

Chapter 2

Case Studies

2.1 Introduction

In this chapter, we present the two case studies that are used as running examples throughout the book.

The first example considers a *decision-analytic model*. This is a popular tool in health economic evaluation, when the objective is to compare the expected costs and consequences of decision options by synthesising information from multiple sources [1]. Examples of decision-analytic models include decision trees or Markov models. As these models are based on several different sources of information, they can offer decision-makers the best available information to reach their decision, as opposed to modelling based on a single randomised clinical trial (RCT). Additionally, RCTs may be limited in scope (e.g. in terms of the temporal follow up) and thus not be ideal for characterising the long-term consequences of applying an intervention. Thus, even when RCT data are available, a health economic evaluation is often extended to a decision-analytic approach to allow decision-makers to capture information about the long-term effects and costs.

The second example is a multi-decision problem. This means that, in contrast to standard economic modelling, where a new intervention $t = 1$ is compared to the status quo $t = 0$, we consider here $T = 4$ potential interventions. This is naturally linked to the wider topic of *network meta-analysis* [2], which is an extension of different statistical methods that allow researchers to pool evidence coming from different sources and include direct and indirect comparisons. This is particularly relevant when head-to-head evidence is only available for some of the interventions under consideration. Suitable statistical modelling can be used to infer about the direct comparisons with no information by using the indirect ones.

For both examples, we first introduce the general background, discussing the disease area and the interventions under consideration. We then describe the assumptions underlying the statistical model (e.g. in terms of distributions for the observed/observable variables and the unobservable parameters). We also show how these can be translated into suitable code to perform a full Bayesian analysis and obtain

samples from the relevant posterior distributions (see Sect. 1.2). Finally, we demonstrate any post-processing required to produce the relevant inputs for the economic model (e.g. the population average differential of costs and benefits—see Sect. 1.4). In the rest of the book, we refer (rather interchangeably) to OpenBUGS [3] and JAGS [4], arguably the most popular software to perform Bayesian analysis by means of Markov Chain Monte Carlo simulations.

The examples are then used to showcase the facilities of BCEA and to explain the process of performing an economic evaluation in R, once the statistical model has been fitted. It is important to note (and we will expand on this point in Chap. 3) that BCEA can be used to perform cost-effectiveness analysis when the full statistical model has not been fitted within the Bayesian framework. Nevertheless, we strongly advocate the use of a Bayesian framework and thus we have included these examples to demonstrate a full Bayesian analysis.

Starting from this chapter on, the text will include frequent code blocks to show how to execute commands and use the BCEA package. R code is presented in code blocks in the text, with each new line starting with the symbol `>`. Indentation indicates lines continuing from the previous statement. Hash symbols (i.e. `#`) in code blocks indicate comments. In-line words formatted in mono-spaced font (such as `this`) indicate code, for example short commands or function parameters.

2.2 Preliminaries: Computer Configuration

In this section, we briefly review the ideal computer configuration we are assuming to run the examples later in this chapter and in the rest of the book. It is difficult to guarantee that these instructions will be valid for every future release of the programmes we consider here, although they have been tested under the current releases.

We assume that the user's computer has the following software installed:

- R and the package BCEA. Other optional packages (e.g. R2OpenBUGS, R2jags, reshape, plyr or INLA) may need to be installed;
- OpenBUGS or JAGS. These are necessary to perform the full Bayesian analyses we discuss in the rest of the book. It is not necessary to install both;
- The R front-end, for example Rstudio (available for download at the webpage <https://www.rstudio.com/>). This is also optional and all the work can be done using the standard R terminal;
- A spreadsheet calculator, e.g. MS Excel or the freely available LibreOffice, which is a decent surrogate and can be downloaded at <https://www.libreoffice.org/>.

In the following, we provide some general instructions, for MS Windows, Linux or Mac OS operating systems.

2.2.1 MS Windows *Users*

For MS Windows users, the set-up should be fairly easy and amounts to the following steps:

1. Install OpenBUGS

- Download the latest release (currently it is version 3.2.3, stored in the file OpenBUGS323setup.exe) from <http://openbugs.net/w/Downloads> and run it by double-clicking on it.

2. Install R

- a. Download R from the Comprehensive R Archive Network (CRAN): <http://cran.r-project.org/bin/windows/> (click on the link “install R for the first time”).
- b. When the process is finished, open R and type in the terminal the following command.

```
> install.packages("BCEA")
> install.packages("R2OpenBUGS")
```

These commands will download and install the packages BCEA and R2OpenBUGS. The latter is needed to interface OpenBUGS with R. Follow the on-screen instructions (you will be asked to select a *mirror* from which to obtain the necessary files). Notice that the command `install.packages("Name_of_the_package")` can be used to install any other R package.

3. (Optional): Install JAGS

- a. Download the installer from the webpage <http://sourceforge.net/projects/mcmc-jags/files/JAGS/4.x/Windows/> by clicking on the latest available executable file (currently, JAGS-4.2.0.exe). Executing this file will install JAGS on the user's machine.
- b. In the R terminal type the command

```
> install.packages("R2jags")
```

This will install the package R2jags, which allows to interface JAGS from R.

2.2.2 Linux *or* Mac OS *Users*

2.2.2.1 Installing R and BCEA

Linux or Mac OS users should follow slightly different approaches. The installation of R is pretty much the same as for MS Windows users. From the webpage <http://cran.r-project.org/> select the relevant operating system (Linux or Mac OS) and then the relevant version (e.g. debian, redhat, suse or ubuntu, for Linux). Follow the instructions to install the software. Once this is done, open R and install the package BCEA following the process described above.

2.2.2.2 Installing OpenBUGS and JAGS in Linux

OpenBUGS runs natively in Linux and so it can be installed following the instructions given at <http://openbugs.net/w/Downloads>. First, download the most recent version of the source file, currently OpenBUGS-3.2.3.tar.gz. Then open a Linux terminal and follow these steps:

1. Unpack the file and move to the newly created directory OpenBUGS-3.2.3. by typing the following commands.

```
tar xzvf OpenBUGS-3.2.3.tar.gz
cd OpenBUGS-3.2.3
```

2. Compile and install the software

```
./configure
make
sudo make install
```

Notice that if the user does not have administrative access, this command will fail. A possible workaround is to specify a location to which OpenBUGS should be installed that is owned by the user, for example

```
./configure --prefix=/home/user/myfolder
make
sudo make install
```

— it is possible to check permission by using the Unix command

```
ls -ls /home/user
```

which returns a list of the folders and files contained in the folder /home/user. This will look something like

```
4 drwxr-xr-x  9 user user   4096 Jul  5 17:02 Desktop
4 drwx----- 25 user user   4096 Jul 14 09:23 myfolder
196 -rw-rw-r--  1 root root 197534 Jul 11 16:08 some_file.png
...
```

and, in this case, the folder myfolder does belong to the user user and thus the installation of OpenBUGS in that folder would be completed successfully.

It is also possible to install JAGS, following these steps:

1. Download the latest tar.gz file (currently, JAGS-4.2.0.tar.gz) from the web-page <http://sourceforge.net/projects/mcmc-jags/files/JAGS/4.x/Source/>.
2. Open a Linux terminal window, extract the content of the archive file and move to the newly created folder JAGS-4.2.0

```
tar xzvf JAGS-4.2.0.tar.gz
cd JAGS-4.2.0
```

3. Run the configuration

```
sudo ./configure --prefix=/usr
sudo make
sudo make install
```

4. Clean up the unnecessary files and folder

```
cd ..
sudo rm -fr JAGS-4.2.0
rm JAGS-4.2.0.tar.gz
```

5. Install R2jags from the R terminal, as discussed in Sect. 2.2.1.

2.2.2.3 Installing OpenBUGS and JAGS in Mac OS

While OpenBUGS does not run natively under Mac OS, a possible workaround is to install a hardware virtualisation software such as Parallels Desktop for Mac OS (<http://www.parallels.com/uk/products/desktop/>), or a “compatibility layer”, such as wine (<https://www.winehq.org/download/>), which allow to run Windows applications from Mac.

Conversely, JAGS does run natively under Mac OS too and can be installed using the following steps:

1. Download the latest .dmg file (currently, JAGS-4.2.0.dmg) from <https://sourceforge.net/projects/mcmc-jags/files/JAGS/4.x/Mac%20OS%20X/>
2. Double click the .dmg file to make its content available (the name will show up in the *Finder* sidebar), usually a window opens showing the content as well;
3. Drag the application from the .dmg window into /Applications to install (you may need an administrator password);
4. Wait for the copy process to finish;
5. Eject the .dmg (by clicking the eject button in the *Sidebar*);
6. Delete the .dmg from Downloads.

Several tutorial are available online to guide in the process of installation and use of both OpenBUGS and JAGS.

2.3 Vaccine

Consider an infectious disease, for instance influenza, for which a new vaccine has been produced. Under the current management of the disease some individuals treat the infection by taking over-the-counter (OTC) medications. Some subjects visit their doctor and, depending on the gravity of the infection, may receive treatment with antiviral drugs, which usually cure the infection. However, in some cases complications may occur. Minor complications will need a second doctor’s visit after which the patients become more likely to receive antiviral treatment. Major complications

are represented by pneumonia and can result in hospitalisation and possibly death. In this scenario, the costs generated by the management of the disease are represented by OTC medications, doctor visits, the prescription of antiviral drugs, hospital episodes and indirect costs such as time off work.

The focus is on the clinical and economic evaluation of the policy that makes the vaccine available to those who wish to use it ($t = 1$) against the null option ($t = 0$) under which the vaccine will remain unavailable. More details of this example can be found in [5] and references therein.

2.3.1 (Bayesian) Statistical Model

2.3.1.1 Assumptions

In a population made of N individuals, consider the number of patients taking up the vaccine when available $V_1 \sim \text{Binomial}(\phi, N)$ where ϕ is the vaccine coverage rate. Obviously, $V_0 = 0$ as the vaccine is not available under the status quo. For convenience, we denote the total number of patients in the two groups, vaccinated ($v = 1$) and non-vaccinated ($v = 0$), by n_{tv} with $n_{t1} := V_t$ and $n_{t0} := N - V_t$, respectively.

The relevant clinical *outcomes* are: $j = 1$ influenza infection; $j = 2$ doctor visit; $j = 3$ minor complications; $j = 4$ major complications; $j = 5$ hospitalisation; $j = 6$ death; and $j = 7$ adverse events of influenza vaccination. For each clinical outcome j , β_j is its baseline rate of occurrence and ρ_v is the proportional reduction in the chance of infection due to the vaccine. Vaccinated patients ($v = 1$) will experience a reduction in the chance of infection by a factor ρ_1 ; conversely, for $v = 0$, individuals are not vaccinated and so the chance of infection is just the attack rate β_1 . This is equivalent to setting $\rho_0 := 0$.

Under these assumptions, the number of individuals becoming infected in each group is $I_{tv} \sim \text{Binomial}(\pi_v, n_{tv})$, where $\pi_v := \beta_1(1 - \rho_v)$ is the probability of infection. Among the infected subjects, the number visiting a doctor for the first time is $GP_{tv}^{(1)} \sim \text{Binomial}(\beta_2, I_{tv})$. Using a similar reasoning, among those who have had a doctor visit, we can define: the number of individuals with minor complications $GP_{tv}^{(2)} \sim \text{Binomial}(\beta_3, GP_{tv}^{(1)})$; the number of those with major complications $P_{tv} \sim \text{Binomial}(\beta_4, GP_{tv}^{(1)})$; the number of hospitalisations $H_{tv} \sim \text{Binomial}(\beta_5, GP_{tv}^{(1)})$; and the deaths $D_{tv} \sim \text{Binomial}(\beta_6, GP_{tv}^{(1)})$. The number of individuals experiencing adverse events due to vaccination is computed as $AE_{tv} \sim \text{Binomial}(\beta_7, n_{tv})$ —obviously, this will be identically 0 under the status quo ($t = 0$) and among those individuals who choose not to take the vaccine up in the vaccination scenario ($t = 1, v = 0$).

The model also includes other parameters, such as the chance of receiving a prescription after the first doctor visit (γ_1) or following minor complications (γ_2) for a number of antiviral drugs (δ); of taking OTC medications (ξ); and of remaining off-work (η) for a number of days (λ). Combining these with the relevant populations

at risk, we can then derive the expected number of individuals experiencing each of these events.

As for the costs, we consider the relevant *resources* as $h = 1$: doctor visits; $h = 2$: hospital episodes; $h = 3$: vaccination; $h = 4$: time to receive vaccination; $h = 5$: days off work; $h = 6$: antiviral drugs; $h = 7$: OTC medications; $h = 8$: travel to receive vaccination. For each, we define ψ_h to represent the associated unit cost for which we assume informative lognormal distributions, a convenient choice to model positive, continuous variables such as costs.

Finally, we include in the model suitable parameters to represent the loss in quality of life generated by the occurrence of the clinical outcomes. Let ω_j represent the QALYs lost when an individual experiences the j -th outcome. We assume that doctor visits do not generate loss in QALYs and therefore set $\omega_2 = \omega_3 := 0$; the remaining ω_j 's are modelled using informative lognormal distributions.

The assumptions encoded by this model are that we consider a population parameter $\theta = (\theta^0, \theta^1)$, with the two components being defined as $\theta^0 = (\beta_j, \gamma_1, \gamma_2, \delta, \xi, \eta, \lambda, \psi_h, \omega_j)$ and $\theta^1 = (\phi, \beta_j, \rho_v, \gamma_1, \gamma_2, \delta, \xi, \eta, \lambda, \psi_h, \omega_j)$. We assume that the components of θ have the distributions specified in Table 2.1, which are derived by using suitable “hyper-parameters” that have been set to encode knowledge \mathcal{D} available from previous studies and expert opinion. For example, the parameter ϕ identifies a probability (the vaccine coverage) and we may have information about past seasons to suggest that this has been estimated to be between 25 and 63%; this can be translated to a Beta distribution whose parameters can be determined so that roughly 95% of the probability mass lie between these two values. See [5] for a more detailed discussion of this point.

It is easy to check that the assumptions in terms of the interval estimates for the parameters are consistent with the choice of distributions in R, for example using something like the following code:

```
> phi <- rbeta(100000, 11.31, 14.44)
> c(quantile(phi, .025), quantile(phi, .5), quantile(phi, .975))
      2.5%      50%      97.5%
0.2581039 0.4368227 0.6292335
```

2.3.1.2 Coding the Assumptions into BUGS/JAGS Language

The assumptions and the model structure defined above can be translated into suitable code to perform a MCMC analysis and obtain estimates from the posterior distributions of all the relevant parameters. For example, we could write the following code to be used with OpenBUGS or JAGS:

```
model {
# 1. Define the number of people in each group n[v,t], where t=1,2 is status
  quo vs vaccination and v=1,2 is non vaccinated vs vaccinated
# t=1: If the vaccine is not available, no one will use it
  # number of vaccinated in the population
  V[1] <- 0
  # number of individuals in the two groups
```

```

n[1,1] <- N - V[1]      # non vaccinated
n[2,1] <- V[1]          # vaccinated

# t=2: When the vaccine is available, some people will use it but
# some people won't
# number of vaccinated in the population
V[2] ~ dbin(phi,N)
# number of individuals in the two groups
n[1,2] <- N - V[2]      # non vaccinated
n[2,2] <- V[2]          # vaccinated

# 2. Vaccination coverage
phi ~ dbeta(a.phi,b.phi)

# 3. Probability of experiencing the clinical outcomes (in total, N.outcomes =
7)
# 1. Influenza infection
# 2. GP visits
# 3. Minor complications (repeat visit)
# 4. Major complications (pneumonia)
# 5. Hospitalisations
# 6. Death
# 7. Adverse events due to vaccination
for (r in 1:4) {
  beta[r] ~ dbeta(a.beta[r],b.beta[r])
}
for (r in 5:6) {
  beta[r] ~ dlnorm(a.beta[r],b.beta[r])
}
beta[N.outcomes] ~ dbeta(a.beta[N.outcomes],b.beta[N.outcomes])

# 4. Vaccine effectiveness in reducing influenza (for v=1, it is obviously 0)
rho[1] <- 0
rho[2] ~ dlnorm(mu.rho,tau.rho)

# 5. Probability of influenza infection
for (t in 1:2) {
  for (v in 1:2) {
    pi[t,v] <- beta[1]*(1-rho[v])
  }
}

# 6. Number of patients experiencing the events for both
interventions & compliance groups
for (t in 1:2) {
  for (v in 1:2) {
    Infected[t,v] ~ dbin(pi[t,v],n[v,t])
    GP[t,v] ~ dbin(beta[2],Infected[t,v])
    Repeat.GP[t,v] ~ dbin(beta[3],GP[t,v])
    Pneumonia[t,v] ~ dbin(beta[4],GP[t,v])
    Hospital[t,v] ~ dbin(beta[5],GP[t,v])
    Death[t,v] ~ dbin(beta[6],GP[t,v])
    Trt[1,t,v] ~ dbin(gamma[1],GP[t,v])
    Trt[2,t,v] ~ dbin(gamma[2],Mild.Compl[t,v])
  }
}

```



```

      Mild.Compl[t,v] <- Repeat.GP[t,v] + Pneumonia[t,v]
    }
  }
  Adverse.events ~ dbin(beta[N.outcomes],n[2,2])

# 7. Probability of experiencing other events (impacts on costs and QALYs/QALDs
)
for (i in 1:2) {
  # Treatment with antibiotics after GP visit
  gamma[i] ~ dbeta(a.gamma[i],b.gamma[i])
}
# Number of prescriptions of antivirals
delta ~ dpois(a.delta)
# Taking OTC
xi ~ dbeta(a.xi,b.xi)
# Being off work
eta ~ dbeta(a.eta,b.eta)
# Length of absence from work for influenza
lambda ~ dlnorm(mu.lambda,tau.lambda)

# 8. Costs of clinical resources (N.resources = 8)
# 1. Cost of GP visit
# 2. Cost of hospital episode
# 3. Cost of vaccination
# 4. Cost of time to receive vaccination
# 5. Cost of days work absence due to influenza
# 6. Cost of antiviral drugs
# 7. Cost of OTC treatments
# 8. Cost of travel to receive vaccination
for (r in 1:N.resources) {
  psi[r] ~ dlnorm(mu.psi[r],tau.psi[r])
}

# 9. Quality of life adjusted days/years loss
# 1. Influenza infection
# 2. GP visits (no QALD/Y loss)
# 3. Minor complications (repeat visit, no QALD/Y loss)
# 4. Major complications (pneumonia)
# 5. Hospitalisations (same QALD/Y loss as pneumonia)
# 6. Death
# 7. Adverse events due to vaccination
omega[1] ~ dlnorm(mu.omega[1],tau.omega[1])
omega[2] <- 0; omega[3] <- 0;
for (r in 4:N.outcomes) {
  omega[r] ~ dlnorm(mu.omega[r],tau.omega[r])
}
}

```

Table 2.1 Distributional assumptions for the model. For each parameter, the distributions are chosen to model the available prior knowledge, represented by existing data or expert opinions. The mathematical form of the distributions is chosen according to the nature of the parameter (i.e. parameters describing probability of occurrence of an event are usually given a Beta distribution), while the values of the hyper-parameters are chosen so that the distribution is consistent with the prior information derived by the clinical literature or expert opinion

Parameter	Mean	2.5%	Median	97.5%	Distribution
ϕ	0.435	0.245	0.436	0.625	Beta (11.31, 14.44)
β_1	0.0701	0.0387	0.0680	0.1116	Beta (13.01, 172.38)
β_2	0.295	0.124	0.288	0.497	Beta (5.80, 13.80)
β_3	0.401	0.388	0.401	0.415	Beta (1909.50, 2851.86)
β_4	0.01339	0.00852	0.01322	0.01938	Beta (20.94, 1538.71)
β_5	0.000378	0.000223	0.000364	0.000616	Lognormal (−7.91, 14.93)
β_6	0.000748	0.000366	0.000702	0.001331	Lognormal (−7.26, 7.66)
β_7	0.1021	0.0255	0.0954	0.2265	Beta (3.50, 31.50)
ρ_1	0.688	0.593	0.686	0.794	Lognormal (−0.374, 0.00524)
γ_1	0.420	0.417	0.420	0.423	Beta (45471.58, 62794.09)
γ_2	0.814	0.806	0.814	0.822	Beta (7701.86, 1759.89)
δ	6.97	2.00	7.00	12.00	Poisson (7.00)
ξ	0.950	0.940	0.950	0.959	Beta (1804.05, 94.95)
η	0.900	0.890	0.900	0.909	Beta (3239.10, 359.90)
λ	2.90	1.22	2.69	5.97	Lognormal (0.98, 0.17)
ψ_1	20.55	12.36	19.77	32.07	Lognormal (3.00, 0.0606)
ψ_2	2661.92	1554.18	2575.67	4106.98	Lognormal (7.85, 0.0606)
ψ_3	7.21	4.22	6.95	11.42	Lognormal (1.95, 0.0606)
ψ_4	10.26	6.16	9.92	15.90	Lognormal (2.29, 0.0606)
ψ_5	46.31	27.20	44.96	70.69	Lognormal (3.80, 0.0606)
ψ_6	3.86	2.39	3.73	5.95	Lognormal (1.31, 0.0606)
ψ_7	1.592	0.949	1.562	2.452	Lognormal (0.44, 0.0606)
ψ_8	0.807	0.484	0.776	1.311	Lognormal (−0.241, 0.0606)
ω_1	4.26	2.14	4.05	7.59	Lognormal (1.40, 0.0993)
ω_4	6.39	3.81	6.23	9.82	Lognormal (1.82, 0.0606)
ω_5	6.34	3.83	6.15	9.94	Lognormal (1.82, 0.0606)
ω_6	15.20	9.09	14.88	23.34	Lognormal (2.70, 0.054)
ω_7	0.556	0.316	0.541	0.932	Lognormal (−0.634, 0.0717)

The model consists of nine modules as annotated in the code above. Notice that the values of the parameters for each distribution are kept as variables (rather than hard-coded as a fixed number). This is in general a good idea, since changes in the assumed values can be reflected directly using the same code. Of course, this means that the numerical value must be passed to the computer code somewhere else in the scripting process. This, however, helps clarify the whole process and makes debugging easier.

As is possible to see, most of the commands in the BUGS/JAGS language are effectively typed in a way that strongly resembles the standard statistical notation, with the twiddle symbol \sim indicating a stochastic relationship (i.e. a probability distribution), while the assignment symbol \rightarrow indicates logical (or deterministic) relationships.

Typically, this code is saved to a text file, say `vaccine.txt`. It is good practice to store the files in a well-structured set of directories or at least to provide pointers for R so that it can search for the relevant files efficiently. Examples include the directory from which R is launched or alternatively in the directory that is currently in use by R (also termed the “working directory”). The R command

```
> setwd("PATH_TO_RELEVANT_FOLDER")
```

can be used to set the working directory to any folder, while the command

```
> getwd()
[1] "/home/user/MyStuff"
```

returns the current (working) directory. Note that R uses Unix-like notation and forward slashes `/` to separate folders in a text string. Conversely, MS Windows uses backward slashes `\` to accomplish the same task. This means that on a MS Windows computer, the working directory will be defined by R as something like

```
> # On a Windows machine:
> getwd()
[1] "C:/user/MyStuff"
```

while the MS Windows notation (e.g. by copying and pasting the address of the folder from the file explorer) would actually be `"C:\user\MyStuff"`. It is thus important to be careful when copying and pasting folder locations from MS Windows into R and the user has two options, both based on Unix-like notation: the first one is to just convert any backward slash to a forward slash. The second option is to escape the backward slashes using a double backward slash (`\\`), for example as in the following R code.

```
> # On a Windows machine, these two commands are the same:
> # 1. using forward slashes
> setwd("C:/user/MyStuff")

> # 2. using double backward slashes
> setwd("C:\\user\\MyStuff")
```

2.3.1.3 R Code to Pre-process and Load the Data

The following R code is used to pre-process and load the data in the R workspace before the model and the health economic analysis can be run.

```
> ## Launches the file Utils.R which contains useful functions used throughout
  this script
> source("http://www.statistica.it/gianluca/BCEABook/WebMaterial/Utils.R")

> ## Loads the values of the hyper-parameters (needed to run the Bayesian model
  using JAGS)
> # Number of people in the populations
> N <- 100000

> # Vaccine coverage
> a.phi <- betaPar2(.434,.6,.95)$res1
> b.phi <- betaPar2(.434,.6,.95)$res2

> # Baseline probabilities of clinical outcomes
> # 1. Influenza infection
> # 2. GP visits
> # 3. Minor complications (repeat visit)
> # 4. Major complications (pneumonia)
> # 5. Hospitalisations
> # 6. Death
> # 7. Adverse events due to vaccination
> N.outcomes <- 7
> mu.beta <- c(.0655,.273,.401,.0128,.00038,.00075)
> upp.beta <- c(.111,.51,.415,.0197,.00067,.000132)
> sd.beta <- c(NA,NA,NA,NA,.0001,.00028)
> a.beta <- b.beta <- numeric()
> for (i in 1:4) {
+ a.beta[i] <- betaPar2(mu.beta[i],upp.beta[i],.975)$res1
+ b.beta[i] <- betaPar2(mu.beta[i],upp.beta[i],.975)$res2
+ }
> for (i in 5:6) {
+ a.beta[i] <- lognPar(mu.beta[i],sd.beta[i])$mulog
+ b.beta[i] <- 1/lognPar(mu.beta[i],sd.beta[i])$sigmalog^2
+ }
> a.beta[N.outcomes] <- betaPar(.1,.05)$a
> b.beta[N.outcomes] <- betaPar(.1,.05)$b

> # Decrease in risk of infection due to vaccination
> mu.rho <- lognPar(.69,.05)$mulog; tau.rho <- 1/lognPar(.69,.05)$sigmalog^2

> # Treatment with antibiotics after GP visit
> mu.gamma <- c(.42,.814); sd.gamma <- c(.0015,.004)
> a.gamma <- b.gamma <- numeric()
> for (i in 1:2) {
+ a.gamma[i] <- betaPar(mu.gamma[i],sd.gamma[i])$a
+ b.gamma[i] <- betaPar(mu.gamma[i],sd.gamma[i])$b
+ }

> # Number of prescriptions of antibiotics
> a.delta <- 7

> # Taking OTC
> a.xi <- betaPar(.95,.005)$a
> b.xi <- betaPar(.95,.005)$b
```

```

> # Being off work
> a.eta <- betaPar(.9,.005)$a
> b.eta <- betaPar(.9,.005)$b

> # Length of absence from work for influenza
> mu.lambda <- lognPar(2.9,1.25)$mulog
> tau.lambda <- 1/lognPar(2.9,1.25)$sigmalog^2

> # Costs (N.resources = 8)
> # 1. Cost of GP visit
> # 2. Cost of hospital episode
> # 3. Cost of vaccination
> # 4. Cost of time off for individuals to receive vaccination
> # 5. Cost of days work absence due to influenza
> # 6. Cost of antibiotics
> # 7. Cost of OTC treatments
> # 8. Cost of travel to receive vaccination
> N.resources <- 8
> m.psi <- c(20.66,2656,7.24,10.16,46.27,3.81,1.6,.81)
> sd.psi <- c(5.015,440.75,1.81,2.54,11.57,.955,.4,.2)
> sd.psi <- .25*m.psi
> mu.psi <- tau.psi <- rep(0,N.resources)
> for (i in 1:N.resources) {
+ mu.psi[i] <- lognPar(m.psi[i],sd.psi[i])$mulog
+ tau.psi[i] <- 1/lognPar(m.psi[i],sd.psi[i])$sigmalog^2
+ }

> # Quality of life weights (N.outcomes = 7)
> # 1. Influenza infection
> # 2. GP visits (no QoL loss)
> # 3. Minor complications (repeat visit, no QoL loss)
> > # 4. Major complications (pneumonia)
> # 5. Hospitalisations
> # 6. Death
> # 7. Adverse events due to vaccination
> m.omega <- c(4.27,0,0,6.35,6.35,15.29,.55)
> sd.omega <- c(1.38,0,0,1.5875,1.5875,3.6,.15)
> mu.omega <- tau.omega <- rep(0,N.outcomes)
> for (i in c(1,4,5,6,7)) {
+ mu.omega[i] <- lognPar(m.omega[i],sd.omega[i])$mulog
+ tau.omega[i] <- 1/lognPar(m.omega[i],sd.omega[i])$sigmalog^2
+ }

```

(notice that the + at the beginning of a line inside the for loops is just R standard notation to indicate commands that span over more than one line).

The very first line of the script executes the file `Utils.R` from its remote location (<http://www.statistica.it/gianluca/BCEABook/WebMaterial/Utils.R>); this file contains a set of functions and commands that are used throughout the script and thus is fundamental to launch it before the rest of the script can be executed. Although the number of files necessary to run the entire analysis may increase (thus, at face value, increasing the complexity of the process), it is actually good programming practice to use a combination of many smaller, focussed scripts, rather than include every commands or functions required in one single, massive file. This, again, makes the process transparent and easier to debug or critically appraise.

The rest of the script defines the values for the parameters used in the distributions associated to the quantities modelled and described above. For example, the function `betaPar2` (which is defined in the file `Utils.R`) can be used to determine the values of the parameters of a Beta distribution so that its average is around 0.436 and 95% of the mass is below the value of 0.6. In particular, running this command on a R terminal gives the following output:

```
> betaPar2(.434, .6, .95)
$res1
[1] 11.30643

$res2
[1] 14.4411

$theta.mode
[1] 0.434

$theta.mean
[1] 0.4391267

$theta.median
[1] 0.437

$theta.sd
[1] 0.09595895
```

`betaPar2` creates a *list* of results: the first two elements of the list, `res1` and `res2` are the estimated values of the parameters to be used with a Beta distribution so that roughly 95% of the probability mass is below 0.6. This is in line with the assumptions presented in Table 2.1 for the parameter ϕ . Again, we can check the appropriateness of this choice by simply typing the following commands to the R terminal.

```
> phi <- rbeta(100000, 11.30643, 14.4411)
> c(quantile(phi, .025), quantile(phi, .5), quantile(phi, .975))
      2.5%      50%      97.5%
0.2566397 0.4371251 0.6296423
```

The other elements of the list are `theta.mode`, `theta.mean`, `theta.median` and `theta.sd`, which store the values for the mode, mean, median and standard deviation

of the resulting Beta distribution. Notice the R “dollar” notation, which can be used to access elements of an object — in other words, if the object *x* is stored in the R workspace and contains the elements *y*, *z* and *w*, then these can be accessed by using the notation *x*\$*y*, *x*\$*z* or *x*\$*w*.

Another thing to notice is that it is fairly easy to annotate the R code in an informative way. This again increases transparency and facilitates the work of reviewers or modellers called upon a critical evaluation of the analysis process. In line with the point we made above about using many simpler and specific files to execute the several steps of the analysis, rather than one large (and potentially messy) file, it is a good idea to save this code to a script file, say *LoadData.R*, again assumed to be stored in the working directory. From within the R terminal, the script can be launched and executed by typing the command

```
> source("LoadData.R")
```

which runs all the instructions in the script sequentially.

2.3.1.4 R Code to Remotely Run BUGS/JAGS and the Bayesian Model

At this point, the user is ready to run the model—in a full Bayesian context, this typically means performing a MCMC analysis (cfr. Sect. 1.2.3) to obtain a sample from the posterior distribution of the random quantities of interest. We reiterate here that these may be unobservable parameters as well as unobserved variables.

R is particularly effective at interfacing with the main software for Bayesian analysis—here we refer to the most popular OpenBUGS [3] and JAGS [4], but there is a R package to interface with a more recent addition, Stan [6]. This means that it is possible to produce a set of scripts that can be run in R to pre-process the data, call the MCMC sampler in the background and run the model (written in a .txt file, as shown above) and then post-process the results, e.g. to obtain the suitable measures of population average costs and effectiveness.

For example, the following commands can be used to run the Bayesian model defined above:

```
> # Loads the package to run OpenBUGS or JAGS from R
> library(R2OpenBUGS)
> library(R2jags)
> # Defines the current as the working directory
> working.dir <- paste(getwd(), "/", sep="")
> # Launches the file Utils.R which contains useful functions used
  throughout this script
> source("http://www.statistica.it/gianluca/BCEABook/WebMaterial/
  Utils.R")
> # Loads the data into R (assumes the file is stored in the
  working directory - if not the full path can be provided)
> source("LoadData.R")

> # Defines the data list to be passed to BUGS/JAGS
> data <- list("N", "a.phi", "b.phi", "mu.rho", "tau.rho", "a.beta", "b.
  beta", "a.gamma", "b.gamma", "mu.omega", "tau.omega", "mu.psi", "tau.
```

```

    psi","N.outcomes","N.resources","mu.lambda","tau.lambda","a.xi",
    "","b.xi","a.eta","b.eta","a.delta")

> # Defines the file with the model code
> filein <- "vaccine.txt"

> # Defines the quantities to be monitored (stored)
> params <- c("beta","phi","omega","rho","Infected","GP","Repeat.
  GP","Pneumonia","Hospital","Death","Mild.Compl","Trt","Adverse
  .events","n","gamma","delta","psi","lambda","pi","xi","eta")

> # Generates the initial values
> inits <- function(){
+ list(phi=runif(1),beta=runif(N.outcomes,0,1),rho=c(NA,runif(1)),
  gamma=runif(2,0,1),delta=rpois(1,2),omega=c(runif(1),NA,NA,
  runif(1),NA,runif(2,0,1)),psi=runif(N.resources,0,10),lambda=
  runif(1),eta=runif(1),xi=runif(1))
+ }

> # Defines the number of iteration, burn-in and thinning, and
  runs BUGS or JAGS
> n.iter <- 100000
> n.burnin <- 9500
> n.thin <- floor((n.iter-n.burnin)/500)

> # 1. This runs OpenBUGS
> vaccine <- bugs(data, inits, params, model.file=filein,n.chains
  =2, n.iter, n.burnin, n.thin, DIC=FALSE, working.directory=
  working.dir)

> # 2. This runs JAGS
> vaccine <- jags(data, inits, params, model.file=filein,n.chains
  =2, n.iter, n.burnin, n.thin, DIC=FALSE, working.directory=
  working.dir, progress.bar="text")

> # Prints the summary stats and attaches the results to the R
  workspace
> print(vaccine,digits=3,intervals=c(0.025, 0.975))

> # In OpenBUGS:
> attach.bugs(vaccine)
> # In JAGS:
> attach.jags(vaccine)

```

For convenience, we can save them in a file, say RunMCMC.R, which can then be run from within the R terminal using the source command.

```
> source("RunMCMC.R")
```

This script proceeds by first loading the relevant packages (which allow R to interface with either OpenBUGS or JAGS); this can be done using the command `library(R2OpenBUGS)` or `library(R2jags)`, depending on the Bayesian software

of choice. Of course, for these to work, either or both OpenBUGS and JAGS need to be installed on the user's machine (we refer interested readers to Sect. 2.2 or the relevant websites, where information is provided on installation and use under different operating systems). In the first part of the script, we also execute the files `Utils.R` and `LoadData.R`, presented above, which prepare the data for either OpenBUGS or JAGS to use. Finally, the current folder is set up as the working directory (but of course, the user can choose any folder for this).

The next step amounts to storing all the relevant input data for the model code into a list. In this case, we need to include all the values for the parameters of the distributions used in the file `vaccine.txt`, which encodes the model assumptions. Then, we instruct R to read the model assumptions from the file `vaccine.txt` and finally we define the “parameters” to be monitored. Again, we note that with this terminology we refer to any unobserved or unobservable quantity for which we require inference in the form of a sample from the posterior distribution.

Before we run OpenBUGS or JAGS we need to define the list of “initial values”, which are used to start the Markov chain(s). Notice that both BUGS or JAGS can randomly generate initial values. However, it is generally better to closely control this process [7]. This can be done by creating a suitable R function that stores in a list random values for all the quantities that need initialisation. These are obtained by specifying the underlying distribution—for instance, in this case we are generating the initial value for ϕ from a `Uniform(0, 1)` distribution (this is reasonable as ϕ is a probability and so it needs to have a continuous value between 0 and 1). In principle, any quantity that is modelled using a probability distribution *and* is not observed needs to be initialised. With reference to the model code presented above, it would not be possible to initialise the node `n[1, 2]`, because it is defined as a deterministic function of other quantities (in this case `N` and `V[2]`).

Finally, we define the total number of iterations, the number of iterations to be discarded in the estimate of the posterior distributions (burn-in) and the possible value of the “thinning”. This refers to the operation of only saving one every l iterations from the Markov Chains. This can help reduce the level of autocorrelation in the resulting chains. For example, we could decide to store 1,000 iterations and obtain this either by saving the last 1,000 runs from the overall process (i.e. by discarding the first 9,000 of the 10,000 iterations produced), or by running the process for 100,000 iterations, discarding the first 9,500 and then saving one every 181 iterations. Of course, the latter alternative involves a longer process just to end up with the same number of samples on which to base the estimation of the posteriors. But the advantage is that it is likely that it will show a lower level of autocorrelation, which means a larger amount of information and thus better precision in characterising the target distributions.

Once these steps have been executed, we can use the commands `bugs` or `jags` to run the model. Both would call the relevant MCMC sampler in the background and produce the MCMC estimates. When the process is finished, the user regains control of the R session. A new object, in this case named `vaccine`, is

created in the current workspace. This object can be manipulated to check model convergence, visualise the summary results (using the `print` method available for both `R2OpenBUGS` and `R2jags`) and save the results (i.e. the simulated values from the posterior distributions) to the R workspace.

For example, a summary table can be obtained as follows (here, we only present the first few rows, for simplicity):

```
> print(vaccine,interval=c(.025,.975),digits=3)
Inference for Bugs model at "vaccine.txt", fit using jags,
 2 chains, each with 10000 iterations (first 9500 discarded), n.thin = 181
 n.sims = 1000 iterations saved
```

	mu.vect	sd.vect	2.5%	97.5%	Rhat	n.eff
Adverse.events	4384.479	2518.102	969.425	10740.800	1.005	310
Death[1,1]	1.573	1.539	0.000	5.000	1.000	1000
Death[2,1]	0.850	1.084	0.000	4.000	1.001	1000
Death[1,2]	0.000	0.000	0.000	0.000	1.000	1
Death[2,2]	0.248	0.545	0.000	2.000	1.000	1000
GP[1,1]	2045.987	896.964	654.925	4092.150	1.000	1000
GP[2,1]	1148.308	543.198	340.925	2435.475	1.000	1000
GP[1,2]	0.000	0.000	0.000	0.000	1.000	1
GP[2,2]	279.658	151.580	78.000	658.325	1.000	1000
Hospital[1,1]	0.764	0.959	0.000	3.000	1.001	1000
Hospital[2,1]	0.438	0.698	0.000	2.000	1.002	620
...						

For each parameter included in the list of quantities to be monitored, this table shows the mean and standard deviation (the columns labelled as `mu.vect` and `sd.vect`), together with the 2.5 and 97.5% quantiles of the posterior distributions (which give a rough approximation of a 95% credible interval).

The final columns of the table (indexed by the labels `Rhat` and `n.eff`, respectively) present some important convergence statistics. The first one is the potential scale reduction \hat{R} , often termed the Gelman–Rubin statistic. This quantity can be computed when the MCMC process is based on running at least two parallel chains and basically compares the within to the between chain variability. The rationale is that when this ratio is close to 1, then there is some evidence of “convergence” because all the chains present similar variability and do not vary substantially among each other, thus indicating that they are all visiting a common area in the parameter’s space. Typically, values below the arbitrary threshold of 1.1 are considered to suggest convergence to the relevant posterior distributions.

The second one is the “effective sample size” n_{eff} . The idea behind this quantity is that the MCMC analysis is based on a sample of `n.sims` iterations (in this case, this is 1,000). Thus, if these were obtained using a sample of independent observations from the posterior distributions, this would be worth exactly 1,000 data points. However, because MCMC is a process in which future observations depends on the current one, there is some intrinsic “autocorrelation”, which means that often a sample of

S iterations has a value in terms of information that is actually lower. This value is quantified by the effective sample size. When $n.\text{eff}$ is close to $n.\text{sims}$, this indicates that the level of autocorrelation is low and that in effect the $n.\text{sims}$ points used to obtain the summary statistics are worth more or less their nominal value. On the other hand, when the two are very different this indicates that the MCMC sample contains less information about the posterior.

For example, because of the autocorrelation, the 1,000 simulations used to characterise the posterior distribution of the node `Adverse.events` are actually equivalent to a sample made by around 310 independent observations from that posterior. In cases such as this, when $\hat{R} < 1.1$ but $n.\text{eff}$ is much smaller than $n.\text{sims}$ we could conclude that the sample obtained has indeed converged to the posterior distribution but does not contain enough information to fully characterise it. For example, the mean and the central part of the distribution may be estimated with good precision, but the tails may not. One easy (albeit potentially computationally intensive) workaround is to run the MCMC for a (much) longer run and possibly increase the thinning.

Additional analyses to check on convergence may be performed, for example by providing traceplots of the chains, e.g. as in Fig. 1.3(d), for example using the following command

```
> traceplot(vaccine)
```

which produces an interactive traceplot for each of the monitored nodes. More advanced graphing and analysis can be done by subsetting the object `vaccine` and accessing the elements stored therein. Details on how to do this are shown, for example, in [7].

2.3.2 Economic Model

In order to perform the economic analysis, we need to define suitable summary measures of cost and effectiveness. The total cost associated with each clinical resource can be computed by multiplying the unit cost ψ_h by the number of patients consuming it. For instance, the overall cost of doctor visit is $(GP_{tv}^{(1)} + GP_{tv}^{(2)}) \times \psi_1$. If, for convenience of terminology, we indicate with N_{tvh} the total number of individuals consuming the h -th resource under intervention t and in group v , we can then extend this reasoning and compute the **average population cost** under intervention t as

$$c_t := \frac{1}{N} \sum_{v=0}^1 \sum_{h=1}^8 N_{tvh} \psi_h. \quad (2.1)$$

Similarly, the total QALYs lost due to the occurrence of the relevant outcomes can be obtained by multiplying the number of individuals experiencing them by ω_j . For example, the total number of QALYs lost to influenza infection can be computed as $I_{tv} \times \omega_1$. If we let M_{tvj} indicate the number of subjects with the j -th outcome

in intervention t and group v , we can define the **population average measure of effectiveness** for intervention t as

$$e_t := \frac{1}{N} \sum_{v=0}^1 \sum_{j=1}^7 M_{tvj} \omega_j. \quad (2.2)$$

The results of the MCMC procedure used to run the model described above can be obtained by simply running the scripts discussed in Sect. 2.3.1. However, they are also available in the R object `vaccine.RData`, which can be directly downloaded at <http://www.statistica.it/gianluca/BCEABook/vaccine.RData>. For example, this can be uploaded to the R session by typing the following command:

```
> load("http://www.statistica.it/gianluca/BCEABook/vaccine.RData")
> ls()
[1] "Adverse.events" "Death"      "GP"          "Hospital"
[5] "Infected"      "N"          "Pneumonia"   "Repeat.GP"
[9] "delta"         "eta"        "gamma"       "lambda"
[13] "n"             "n.sims"     "omega"       "psi"
[17] "xi"
```

Each of these R objects contains $n_{\text{sims}} = 1000$ simulations from the relevant posterior distributions. Before the economic analysis can be run, it is necessary to define the measures of overall cost and effectiveness given in Eqs. (2.1) and (2.2), respectively. This can be done using the results produced by the MCMC procedure with the following R code. Notice that since the utilities are originally defined as quality adjusted life days, it is necessary to rescale them to obtain QALYs.

```
> ## Compute effectiveness in QALYs lost for both strategies
> 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)
```

The notation `Infected[,t,1]` indicates all the simulations (the first dimension of the array) for the t -th intervention (which the `for` loop sets sequentially to 1 and 2 to indicate $t = 0, 1$, respectively) and for the first vaccination group. Similarly, `Infected[,t,2]` indicates all the simulations for the t -th intervention and for the second vaccination group. Thus, each of these two elements effectively produces the value M_{tv1} (where $j = 1$ indicates the first outcome) and consequently, the code `((Infected[,t,1] + Infected[,t,2])*omega[,1]/365)/N` does identify

the quantity $\frac{1}{N} \sum_{v=0}^1 M_{tv1} \omega_1$. Following a similar reasoning for all the other outcomes and summing them all up, we do obtain the measure of effectiveness, which is stored in a matrix `e` with n_{sims} rows and 2 columns (one for each intervention considered).

We can follow a similar strategy to identify the costs associated with each intervention. First we define the number of “users” (which we indicated earlier as N_{tvh} and according to the resource depends on the number of doctor (general practitioner) visits, hospitalisations, infections, repeated hospitalisations, or individuals at risk); then we multiply these by the associated cost (contained in the variable `psi`). Then we sum all the components to derive the overall average cost for each treatment strategy.

```
> ## Compute costs for both strategies
> cost.GP <- cost.hosp <- cost.vac <- cost.time.vac <- cost.time.off <- cost.
  trt1 <- cost.trt2 <- cost.otc <- cost.travel <- matrix(0,n.sims,2)
> for (t in 1:2) {
  cost.GP[,t] <- (GP[,t,1]+GP[,t,2]+Repeat.GP[,t,1]+ Repeat.GP[,t,2])*psi
    [,1]/N
  cost.hosp[,t] <- (Hospital[,t,1]+Hospital[,t,2])*psi[,2]/N
  cost.vac[,t] <- n[,2,t]*psi[,3]/N
  cost.time.vac[,t] <- n[,2,t]*psi[,4]/N
  cost.time.off[,t] <- (Infected[,t,1]+Infected[,t,2])*psi[,5]*eta*lambda/N
  cost.trt1[,t] <- (GP[,t,1]+GP[,t,2])*gamma[,1]*psi[,6]*delta/N
  cost.trt2[,t] <- (Repeat.GP[,t,1]+Repeat.GP[,t,2])*gamma[,2]*psi[,6]*delta/
    N
  cost.otc[,t] <- (Infected[,t,1]+Infected[,t,2])*psi[,7]*xi/N
  cost.travel[,t] <- n[,2,t]*psi[,8]/N
}
> c <- cost.GP + cost.hosp + cost.vac + cost.time.vac + cost.time.off + cost.
  trt1 + cost.trt2 + cost.travel + cost.otc
```

At this point we are ready to run the Decision Analysis and the Uncertainty Analysis, which BCEA can take care of. We present these parts in Chaps. 3 and 4.

2.4 Smoking Cessation

In this example, we will consider a set $\mathcal{T} = \{t = 0, 1, 2, 3\}$ of $T = 4$ potential interventions to help smoking cessation; in particular, $t = 0$ represents no contact (status quo); $t = 1$ is a self-help intervention; $t = 2$ is individual counselling; and $t = 3$ indicates group counselling. The interest is in the joint evaluation of the T interventions. The analysis will be conducted in the form of a cost-consequence analysis, and as such costs and health effects will not be correlated as it usually happens in the wider framework of a cost-utility analysis. However, it should be noted that there

is no substantive difference between a cost-consequence and cost-utility analysis, as argued in [8]. Therefore, the costs and health effects will be analysed separately and then jointly to produce a summary of the comparative cost-effectiveness profiles of the interventions considered.

The available clinical evidence is made by a set of trials in which some combinations of the available interventions have been considered. However, not all the possible pairwise comparisons are observed. The use of a Bayesian model embedding some suitable exchangeability assumptions allows the estimation of a suitable measure of effectiveness for *all* the interventions (using all the available evidence for each $t \in \mathcal{T}$) and for *all* the possible pairwise comparisons. This is a meta-analysis technique generally referred to as Mixed Treatment Comparison (MTC), which expands the concepts of Bayesian evidence synthesis to generate a network of evidence that can be used to produce the required estimations. More detailed discussion is presented in [9] and [10]. The data used in this example were originally reported in [11].

2.4.1 (Bayesian) Statistical Model

Assumptions

The dataset includes $N = 50$ data points nested within $S = 24$ studies. For each study arm $i = 1, \dots, N$ we observe a variable r_i indicating the number of patients quitting smoking out of a total sample size of n_i individuals. In addition, we also record a variable t_i taking on the possible values 1, 2, 3, 4, indicating the treatment associated with the i -th data point. The nesting within the trial is accounted for by a variable s_i taking values in $1, \dots, S$.

Most studies are simple head-to-head comparisons (i.e. comparing only two interventions), while two of the them are multi-arm trials (the first one involving $t = 1, 3, 4$, and the second one comparing $t = 2, 3, 4$). Most trials compare one of the active treatments $t = 2, 3, 4$ against the control treatment “No intervention”. Five of the studies consider comparisons between two or more active treatments. The full dataset is presented in Table 2.2.

Figure 2.1 shows the description of the “network” of data available—the process of combining this information into a consistent framework is often referred to as “Network Meta-Analysis” (NMA).

For each study arm we model the number of observed quitters as the realisation of a Binomial random variable:

$$r_i \sim \text{Binomial}(p_i, n_i)$$

where p_i is the specific probability of smoking cessation. The main objective of the model is to use the available data to derive a pooled estimation for π_t , the intervention-specific probability of smoking cessation.

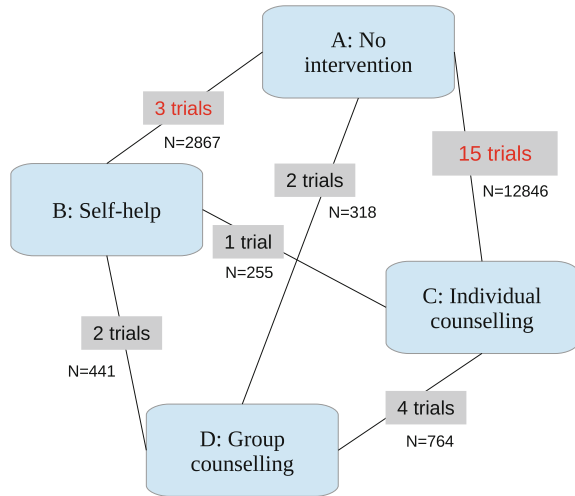
Table 2.2 The dataset containing information on the $S = 24$ trials on smoking cessation. The data were originally reported in [11]

Study (s_i)	Intervention (t_i)	Quitters (r_i)	Participants (n_i)	Comparator (c_i)
1	1	9	140	1
1	3	23	140	1
1	4	10	138	1
2	2	11	78	2
2	3	12	85	2
2	4	29	170	2
3	1	75	731	1
3	3	363	714	1
4	1	2	106	1
4	3	9	205	1
5	1	58	549	1
5	3	237	1561	1
6	1	0	33	1
6	3	9	48	1
7	1	3	100	1
7	3	31	98	1
8	1	1	31	1
8	3	26	95	1
9	1	6	39	1
9	3	17	77	1
10	1	79	702	1
10	2	77	694	1
11	1	18	671	1
11	2	21	535	1
12	1	64	642	1
12	3	107	761	1
13	1	5	62	1
13	3	8	90	1
14	1	20	234	1
14	3	34	237	1
15	1	0	20	1
15	4	9	20	1
16	1	8	116	1
16	2	19	149	1
17	1	95	1107	1
17	3	143	1031	1
18	1	15	187	1

(continued)

Table 2.2 (continued)

Study (s_i)	Intervention (t_i)	Quitters (r_i)	Participants (n_i)	Comparator (c_i)
18	3	36	504	1
19	1	78	584	1
19	3	73	675	1
20	1	69	1177	1
20	3	54	888	1
21	2	20	49	2
21	3	16	43	2
22	2	7	66	2
22	4	32	127	2
23	3	12	76	3
23	4	20	74	3
24	3	9	55	3
24	4	3	26	3

Fig. 2.1 A graphical representation of the network of evidence for the smoking cessation studies

We use the following strategy. First we model the probabilities p_i using a structured formulation

$$\text{logit}(p_i) = \mu_{s_i} + \delta_{s_i, t_i} (1 - \mathbb{I}\{t_i = b_{s_i}\}) .$$

The parameter μ_{s_i} represents a study-specific baseline value, which is common to all interventions being compared in study s_i . Notice that for each $i = 1, \dots, N$, s_i takes on the integer values in $[1; S]$. Thus the vector $\boldsymbol{\mu}$ comprises of S elements.

The parameter δ_{s_i, t_i} represents the incremental effect of treatment t_i with respect to the reference intervention being considered in the study s_i . Specifically, we assume by common convention that the intervention associated with the minimum label value found in each study, is the reference intervention for that study. This formulation allows for a clear specification of study-specific effects and can be easily extended to include study-treatment interaction. The reference (or baseline) intervention for each study is indicated by b_{s_i} ; thus $\delta_{s_i, b_{s_i}} = 0$, with the effect of the baseline intervention for each study s represented by μ_s . Consequently, in each study s we assume that the comparator's effect is the study baseline and that the incremental effect of treatment t is represented by $\delta_{s, t}$ if $t \neq b_s$.

The parameters in μ are given independent minimally informative Normal distributions $\mu_s \stackrel{iid}{\sim} \text{Normal}(0, v)$, where v is a large fixed value identifying the initial value of the variance of the distributions. On the contrary, we assume that the parameters δ_{s_i, t_i} represent “structured” effects

$$\delta_{s_i, t_i} \sim \text{Normal}(md_i, \sigma^2)$$

with

$$md_i = d_{t_i} - d_{b_{s_i}}.$$

The parameters $\mathbf{d} = (d_1, \dots, d_T)$ represent some pooled intervention-specific effect and the mean md_i is computed as the average difference between the effect for the intervention in row i and the effect for the reference intervention b_{s_i} in study s_i . We assume that $d_1 = 0$, i.e. that the reference intervention has no effect other than the baseline level, while we model $d_i \stackrel{iid}{\sim} \text{Normal}(0, v)$ for $i = 2, \dots, T$.

The parameters \mathbf{d} are defined on the logit scale, and thus in order to compute the estimated probability of smoking cessation on the natural scale for each treatment we need to rescale them. We proceed by estimating the effect for the baseline intervention $t = 1$. Since d_1 was set to 0, the treatment effect π_0 on the logit scale is given by the average of the baseline effects in the trials including the intervention $t = 1$. The treatment effects $\pi_t, t = 1, \dots, T$ are calculated as:

$$\pi_0 = \frac{1}{\sum_{s=1}^S \mathbb{I}\{b_s = 1\}} \sum_{s: b_s=1} \mu_s$$

$$\text{logit}(\pi_t) = \pi_0 + d_t, \quad t = 1, \dots, T$$

where $s : b_s = 1$ indicates the subset of studies including the reference intervention arm $t = 1$ as a comparator. The expression $\sum_{s=1}^S \mathbb{I}\{b_s = 1\}$ indicates the number of studies including treatment $t = 1$, since it is the sum of the indicator function over the trials including that intervention as the baseline comparator.

2.4.1.1 Running the MTC Model in JAGS

We run the MTC model in JAGS (although the code presented below will also work in OpenBUGS with only minor modifications—cfr. Sect. 2.3.1.4). To run the Bayesian evidence synthesis model, it is necessary to store the model specification in a text file that will be then interpreted by JAGS. This file contains the description of the Bayesian model in terms of the stochastic and deterministic relationships between the variables building the model network or graph (more precisely, a *direct acyclic graph*, or DAG).

The model is an adaptation from the specifications reported by Welton et al. (2012) and the NICE Decision Support Unit (2013) [9, 10]. The JAGS code used for the analysis of the smoking cessation data is shown below:

```
### JAGS model ###
model{
  for(i in 1:nobs){
    r[i]~dbin(p[i],n[i])
    p[i] <- ilogit(mu[s[i]]+delta[s[i],t[i]])
    delta[s[i],t[i]] ~ dnorm(md[i],tau)
    md[i] <- d[t[i]]-d[b[s[i]]]
  }
  for(i in 1:ns){
    mu[i]~dnorm(0,.0001)
    AbsTrEf[i] <- ifelse(b[i]==1,mu[i],0)
  }
  pi0 <- sum(AbsTrEf[])/incb
  tau <- pow(sd,-2)
  sd~dunif(0.00001,2)
  d[1] <- 0
  for(k in 2:nt){
    d[k]~dnorm(0,.0001)
  }
  for(j in 1:nt){
    logit(pi[j]) <- pi0+d[j]
    for(k in 1:nt){
      lor[j,k] <- d[j]-d[k]
      log(or[j,k]) <- lor[j,k]
      rr[j,k] <- pi[j]/pi[k]
    }
  }
}
```

To run the analysis it is necessary to save the model in a plain text file. No specific extensions are required and in this example we will save the file with the name `smoking_model_RE.R` in the directory from which we run R. We will assume that the csv file containing the data inputs (i.e. `smoking_data.csv`) is in the same folder.

The directory R is using can be displayed using the command `getwd()` and can be modified by specifying the desired address as the argument of the function `setwd`, i.e. `setwd("PATH_TO_NEW_DIRECTORY")`.

It is necessary to import the data into R and to pre-process the inputs prior to running the Bayesian model. This can be done by running the following code:

```
> # load the R2jags package and the the data file
> library(R2jags)
> smoking=read.csv("smoking_data.csv",header=TRUE)

> # specify the name of the model file
> model.file="smoking_model_RE.R"

> # copy smoking data.frame columns to local variables
> attach(smoking)
> nobs=nobs; s=s; t=t; r=r_i; n=n_i; b=b_i+1
> detach(smoking)

> # number of trials
> ns=length(unique(s))
> # number of comparators
> nt=length(unique(t))
> # number of observations
> nobs=dim(smoking)[1]
> # how many studies include baseline
> incb=sum(table(s,b[,1])>0)
```

The package `R2jags` is necessary to connect R and JAGS, and is loaded with the command `library(R2jags)`. The command `read.csv` is used to read into R the data inputs contained in the csv file `smoking_data.csv`, which will be saved as a `data.frame` object. Since the quantities need to be available in the R workspace, they are saved as new R variables. The baseline treatment is incremented by one when saving it with the command `b=b_i+1`, so that the comparator $t = 0$ (no intervention) is associated with the index 1, the intervention $t = 1$ (self-help) with the index 2, and so on. This is because both R and JAGS index arrays with the first element starting from 1 (as opposed to 0). The total number of studies, the arm index for each observation in the respective trial, the number of comparators and observations and the number of trials including the baseline reference treatment, in this case $t = 0$ (no intervention), are also calculated from the data.

The `jags` function used to run the Bayesian evidence synthesis model requires several inputs:

- `data`: a named list including all the inputs needed by the model;
- `inits`: a list of initial values or a function generating the initial values for (a subset of) the stochastic parameters in the model. In this example, we set `inits` to `NULL`, which means that JAGS will choose at random the initial values for all the parameters in the model. The initial values of the parameters will be randomly drawn from the space of values they can assume, determined by their stochastic definition;
- `parameters.to.save`: a vector of variables to monitor, i.e. the parameters of interest. JAGS will save the output of the simulations from the associated posterior distributions only of the monitored parameters;

- `model.file`: the name or address of the file containing the model. Since we previously saved the model in the R as the file `smoking_model_RE.R`, the name of this file will be the value passed to this argument;
- `n.chains`: the number of parallel Markov chains to run. It is highly recommended that these are at least 2, to allow for checking the convergence and the mixing of the chains;
- `n.iter`: the number of iterations to perform for each chain from initialisation;
- `n.thin`: the thinning rate, i.e. after how many iterations a single value from the posterior distribution is saved, discarding the others;
- `n.burnin`: the length of the burn-in, i.e. the number of simulations to discard after the initialisation of the chains before saving any value. If not specified as in this case, by default it is set to `n.iter/2`.

More details on how to run a JAGS model and then post-process its results for the purposes of health economic analysis are given in [7].

At this point, all the necessary data inputs have been pre-processed and it is possible to run the MTC analysis model:

```
> # define data and parameters to monitor
> inputs=list("s","n","r","t","ns","nt","b","nobs","incb","na")
> pars=c("rr","pi","p","d","sd","T")

> smoking_output <- jags(data=inputs, inits=NULL, parameters.to.save=pars,
  model.file=model.file, n.chains=2, n.iter=10000, n.thin=10)
```

The `jags` function will save the output of the model in the `rjags` object which we called `smoking_output`. A summary of the model results can be printed out by executing the following line of code:

```
> print(smoking_output)
Inference for Bugs model at "smoking_model_RE.txt", fit using jags,
2 chains, each with 20000 iterations (first 10000 discarded), n.thin = 10
n.sims = 2000 iterations saved
```

	mu.vect	sd.vect	2.5%	25%	50%	75%	97.5%	Rhat	n.eff
d[1]	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	1
d[2]	0.499	0.395	-0.278	0.245	0.501	0.757	1.306	1.001	2000
d[3]	0.843	0.239	0.383	0.684	0.833	0.995	1.338	1.000	2000
d[4]	1.107	0.446	0.248	0.817	1.094	1.391	2.011	1.001	2000
pi[1]	0.062	0.012	0.041	0.054	0.061	0.069	0.086	1.002	1000
pi[2]	0.100	0.031	0.053	0.078	0.096	0.117	0.172	1.001	2000
pi[3]	0.132	0.021	0.096	0.118	0.131	0.145	0.174	1.003	730
pi[4]	0.169	0.051	0.087	0.134	0.164	0.199	0.287	1.001	2000
rr[1,1]	1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000	1
rr[2,1]	1.685	0.635	0.774	1.257	1.585	2.003	3.232	1.001	2000
rr[3,1]	2.200	0.497	1.416	1.858	2.129	2.469	3.346	1.000	2000
rr[4,1]	2.878	1.181	1.254	2.090	2.657	3.432	5.677	1.001	2000
rr[1,2]	0.676	0.252	0.309	0.499	0.631	0.795	1.292	1.001	2000
rr[2,2]	1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000	1
rr[3,2]	1.454	0.551	0.672	1.054	1.366	1.726	2.814	1.001	2000
rr[4,2]	1.849	0.814	0.727	1.291	1.691	2.253	3.865	1.001	2000
rr[1,3]	0.476	0.103	0.299	0.405	0.470	0.538	0.706	1.000	2000
rr[2,3]	0.785	0.295	0.355	0.579	0.732	0.949	1.488	1.001	2000
rr[3,3]	1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000	1
rr[4,3]	1.324	0.489	0.596	0.989	1.249	1.559	2.464	1.001	1700
rr[1,4]	0.403	0.159	0.176	0.291	0.376	0.478	0.798	1.001	2000
rr[2,4]	0.646	0.288	0.259	0.444	0.591	0.774	1.375	1.001	2000
rr[3,4]	0.856	0.316	0.406	0.641	0.801	1.011	1.677	1.001	1700
rr[4,4]	1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000	1
sd	0.601	0.128	0.395	0.510	0.589	0.676	0.896	1.007	2000
deviance	281.227	9.952	263.929	274.289	280.731	287.548	302.935	1.001	2000

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 = 49.5 and DIC = 330.8
DIC is an estimate of expected predictive error (lower deviance is better).
```

The table above reports for each monitored variable the estimated mean, standard deviation, several percentiles of the posterior distributions (2.5, 25, 50, 75 and 97.5%), the convergence Gelman–Rubin \hat{R} diagnostic and the effective sample size n_{eff} . The latter gives an indication of the presence of autocorrelation within the chains by quantifying the information contained in the vector of simulations used to estimate every parameter. The percentiles can be used to approximate a credibility interval (CrI), which is an interval of values containing a posterior probability mass equal to 0.95; assuming the unimodality of the distributions (as it is the case), the posterior 95% CrI can be approximated by taking the 2.5 and 97.5% percentiles as the lower and upper bound, respectively.¹

The diagnostics measures indicate a good convergence of the model, with the Gelman–Rubin \hat{R} statistics being below 1.1 for all the measures. In addition the effective sample size does not signal the presence of autocorrelation within the chains.

From this output we can observe that the most effective treatment is $t = 4$ (Group counselling) with an associated probability of patients quitting smoking equal to $\pi_4 = 0.17$ (95% CrI [0.09; 0.29]). It is followed by: $t = 3$ (Individual counselling) associated with a probability of quitting of $\pi_3 = 0.13$ (95% CrI [0.10; 0.17]); $t = 2$ (Self-help) with a probability of $\pi_2 = 0.10$ (95% CrI [0.05; 0.17]); and lastly $t = 1$ (No intervention) with an estimated probability of quitting equal to 0.06 (95% CrI [0.04; 0.09]). The results are represented graphically in Fig. 2.2. It can be observed that the uncertainty associated with the effect size of group counselling is high, with a 95% credible interval wider than the ones for the other interventions.

The plot in Fig. 2.2 can be reproduced using the following code:

```
> attach.jags(smoking_output)
> tr.eff=data.frame(t(apply(pi,2,quantile,c(0.025,0.975))))
> names(tr.eff)=c("low","high")
> treats=c("No intervention","Self-help","Individual counselling",
           "Group counselling")
> tr.eff=cbind(tr.eff,mean=smoking_output$BUGSoutput$mean$pi,
              interventions=factor(treats,levels=treats))
> detach.jags()
```

¹It should be noted that the estimation of the “effective number of parameters” p_D is controversial. The definition reported in [12] and in [3], which is also the one adopted in BUGS, should be preferred instead of the one reported by R2jags [13]. This statistic is calculated by R2jags as:

$$p_D = \text{Var}[\bar{D}_{\text{model}}]/2$$

while both [12] and in [3] report that the preferred definition is:

$$p_D = \bar{D}_{\text{model}} - D(\hat{\theta})$$

where \bar{D}_{model} is the posterior deviance of the model and $D(\hat{\theta})$ is the deviance in correspondence of the estimated posterior mean of the vector of parameters θ . It should be noted that the definition of p_D has a direct impact on the deviance information criteria (DIC), which is an index commonly used for model comparisons, defined as $\text{DIC} = \bar{D}_{\text{model}} + p_D = D(\hat{\theta}) + 2p_D$.

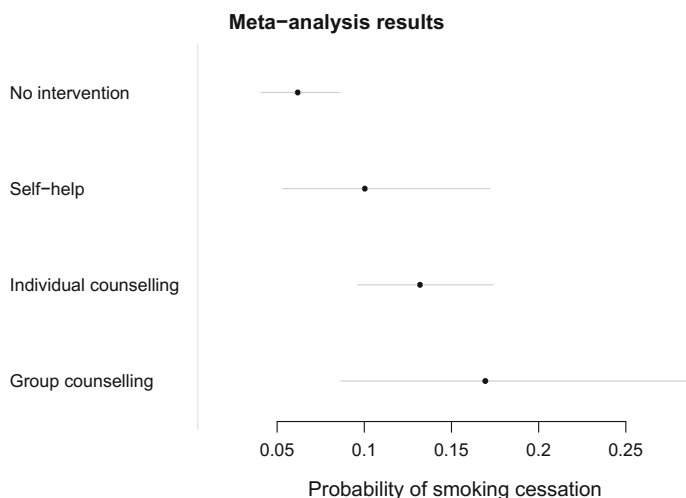


Fig. 2.2 Mean and 95% credible intervals (CrI) of the re-estimated treatment effects of each treatment. Group counselling resulted in having the best estimated efficacy, followed by individual counselling, self-help and no intervention. The credible interval associated with the group counselling estimate was substantially wider than the ones for the other comparators

```
> ggplot(tr.eff) + geom_point(aes(y=mean,x=interventions)) +
  geom_errorbar(aes(x=interventions,ymin=low,ymax=high),width
    =0.15) + coord_flip() + theme_bw() + labs(x="",y="Probability
    of smoking cessation",title="Meta-analysis results")
```

The simulations from the posterior distributions of the parameters that will be used in the economic model are stored in the `BUGSoutput` element of the `rjags` output object. The vectors of simulations can be attached to the current workspace by using the command `attach.jags`, which makes the values available in the workspace.² As the economic model will be based on the 2,000 values obtained in JAGS, it is necessary to extract these values from the output object. In the following code, we attach the JAGS output to the workspace and copy the values simulated from the posterior distributions of the estimated probability of cessation for each treatment $\pi = (\pi_1, \pi_2, \pi_3, \pi_4)$ in a 2000×4 matrix `pi`. The latter will be used as inputs for the economic model.

```
> attach.jags(smoking_output)
> pi <- pi
```

²When using OpenBUGS and R2OpenBUGS, the object can be attached to the R workspace using the command `attach.bugs(object)` or `attach.jags(object)`, respectively.

2.4.2 *Economic Model*

Similarly to the Vaccine example, we need now to include other variables and, generally, post-process the output of the Bayesian model to obtain the quantities necessary to perform the Decision and Uncertainty Analyses.

For example, in this case no data on the costs were provided in addition to the effectiveness reported in [11]. Thus, for the purposes of this example, we extracted information published in [14], who reported costs for different class of interventions for smoking cessation. The costs in British pounds were taken from [15]. Although the interventions reported in [11] were not described in detail, a comparison of the meta-analysis results with the comparative efficacy measures given in [14] showed consistent results, indicating substantial similarity between the interventions in the two studies.

The costs for the comparators included in the analysis are composed as follows:
No intervention:

- No costs: £0;

Self-help:

- Nicotine replacement therapy (NRT) for five weeks (35 patches at £1.30 each);

Individual counselling:

- NRT for five weeks (35 patches at £1.30 each);
- Five clinic visits (£10.00 each);

Group counselling:

- NRT for five weeks (35 patches at £1.30 each);
- Five group visits (£19.46 each).

The total average costs per intervention were: £0 for $t = 0$; £45.50 for $t = 1$; £95.50 for $t = 2$; and £142.80 for $t = 3$. Due to the expected variability associated to the compliance to the interventions in general practice and to the potential need of additional counselling and pharmacological treatment for some patient, it is reasonable to describe the uncertainty associated with the costs with a probability distribution.

For simplicity, a triangular distribution is associated with all treatment costs (excluding the reference “No intervention” comparator), with limits defined by the average intervention cost $\pm 30\%$. The triangular distribution is a triangle-shaped curve with a null associated density of probability outside the specified lower and upper bounds. It increases linearly from the lower bound to its mode, and decreases linearly up to the upper limit. A graphical representation is given in Fig. 2.3. A real-world analysis could be based on more appropriate assumptions for the cost distributions.

The distributions of the costs need to be simulated to be inputted in the cost-effectiveness model. The reference comparator $t = 0$ is assumed not to have an associated costs, i.e. its cost is always null. In formal terms, a degenerate probability distribution which assumes the value zero with probability equal to one is

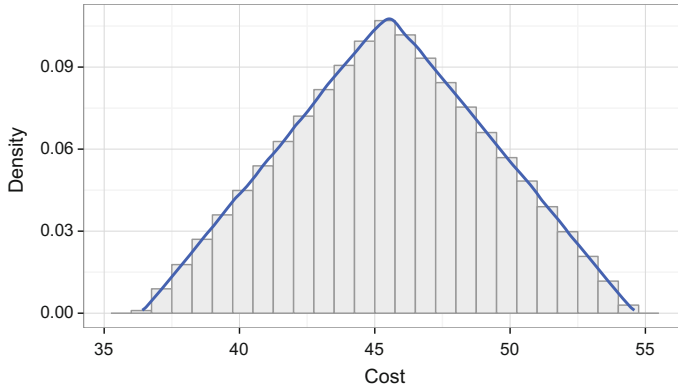


Fig. 2.3 The distribution represents the uncertainty associated with the costs for the self-help intervention. The curve is shaped as a triangle, hence its name. In this case the mean is equidistant to the lower and upper bound, thus corresponding to the mean (and median) of the distribution

assigned to this parameter. The costs for the other interventions are simulated from the intervention-specific triangular distributions described above. Functions to sample from a triangular distribution are not included in the default libraries of R, thus the `triangle` package needs to be installed to use the following code. The package is available on CRAN, and can be installed as usually from a GUI or by inputting the following command:

```
> install.packages("triangle")
```

The code to obtain the simulated values from the probability distributions of the costs, stored in the cost matrix, is presented below. Since we populated the matrix with zeroes when creating the object, the costs for $t = 0$ are automatically assigned. The function `rtriangle` accepts as arguments the number of simulations needed, the lower bound of the distribution a and the upper bound b . If not specified, the mode of the distribution c is calculated by default as the average of the two extremes; since we are using symmetric distributions and thus the mode corresponds to the mean, there is no need to specify this parameter.

Table 2.3 Life expectancy increments gained by smoking cessations per gender and age at quitting. Source: [16]

Life years gained relative to continuing smokers		
Age at quitting	Men	Women
35	8.5	7.7
45	7.1	7.2
55	4.8	5.6
65	4.6	5.1

Table 2.4 Proportion of smokers per age group. The data on smoking statistics have been published by the charity *Action on Smoking and Health* in October 2013, reporting the prevalence of cigarette smoking in the UK. Source [17]

Age group	Proportion of smokers (%)
16–19	15
20–24	29
25–34	27
35–49	23
50–59	21
60+	13

```

> library(triangle)
> cost.t1=45.5
> cost.t2=95.5
> cost.t3=142.8
> c=matrix(data=0,nrow=2000,ncol=4)
> c[,2]=rtriangle(2000,a=cost.t1*.8,b=cost.t1*1.2)
> c[,3]=rtriangle(2000,a=cost.t2*.8,b=cost.t2*1.2)
> c[,4]=rtriangle(2000,a=cost.t3*.8,b=cost.t3*1.2)

```

As for the measure of effectiveness, we can use data from [16] on the increments in life expectancy gained. The authors analysed the increments in life expectancy observed in a US survey, reported in Table 2.3.

Data on the prevalence of smoking in the British setting were obtained from the 2013 report of the charity *Action on Smoking and Health* (ASH) [17]. The data have been summarised in Table 2.4. A split by both gender and age was not included, but the overall proportions of men and women smoking were reported as 22 and 20% in 2012, respectively. This means that the proportion of men among smokers was 52%. It has been assumed that the proportion was not different among the age groups due to lack of data.

The distribution by age of the population in the UK was taken from the 2011 census by the Office of National Statistics (ONS). The data tables are available from the ONS website.³

The data for the life years gained reported in [16] did not include individuals younger than 35 or older than 65 years. For simplicity, we assume here that the gain for quitters younger than 35 years was the same observed for the quitters at 35 years and that individuals aged 65–80 years quitting had the same gain as 65 years old quitters (Table 2.5).

The average life expectancy was calculated using a simulation-based approach. For each of the 2,000 simulations, 1,000 smoking individuals were drawn from the distribution of smokers per age, calculated on the basis of the age distribution in the UK and the proportion of smokers per age group. The gender was simulated from

³At the address <http://www.ons.gov.uk/ons/publications/re-reference-tables.html?edition=tcn%3A77-270247>.

Table 2.5 Data inputs for the simulation of life years gained by smoking cessation. The dataset is contained in the file `smoking_cessation_simulation.csv`

Age	Population	Proportion of smokers	Smokers	Proportion of age group	Male life years	Female life years
15–19	3,997,000	0.15	599,550	0.08	8.50	7.70
20–24	4,297,000	0.29	1,246,130	0.09	8.50	7.70
25–29	4,307,000	0.27	1,162,890	0.09	8.50	7.70
30–34	4,126,000	0.27	1,114,020	0.08	8.50	7.70
35–39	4,194,000	0.23	964,620	0.09	8.50	7.70
40–44	4,626,000	0.23	1,063,980	0.09	7.10	7.20
45–49	4,643,000	0.23	1,067,890	0.09	7.10	7.20
50–54	4,095,000	0.21	859,950	0.08	4.80	5.60
55–59	3,614,000	0.21	758,940	0.07	4.80	5.60
60–64	3,807,000	0.13	494,910	0.08	4.60	5.10
65–69	3,017,000	0.13	392,210	0.06	4.60	5.10
70–74	2,463,000	0.13	320,190	0.05	4.60	5.10
75–79	2,006,000	0.13	260,780	0.04	4.60	5.10

a Binomial model based on the split reported in the ASH smoking statistics and the life years reported in [16] were assigned.

To obtain the 2,000 simulations from the posterior distribution of the average life years gained by quitters, the code below has been used. Notice that, in order to use the code, the file `smoking_cessation_simulation.csv` needs to be available in the same directory from which R is run, or the correct address to the file needs to be specified. Each of the 1,000 individuals in the cohorts are associated with a simulated age. This is drawn from a multinomial distribution with a vector of probabilities equal to the observed frequency for each age group. The gained life years are calculated for each group based on the gender split. The results are then averaged over the sample, to obtain a vector composed by 2,000 elements. To repeat the process 4 times, obtaining 2,000 simulations for each treatment, 8,000 samples from the multinomial distribution are taken. These are successively arranged in a matrix with 2,000 rows and 4 columns.

```
> data=read.csv(file="smoking_cessation_simulation.csv")
> life.years=with(data,rmultinom(2000*4,1000,pr.age) *
  (.52*Male.ly +.48*Female.ly))
> life.years=matrix(apply(life.years,2,sum)/1000,
  nrow=2000, ncol=4)
```

At this point it is possible to obtain the life years gained for each intervention. It is only necessary to multiply the probability of smoking cessation π for each treatment by the average number of life years gained by quitting. This can be obtained by a multiplication of the two quantities:

```
> e=pi*life.years
```

Again, this process is completed by running BCEA and performing the Decision and Uncertainty Analysis (as described in details in Chaps. 3 and 4).

References

1. S. Petrou, A. Gray, *Brit. Med. J.* **342** (2011), <http://dx.doi.org/10.1136/bmj.d1766>
2. G. Lu, A. Ades, *Stat. Med.* **23**, 3105 (2004)
3. D. Lunn, C. Jackson, N. Best, A. Thomas, D. Spiegelhalter, *The BUGS Book—A Practical Introduction to Bayesian Analysis* (Chapman Hall/CRC, New York, NY, 2013)
4. M. Plummer, (2015), https://sourceforge.net/projects/mcmc-jags/files/Manuals/4.x/jags_installation_manual.pdf/download
5. G. Baio, A.P. Dawid, *Stat. Methods Med. Res.* (2011). doi:[10.1177/0962280211419832](https://doi.org/10.1177/0962280211419832)
6. Stan: A C++ Library for Probability and Sampling, Version 2.8.0 (2015), <http://mc-stan.org/>. Accessed 22 Sept 2015
7. G. Baio, *Bayesian Methods in Health Economics* (Chapman Hall/CRC Press, Boca Raton, FL, 2012)
8. D. Wilkinson, *Perform. Improv. Q.* (1999)
9. N.J. Welton, A.J. Sutton, N.J. Cooper, K.R. Abrams, A.E. Ades, *Evidence Synthesis for Decision Making in Healthcare* (John Wiley & Sons, Ltd, 2012)
10. S. Dias, N. Welton, A. Sutton, A. Ades, Technical support documents: Evidence synthesis series. Tech. rep., National Institute for Health and Care Excellence (NICE), Decision Support Unit (2013), [http://www.nicedsu.org.uk/Evidence-Synthesis-TSD-series\(2391675\).htm](http://www.nicedsu.org.uk/Evidence-Synthesis-TSD-series(2391675).htm)
11. V. Hasselblad, **18**, 37 (1998)
12. A. Gelman, J. Carlin, H. Stern, D. Dunson, A. Vehtari, D. Rubin, *Bayesian Data Analysis*, 3rd edn. (Chapman Hall/CRC, New York, NY, 2013)
13. Y.S. Su, M. Yajima, R2jags—a package for running jags from R. (2012), <http://cran.r-project.org/web/packages/R2jags/index.html>
14. W.D. McGhan, M. Smith, *Am. J. Health-Syst. Pharm.* **53**, 45 (1996)
15. S. Flack, M. Taylor, P. Trueman, Cost-effectiveness of interventions for smoking cessation. Tech. rep. York Health Economics Consortium (2007)
16. D.H. Taylor, Jr, V. Hasselblad, S.J. Henley, M.J. Thun, F.A. Sloan, **92**(6) (2002)
17. ASH: Action on Smoking and Health. ASH fact sheet on smoking statistics. (2013), http://ash.org.uk/files/documents/ASH_106.pdf

Bayesian Cost-Effectiveness Analysis with the R
package BCEA

Baio, G.; Berardi, A.; Heath, A.

2017, XVI, 168 p. 55 illus., 26 illus. in color., Softcover

ISBN: 978-3-319-55716-8