Working with the R package BCEA

Gianluca Baio

Department of Statistical Science | University College London

g.baio@ucl.ac.uk
http://www.statistica.it/gianluca/
https://egon.stats.ucl.ac.uk/research/statistics-health-economics/
https://github.com/giabaio
https://github.com/StatisticsHealthEconomics
@gianlubaio

ISPOR Short courses, Internetville

17 June 2021

An R package for (Bayesian) cost-effectiveness analysis

What is BCEA not?

- BCEA is **not** a package to automatically run a Bayesian analysis
  - It cannot build the health economic model for you
  - It does not prepare the data to be used in the model
  - It does not automatically run the MCMC simulations
  - It does not choose the prior distributions for you

So what *is* it then?

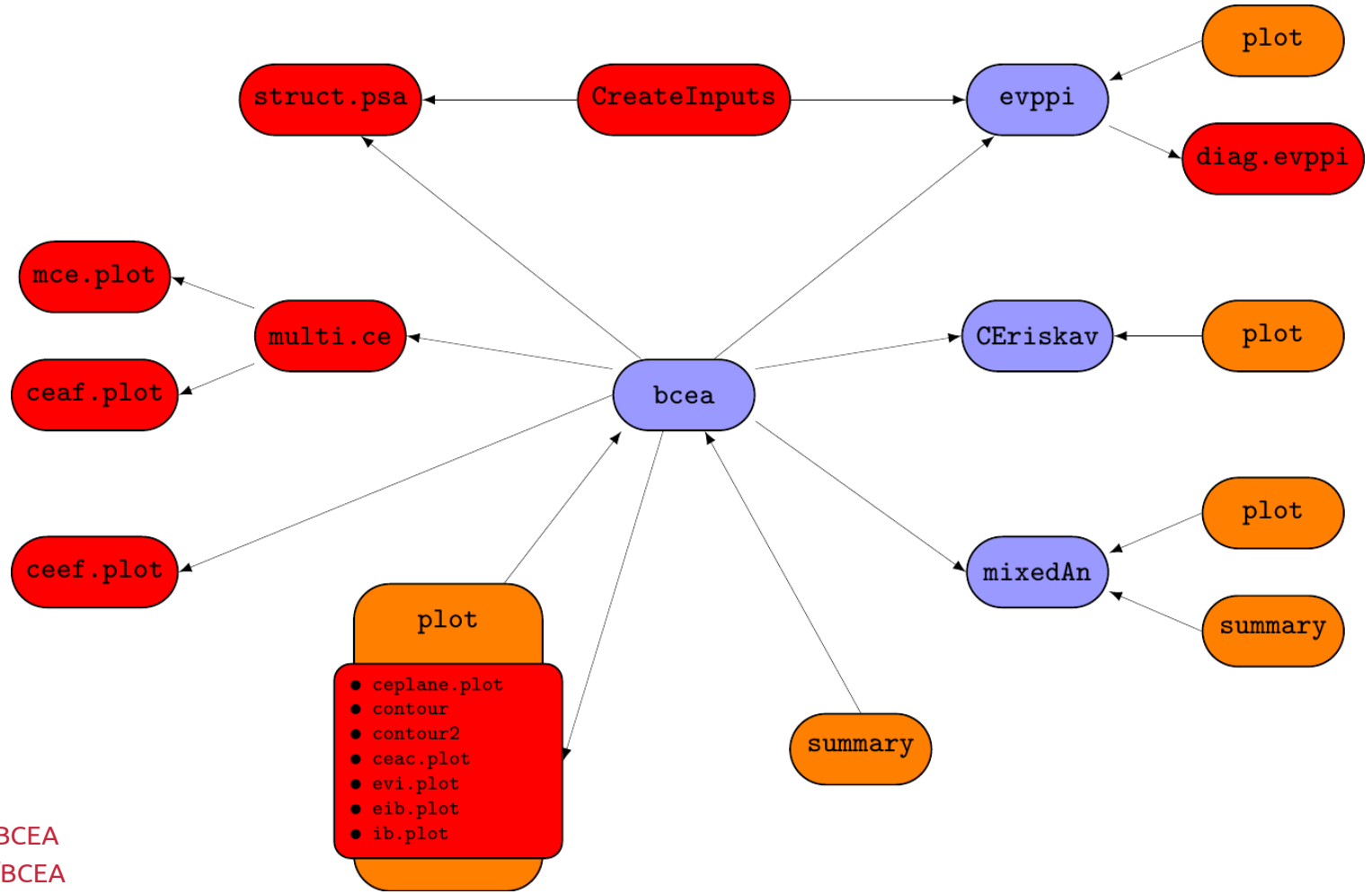
- BCEA provides a set of specific functions to systematically post-process the output of a Bayesian health economic model
- Uses R http://cran.r-project.org/
  - Very good at interacting with standard MCMC software
  - BUGS: www.mrc-bsu.cam.ac.uk/bugs/winbugs/contents.shtml
  - JAGS: www.mcmc-jags.sourceforge.net/
  - rstan: https://mc-stan.org/users/interfaces/rstan
  - **Free** and there is a very large community of contributors
  - Specifically designed for statistical analysis and has very good graphical capabilities

A R package for (Bayesian) cost-effectiveness analysis

BCEA and its use directly in R are designed with these objectives in mind

1. Checking the model assumptions
   - Do we mean what we mean (eg in terms of PSA simulations)?...
   - Simulation error (especially, **but not only**, for a Bayesian approach)
2. Produce the base-case economic evaluation
   - What's the most cost-effective intervention, given current evidence?
   - Cost-effectiveness plane, Expected Incremental Benefit (as a function of k),...
3. Perform uncertainty analysis
   - Standard PSA (mandatory): Cost-effectiveness Plane, CEAC, ...
   - Fairly easy (but not always used): CEAF
   - More advanced/"too difficult" (rarely used): EVP(P)I/EVSI
4. Standardised reporting
   - Graphical tools (use **excellent** R facilities)

An R package for (Bayesian) cost-effectiveness analysis



www.statistica.it/gianluca/BCEA
https://github.com/giabaio/BCEA

Part I

Using BCEA to summarise outputs of an economic model

How does BCEA work?

- BCEA is available from CRAN
  – Current *stable* version: 2.3-1.1 (26 August 2019)
- But it is also under constant development in the GitHub repository
  – Current *stable* version: 2.3-2 (30 January 2020)
  – Current *development* version: 2.4-1 (5 February 2021) – **major** update (soon to be released!)

```
# Install BCEA (only required once and needs an internet connection!).

# You can either get the "official" version from CRAN
install.packages("BCEA")

# Or the development version from the GitHub repository
devtools::install_github("giabaio/BCEA")              # stable version (2.3-2)

devtools::install_github("giabaio/BCEA",ref="devel")   # development version (2.4-1)
```

NB: The beauty of the GitHub version is that it can be updated on the fly and be immediately available for users!

How does BCEA work?


```
library(dplyr)  # (Not necessary - helpful for data manipulation!)

library(BCEA)   # Then loads the package (so you can access its functions)
data(Vaccine)   # Loads an example dataset
```

- The "Vaccine" example is a fictional cost-effectiveness model for and influenza vaccine, based on evidence synthesis (and a real case)
- 2 treatment options ("Standard of care" vs "Vaccination") and overall 63 parameters
- Discussed in details in Baio et al, 2017 and Baio and Dawid, 2011
- In this case, PSA simulations obtained from a full Bayesian model, but could be done in a spreadsheet and imported into R

How does BCEA work?


```
# Creates a matrix with the underlying model simulations
inp = CreateInputs(vaccine, print.lincom=FALSE)          # NB: this option will be slightly different in 2.4-1!
# Visualise the output
inp$mat %>% as_tibble()                                  # "piping" ('%>%') and formatting nicely
```

```
# A tibble: 1,000 x 56
   Adverse.events Death.1.1. Death.2.1. Death.2.2. GP.1.1. GP.2.1. GP.2.2. Hospital.1.1. Hospital.2.1. Hospital.2.2. Infected.1.1
            <dbl>      <dbl>      <dbl>      <dbl>   <dbl>   <dbl>   <dbl>          <dbl>          <dbl>          <dbl>        <dbl
 1           1466          1          0          0    1664     958     230              0              1              0          599
 2           5329          1          1          0    1414     748     276              0              0              1          747
 3           5203          1          1          0     809     489      80              0              0              0          671
 4           2351          2          0          0    1761    1157     261              1              0              0          483
 5           8303          1          2          0    2472     964     432              1              1              0          474
 6           3607          1          1          0    2224    1342     260              1              0              0          493
 7           6304          4          1          1    3478    1107     591              2              1              0         1108
 8           4337          1          1          1    1483     799     189              0              0              0          386
 9           5482          0          0          0    1587     798     279              0              0              0          516
10           3125          2          2          0    2578    1681     243              0              0              0          726
# … with 990 more rows, and 33 more variables: Trt.1.2.2. <dbl>, Trt.2.2.2. <dbl>, beta.1. <dbl>, beta.2. <dbl>, beta.3. <dbl>, b
#   pi.1.2. <dbl>, psi.1. <dbl>, psi.2. <dbl>, psi.3. <dbl>, psi.4. <dbl>, psi.5. <dbl>, psi.6. <dbl>, psi.7. <dbl>, psi.8. <dbl>
```

How does BCEA work?


```
# Defines the number of simulations considered
n.sims=inp$mat %>% nrow()                    # applies the function 'nrow' (number of rows) to the object 'inp$mat'

                                             # NB: in R 4.1.0, can also use 'native' pipe ('|>')
                                             # (probably a bit quicker, but in most cases, may be immaterial...)

# Aggregates the model inputs to compute (e,c)
QALYs.inf = QALYs.pne <- QALYs.hosp <- QALYs.adv <- QALYs.death <- matrix(0,n.sims,2)
for (t in 1:2) {
  QALYs.inf[,t] = ((Infected[,t,1] + Infected[,t,2])*omega[,1]/365)/N
  QALYs.pne[,t] = ((Pneumonia[,t,1] + Pneumonia[,t,2])*omega[,4]/365)/N
  QALYs.hosp[,t] = ((Hospital[,t,1] + Hospital[,t,2])*omega[,5]/365)/N
  QALYs.death[,t] = ((Death[,t,1] + Death[,t,2])*omega[,6])/N
}
QALYs.adv[,2] = (Adverse.events*omega[,7]/365)/N
e = -(QALYs.inf + QALYs.pne + QALYs.adv + QALYs.hosp + QALYs.death) + ...
```

- NB: The data stored in the `Vaccine` object (built-in in `BCEA`) already contains the objects `e`, `c` that can be used to run the decision analysis...
- So, this step is actually not needed (but documented in 🕮 Baio et al, 2017)

How does BCEA work?


```
cbind(e,c) %>% as_tibble(.name_repair="universal")     # ensures that the columns are named
```

```
# A tibble: 1,000 x 4
   Status.Quo...1 Vaccination...2 Status.Quo...3 Vaccination...4
            <dbl>           <dbl>          <dbl>           <dbl>
 1      -0.00105        -0.000899          10.4            16.3
 2      -0.000884       -0.000732           5.83            9.37
 3      -0.000890       -0.000698           5.78           15.9
 4      -0.00164        -0.00114           12.2            18.7
 5      -0.00135        -0.000957           9.79           16.5
 6      -0.00143        -0.000936           6.56            9.69
 7      -0.000960       -0.00105            8.45           11.3
 8      -0.00181        -0.00139            6.76            9.99
 9      -0.000842       -0.000556           3.60           10.1
10      -0.00168        -0.00105            4.09           11.0
# … with 990 more rows
```

- These calculations *can* be done also in a spreadsheet (nothing more than algebra, **once you have the simulations**)

How does BCEA work?

- At this point, we are ready to call the function `bcea` that runs the economic analysis, for example something like

```
treats = c("Status quo","Vaccination")
m = bcea(e=e,c=c,ref=2,interventions=treats,Kmax=50000)
```

- The inputs to the function are
  - `e`: a **matrix** containing the simulations for the clinical benefits (that is $n_{\text{sim}} \times n_{\text{int}}$ values)
  - `c`: a **matrix** containing the simulations for the costs (that is $n_{\text{sim}} \times n_{\text{int}}$ values)
  - `ref`: an indication of which intervention is to be taken as reference (default: the intervention in the first column of `e` or `c`)
  - `interventions`: a vector of labels for the interventions being compared
  - `Kmax`: the maximum value of k, the parameter of willingness to pay

- The output is an object `m` containing several elements

```
names(m)
```

```
 [1] "n.sim"         "n.comparators" "n.comparisons" "delta.e"       "delta.c"       "ICER"          "Kmax"          "k"
[19] "interventions" "ref"           "comp"          "step"          "e"             "c"
```

How does BCEA work?

Can visualise the output in various formats (tables/graphs)

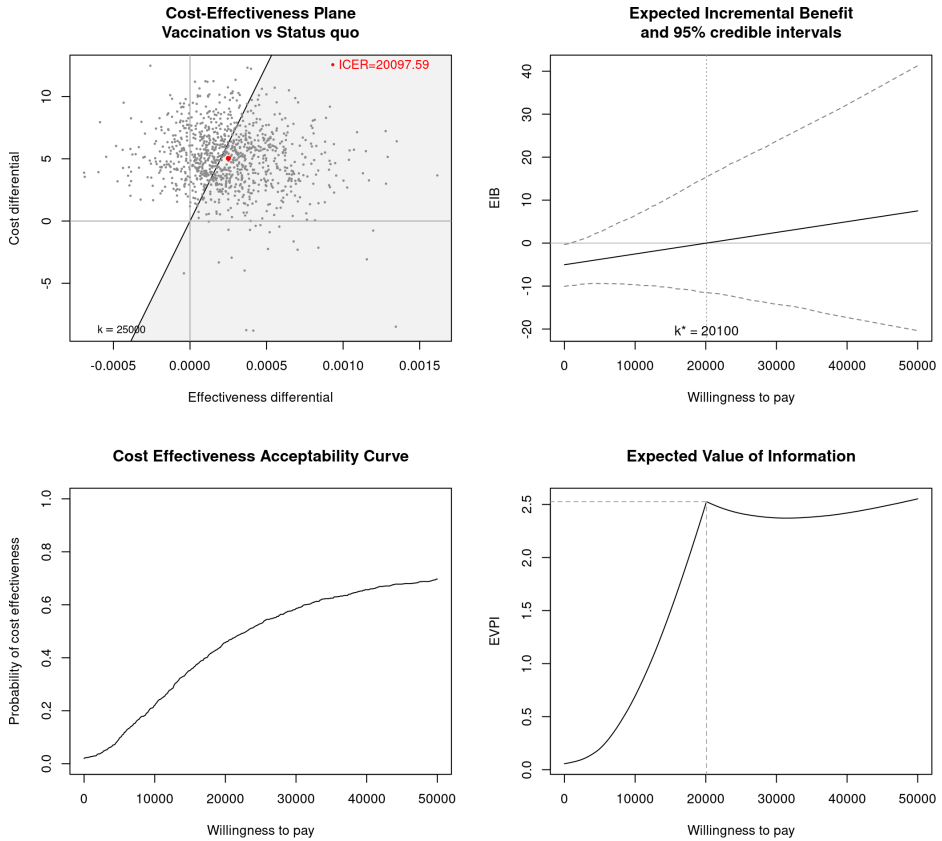
```
# The 'summary' "method" produces a tabular output
summary(m)
```

```
Cost-effectiveness analysis summary

Reference intervention:  Vaccination
Comparator intervention: Status quo

Optimal decision: choose Status quo for k<20100 and Vaccination for k>=20100


Analysis for willingness to pay parameter k = 25000

            Expected utility
Status quo           -36.054
Vaccination          -34.826

                            EIB  CEAC  ICER
Vaccination vs Status quo 1.2284 0.529 20098

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

EVPI 2.4145
```

How does BCEA work?

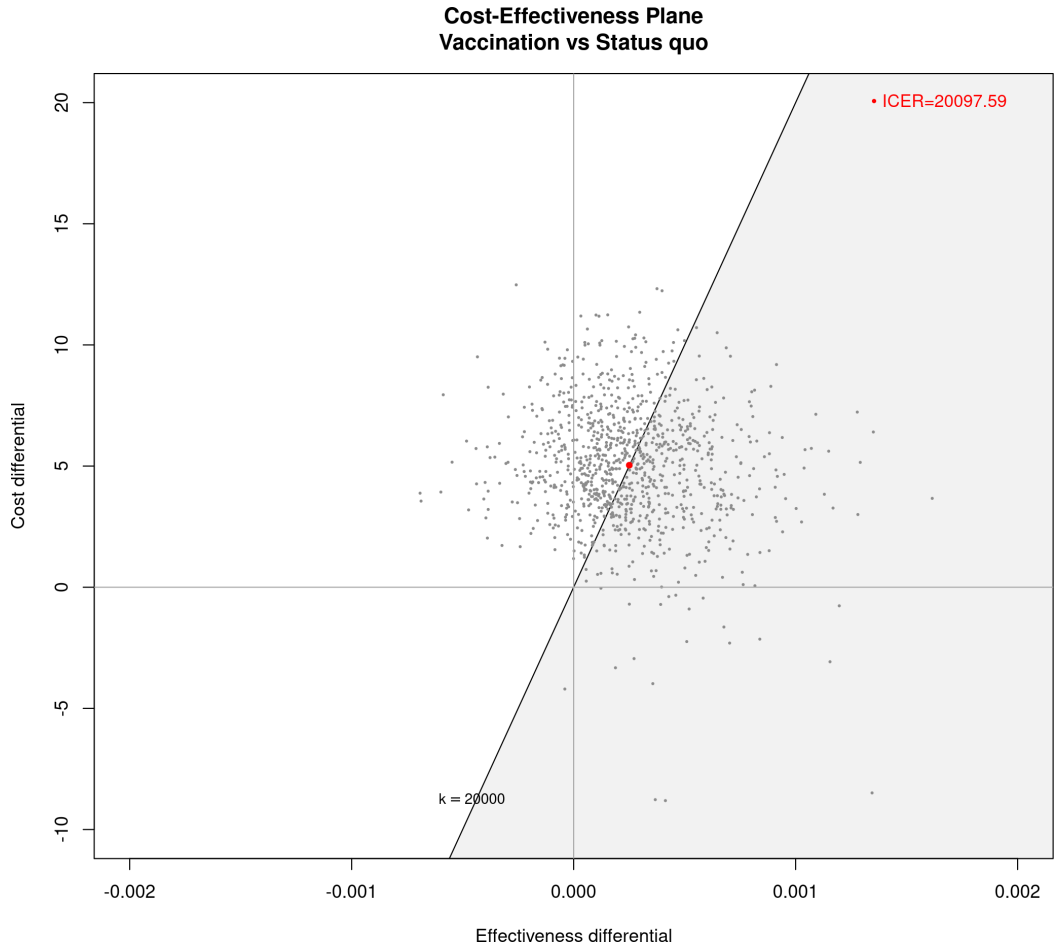
Can visualise the output in various formats (tables/graphs)

```
# The 'plot' "method" produces a *specific* version of graphical output
plot(m)
```



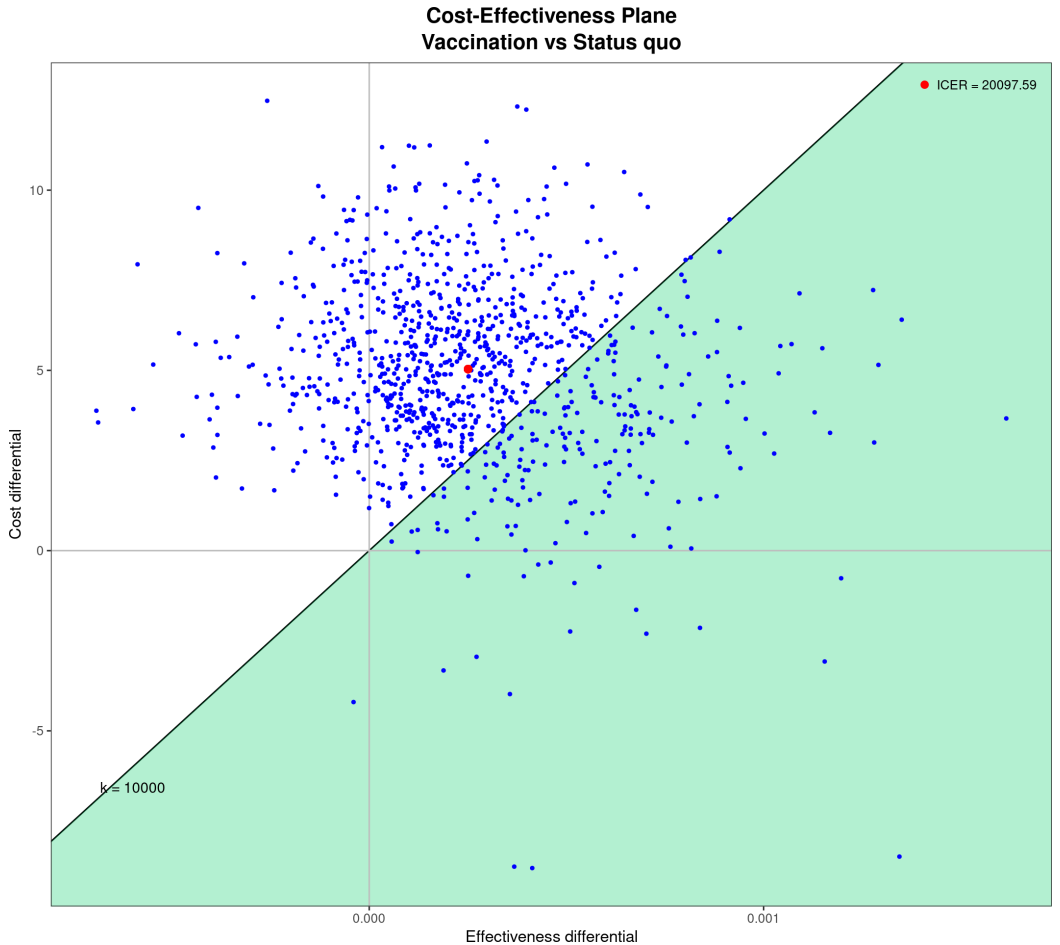
How does BCEA work?

```
ceplane.plot(m,wtp=20000,xlim=c(-.002,.002),ylim=c(-10,20))
```



How does BCEA work?

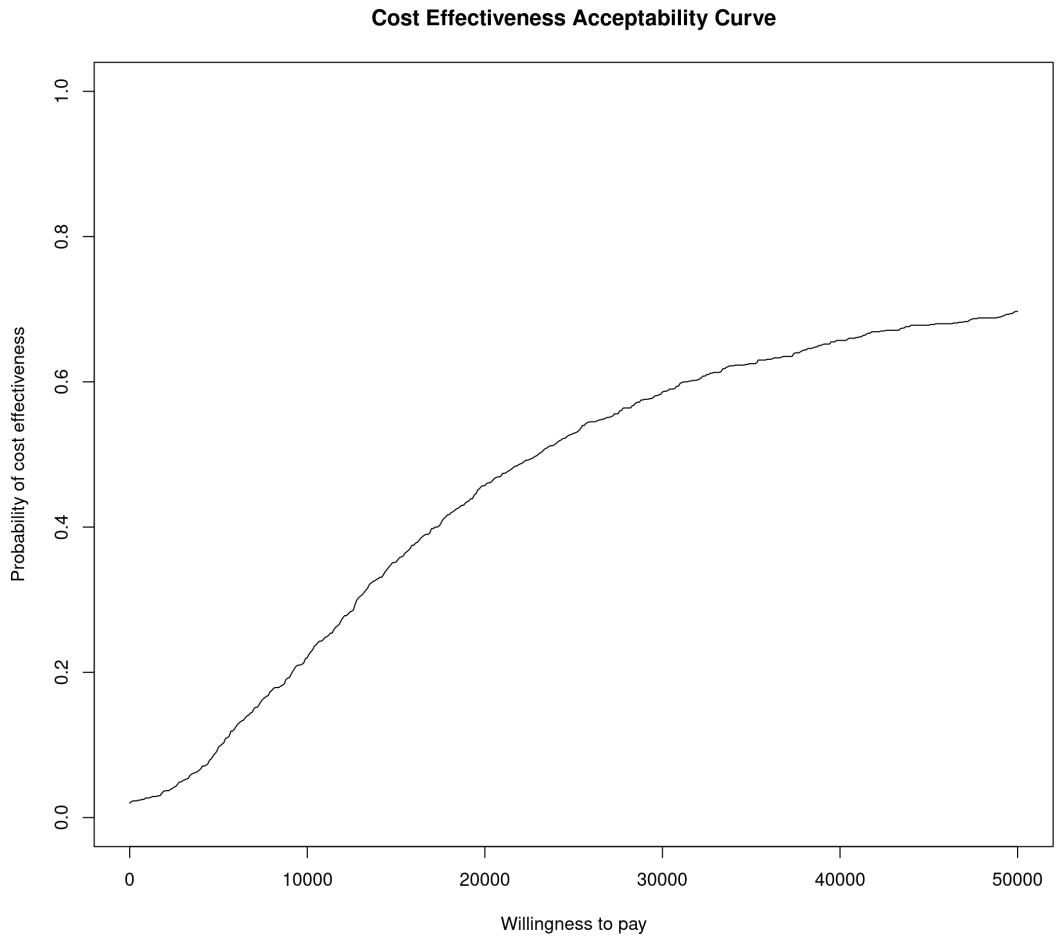
```
# Using 'ggplot', you can go crazy with customisation...
ceplane.plot(m,wtp=10000,graph="gg",point_colors="blue",point_sizes=.8,area_color="springgreen3")
```



https://ggplot2.tidyverse.org/

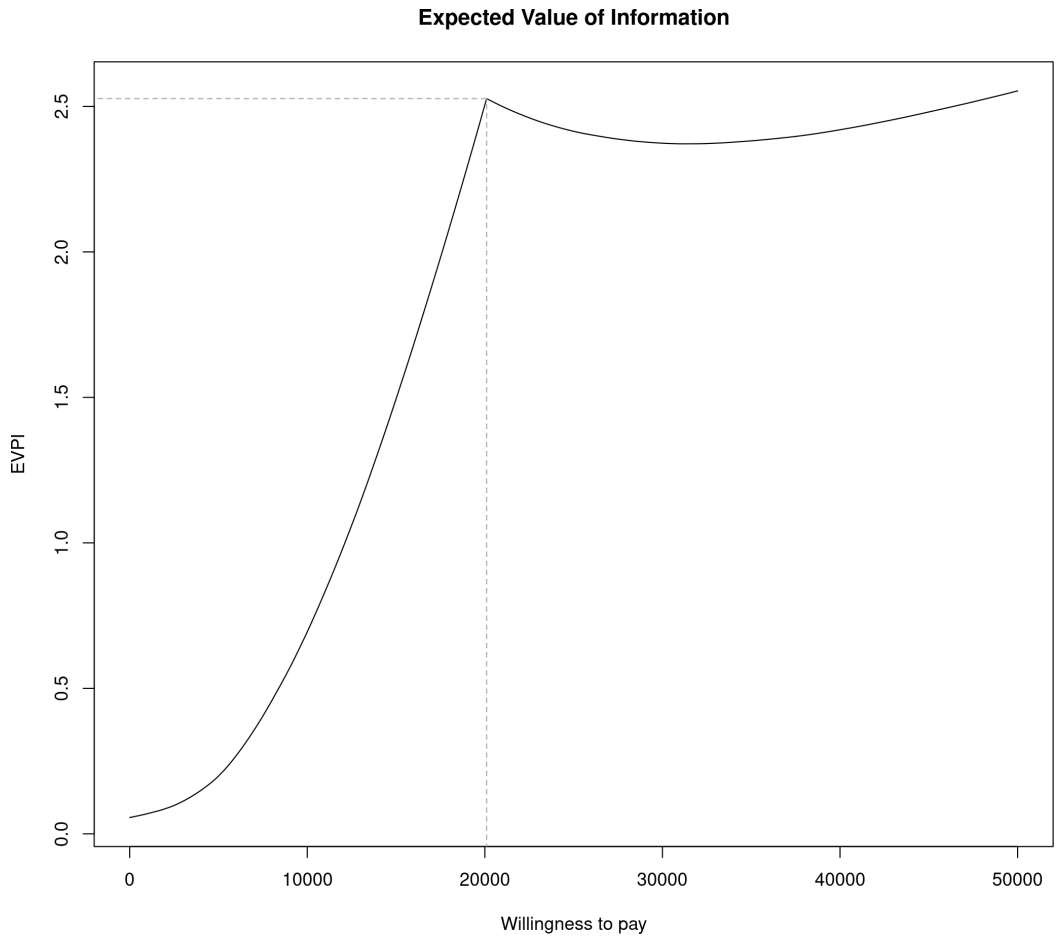
How does BCEA work?

```
# Plots the Cost-Effectiveness Acceptability Curve
ceac.plot(m)
```



How does BCEA work?

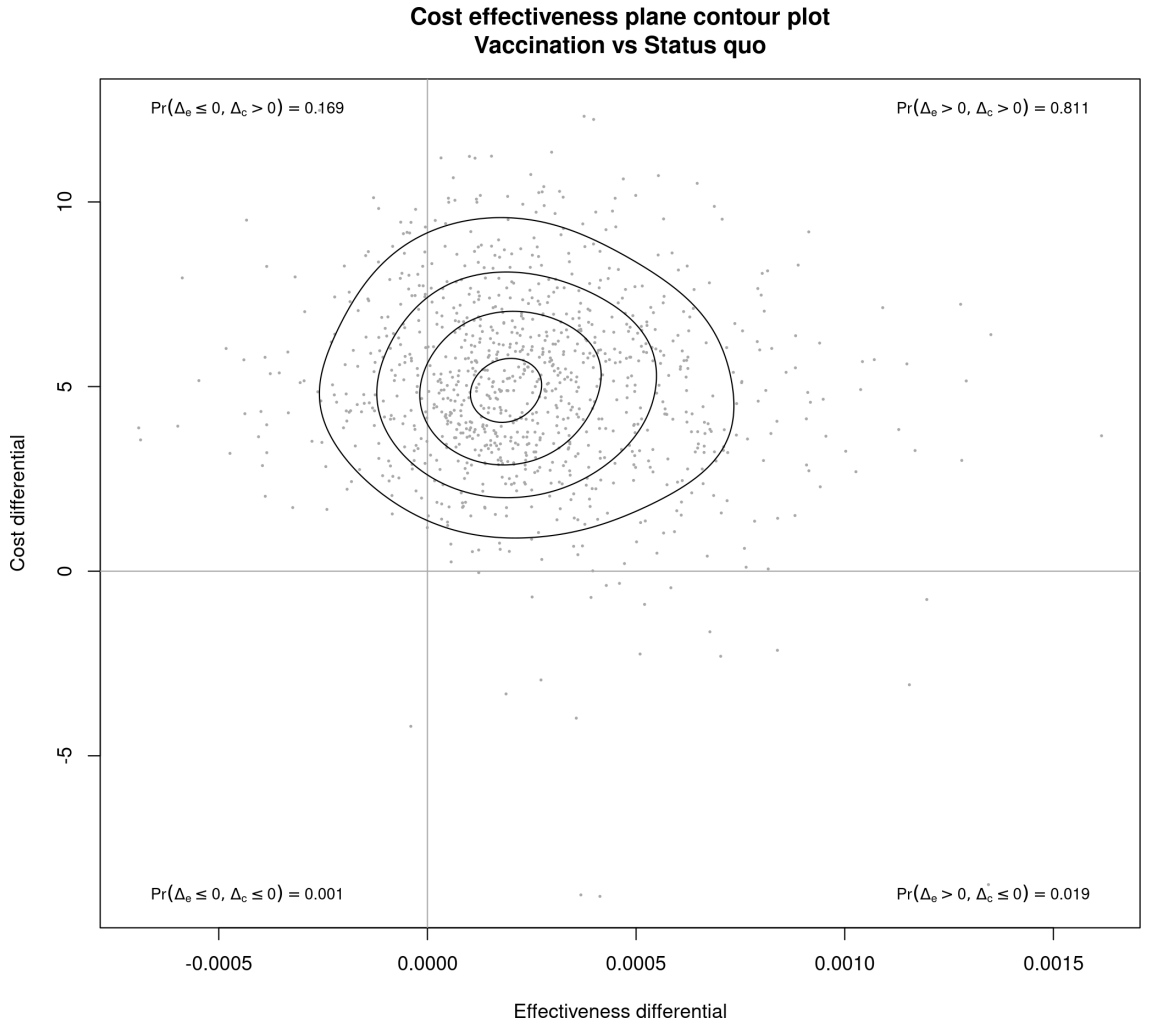
```
# Plots the Expected Value of Partial Information (EVPI)
evi.plot(m)
```



Specialised plots

- Can generate a *contour* plot of the cost-effectiveness plane and estimate the proportion of points in each quadrant

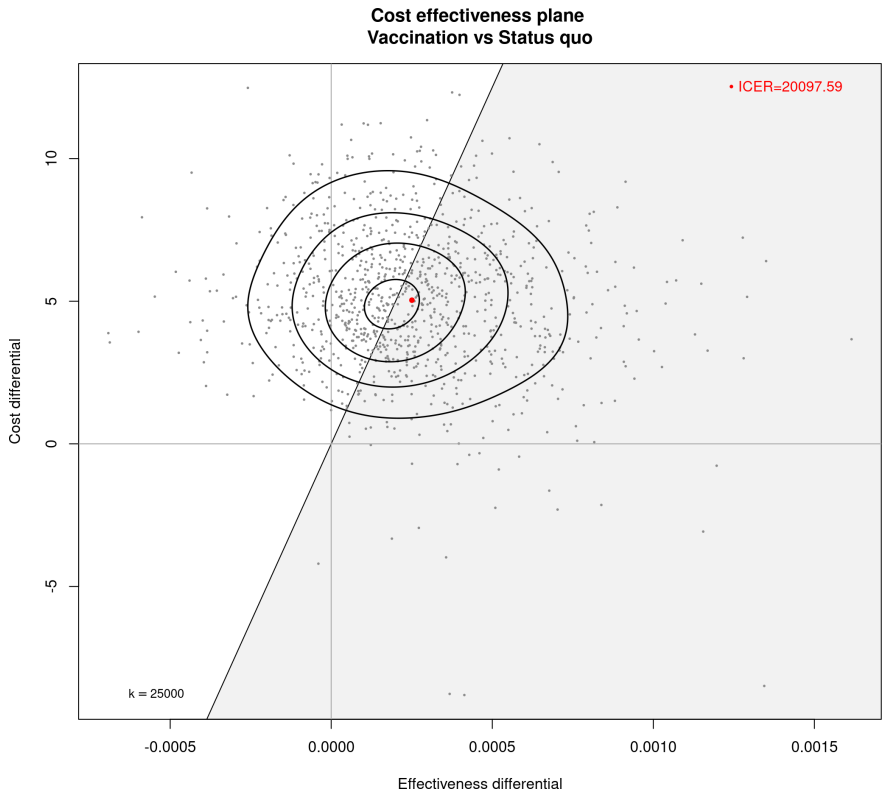
```
# "Basic" contourplot
contour(m)
```



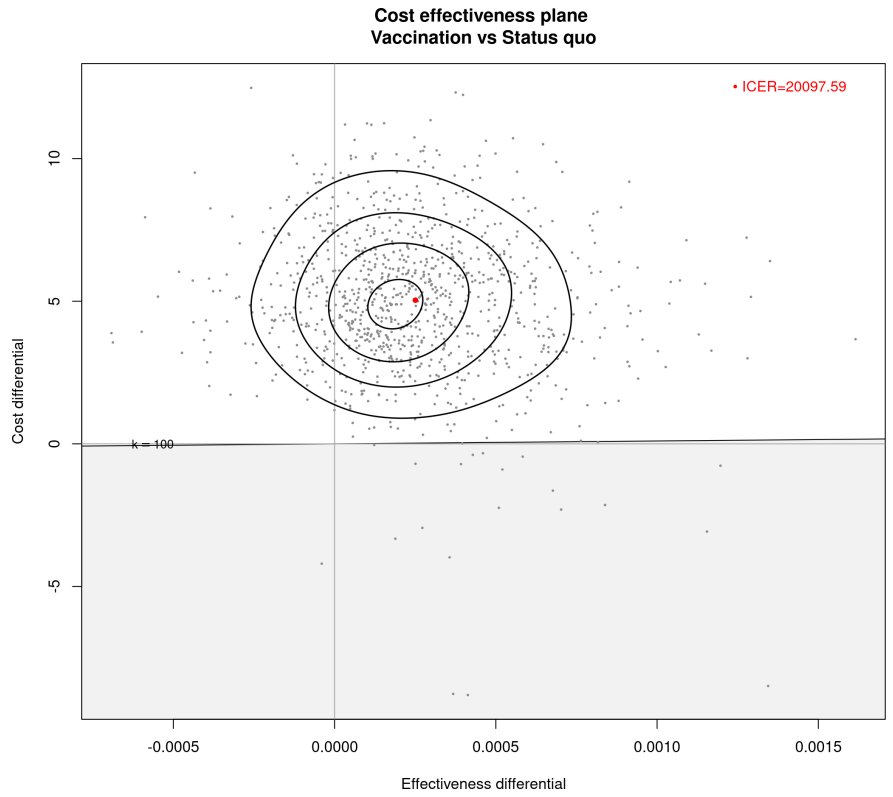
Specialised plots

- The specialised function `contour2` also shows the **sustainability area**

```
contour2(m)
```




```
contour2(m,wtp=100)
```



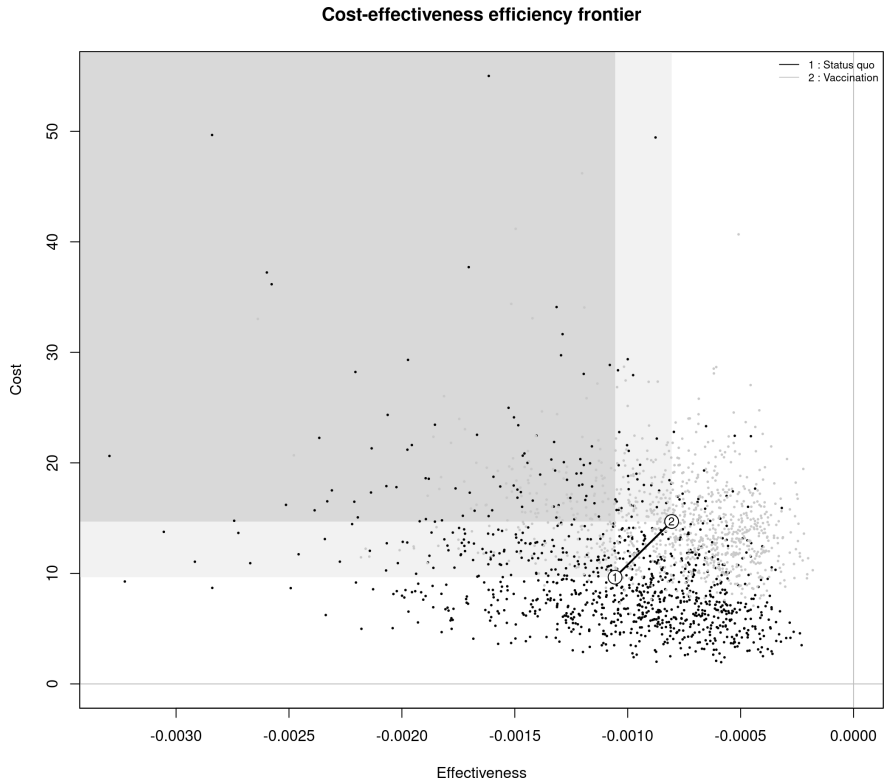
Specialised plots

Cost-effectiveness efficiency frontier

```
ceef.plot(m,print.plot=FALSE)
```

```
Cost-effectiveness efficiency frontier summary

Interventions on the efficiency frontier:
            Effectiveness   Costs Increase slope Increase ang
Status quo     -0.00105595  9.6555             NA
Vaccination    -0.00080537 14.6914          20098          1.5
```

```
ceef.plot(m,print.summary=FALSE)
```



Exporting graphical output

- R has excellent graphical facilities and the graphs produced by BCEA can be easily exported to many different formats

```
# "Opens" the graphical device
pdf("NAME_OF_THE_FILE",width=8,height=8)       # for 'pdf', units are in inches
# Makes the plot
ceplane.plot(BCEA_OBJECT)                      # of course, specify whatever name you've chosen when creating the object...
# "Closes" the graphical device
dev.off()


# "Open" the graphical device"
jpeg("NAME_OF_FILE.jpg",width=480,height=480)  # for 'jpeg' units are in px
# Makes the plot
ceplane.plot(BCEA_OBJECT)
# "Closes" the graphical device
dev.off()
```

NB: Rstudio and rmarkdown can do even more – that's for another time...

Part II

Advanced use of BCEA

Multiple treatment comparisons

Probabilistic "depression model"

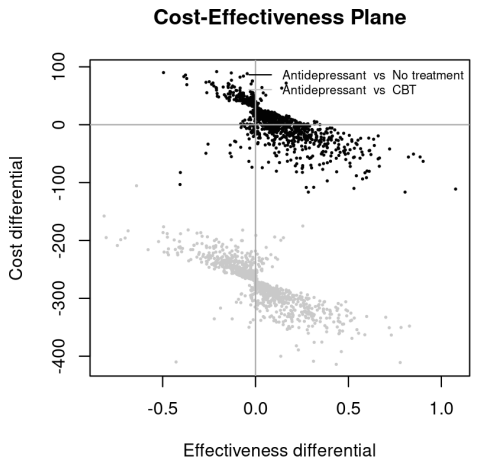
- Fictional model comparing antidepressants to cognitive behaviour therapy (CBT) and no treatment in people with depression
- Statistical modelling based on evidence synthesis
  - Benefits: based on QALYs
  - Costs: associated with treatments and various resources use
- Economic modelling: two matrices with relevant population summaries
  - `effects`
  - `costs`

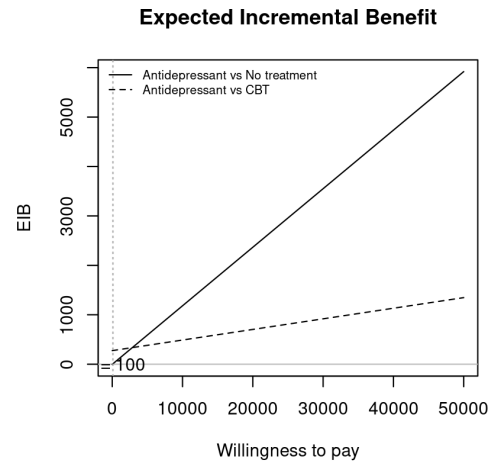
- NB: The details of the actual modelling are *not* important for the purposes of demonstrating the example...

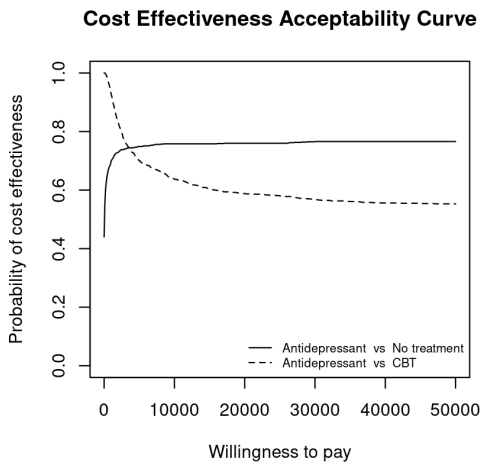
Multiple treatment comparisons

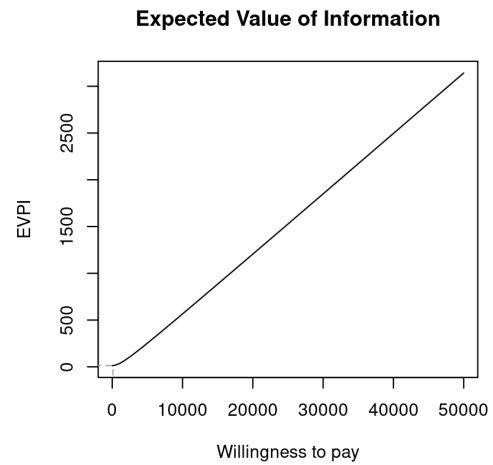
Probabilistic "depression model"

```
# Intervention labels
t.names<-c("No treatment","CBT","Antidepressant")

# "Standard" analysis: pairwise comparisons
depression.bcea = bcea(effects,costs,
                       interventions=t.names,ref=3)
# the third intervention is the reference

# Plots the results
plot(depression.bcea)
```





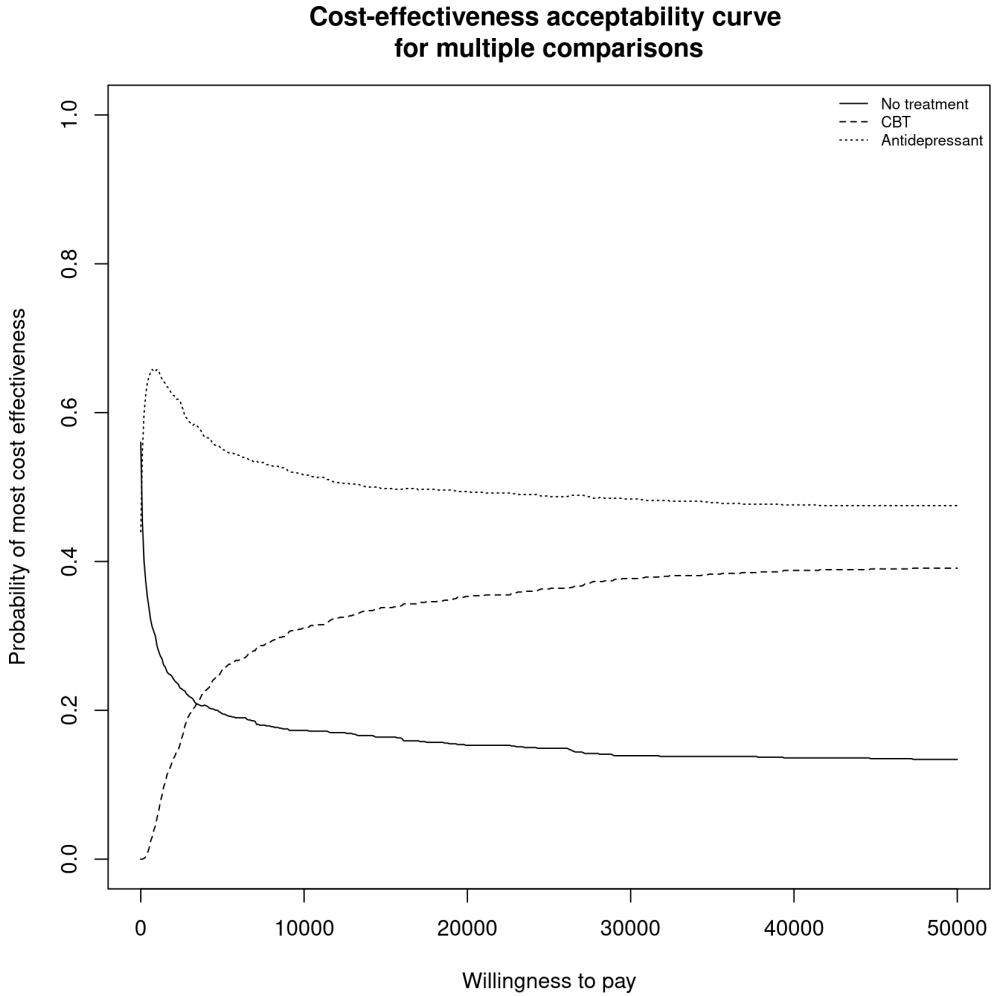




Multiple treatment comparisons

Probabilistic "depression model"

```
# For multiple treatment comparison
depression.multi.ce = multi.ce(depression.bcea)

# Specialised plot method
`mce.plot`(depression.multi.ce,pos=c(1,0.8),
           graph=c("base","ggplot2"))
```

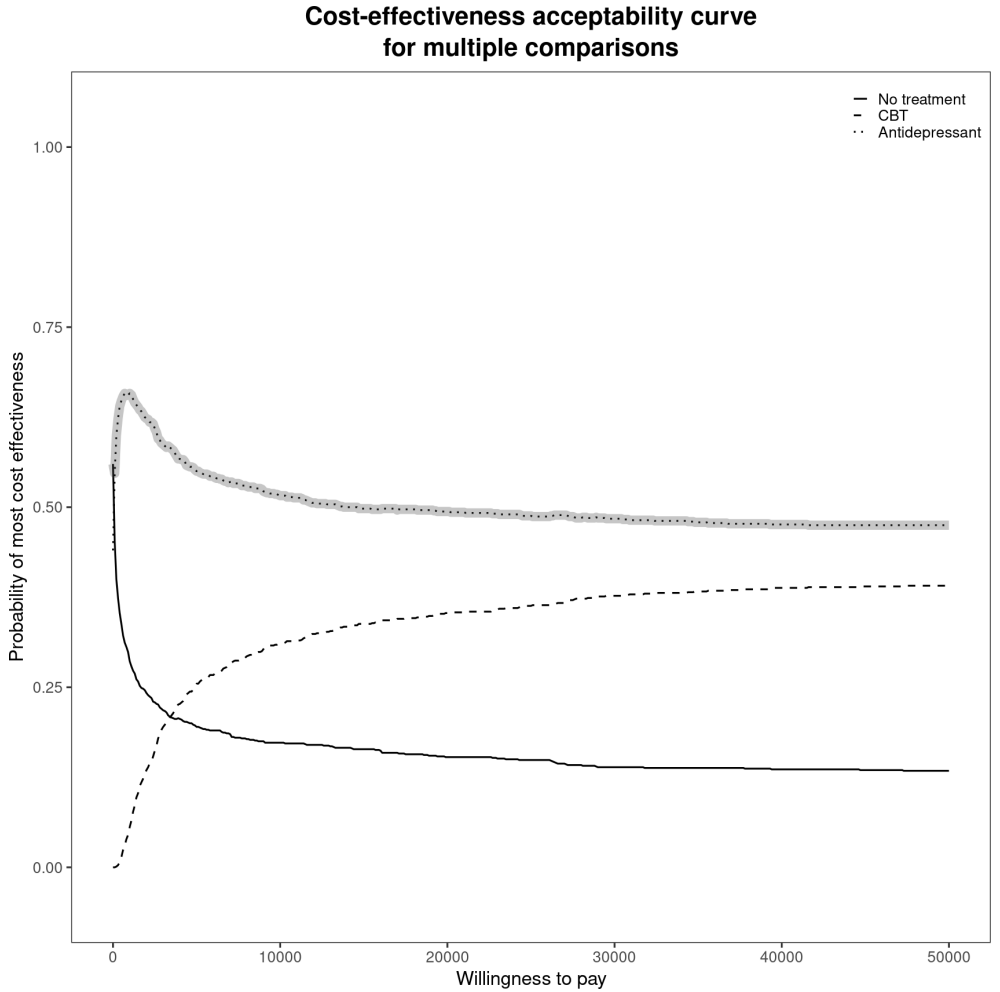


Multiple treatment comparisons

Probabilistic "depression model"

- Can use `ggplot` to customise the graph

```
mce.plot(depression.multi.ce,pos=c(1,1),graph="ggplot2") +
  ggplot2::stat_summary(fun=max, geom="line",
                        colour="grey25", alpha=.3, lwd=2.5)
```

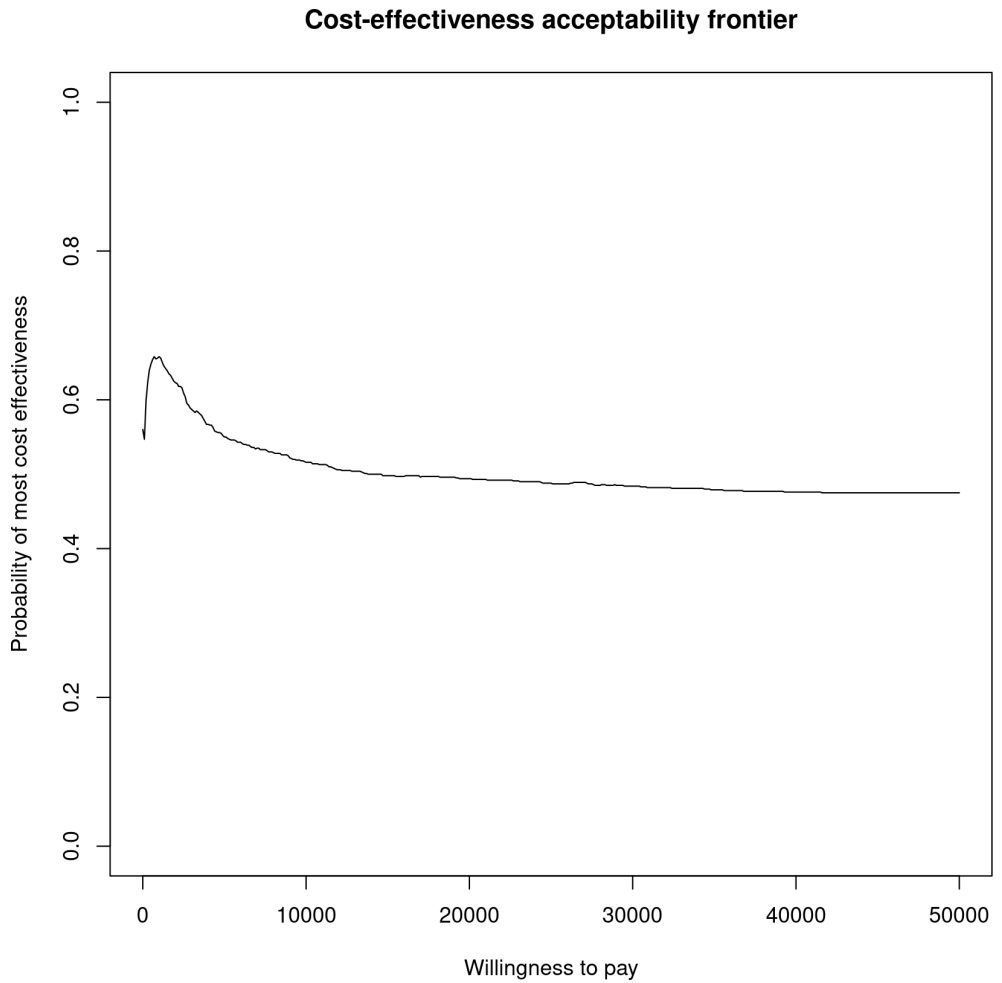


Multiple treatment comparisons

Probabilistic "depression model"

- Can use a specialised plotting method for the *cost-effectiveness acceptability* ***frontier***

```
# Specialised plot
ceaf.plot(depression.multi.ce)
```



BCEAweb

- Inspired by similar projects – eg SAVI
- Create a web interface to use BCEA without even opening R (or even having it installed on your computer!)

- Typical work flow
  1. Design the economic model (eg Markov model, decision tree, ...)
  2. Run the statistical analysis to estimate the quantities of interest (eg survival analysis, evidence synthesis, ...)
  3. Run the economic model and obtain "PSA samples"

  4. Upload "PSA samples", including values for (e, c) to `BCEAweb`
  5. Use `BCEA` in the background to do **all** the economic analysis
  6. Create reports that can be used as the basis for papers, reimbursement files, ...

BCEAweb

```
# Creates a matrix with the underlying model simulations
inp = CreateInputs(vaccine, print.lincom=FALSE)

# Runs BCEAweb
BCEAweb(e=e,                  # matrix of simulations for the effectiveness
        c=c,                  # matrix of simulations for the costs
        parameters=inp$mat    # matrix of simulations for all the model parameters
       )
```

- BCEAweb exists as a standalone webapp
  - Can access it here
- **Or**, you can launch your own "local" version from the BCEA package (as in the code above)!
  - This will launch a web page from which you can manipulate your output (➔ Live Demo)