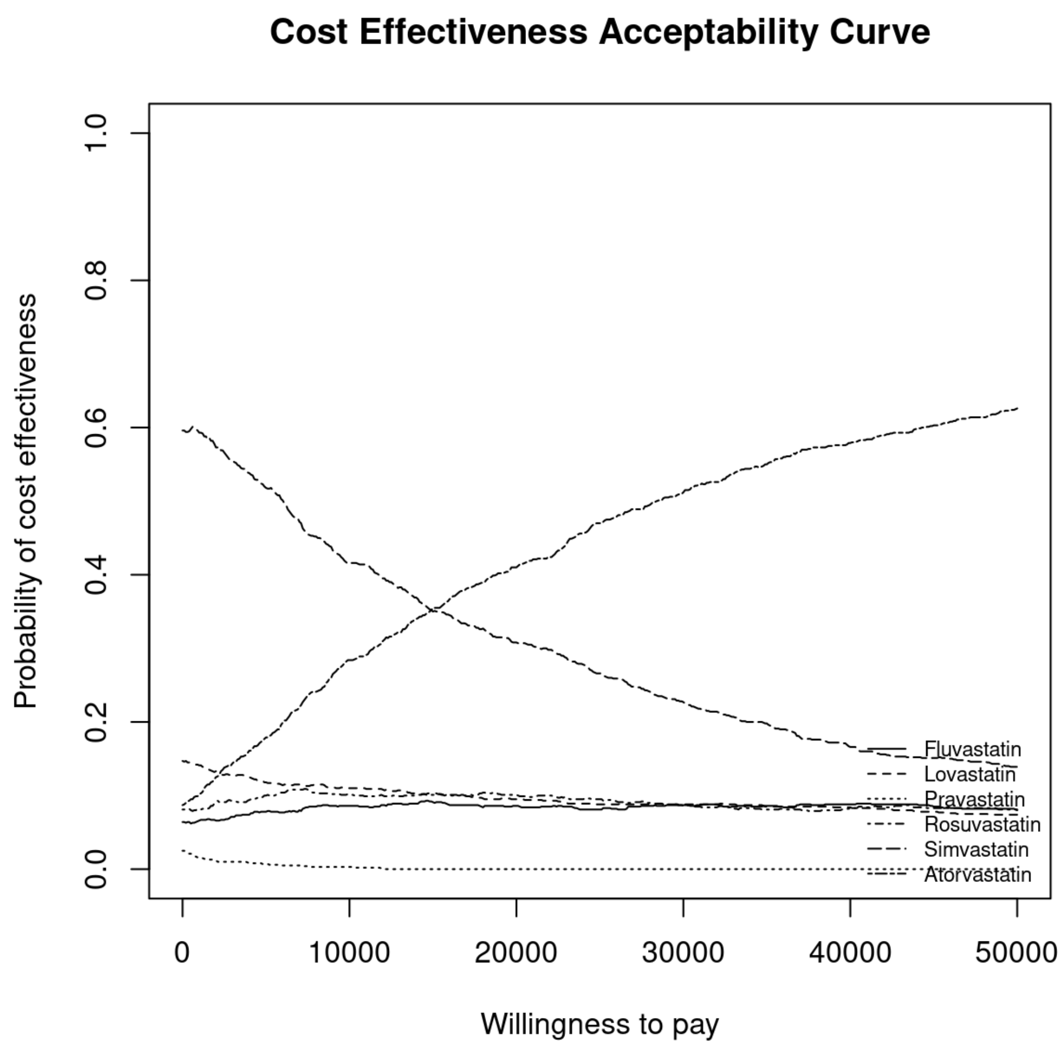
and we could plot the cost-effectiveness plane with the following command.

```
ceplane.plot(m3$he)
```



or generate multiple treatments comparison cost-effectiveness acceptability curves with the following commands.

```
m3.multi<-multi.ce(m3$he)
ceac.plot(m3.multi)
```



Notice that because *in this particular case* one of the models is effectively given an almost 100% weight, the model average will resemble it almost identically.

© 2022-2024 · Based on Wowchemy (version 5.3.0) for Hugo · ***Latest update: 12 Jun 2022***