Example 5.1: ***10TT.*** The 10 Top Tips trial (10TT, **cit?**) blah blah blah... Table 5.2 shows the first few rows of the dataset.

Table 5.2: The first few rows of the 10TT dataset

		Demographics[a,b]				HRQL data[c]						
ID	Trt	Sex	Age	BMI	GP	u_0	u_3	u_6	u_{12}	u_{18}	u_{24}	c^{d}
2	1	F	66	1	21	0.088	0.85	0.69	0.088	0.69	0.59	4230
3	1	M	57	1	5	0.796	0.8	0.8	0.62	0.8	1	1585
4	1	M	49	2	5	0.725	0.73	0.8	0.848	0.8	0.29	331
12	1	M	64	2	14	0.85	0.85	1	1	0.85	0.73	1034
13	1	M	66	1	9	0.848	0.85	0.85	1	0.85	0.73	1321
21	1	M	64	1	3	0.848	1	1	1	0.85	1	521

[a] BMI=Categorised body mass index
[b] GP=Number of GP visits
[c] QoL measurements at baseline, 3, 6, 12, 18 and 24 months
[d] Total costs

Computing the QALYs. We can use the following R code to manipulate the original data and compute the QALYs, given the HRQL measurements at the various time points in the follow-up of the study, essentially applying Equation 5.1.

```
# Defines a simple function to do the discounting
disc=function(x, year, disc_rate = 0.035) {
  x / ((1 + disc_rate)^(year - 1))
}

# Temporary computation of QoL + QALY with discounting by using long format
df_qol=ttt |>
  ## select columns of interest
  select(id, arm, contains("qol")) |>
  ## convert dataframe from wide to long format
  pivot_longer(
    contains("qol"), names_to = "month", names_prefix = "qol_",
    names_transform = list(month = as.integer), values_to = "qol"
  ) |>
  # convert month to follow-up year; set baseline to be in year 1
  mutate(year = pmax(1, ceiling(month/12))) |>
  ## apply discounting
  mutate(qol_disc = disc(qol, year)) |>
  # group by individual
  group_by(id) |> mutate(
  # time duration between measurements
    delta=month-dplyr::lag(month,default = 0),
  # sum of utilities between two consecutive time points
    du=qol_disc+dplyr::lag(qol_disc,default = 0),
  # area under the curve (AUC)
    auc=du*delta/2
```

```
  ) |>
  # compute the QALYs (in years so divide by 12!), as sum of the AUC
  # for all time points
  summarise(qaly=sum(auc)/12) |> ungroup()

# Merges the overall QALYs to the main dataset
ttt=ttt |> left_join(df_qol,by=c("id"="id")) |>
  mutate(
    Trt=arm+1,arm=factor(arm),
    arm=case_when(arm=="0"~"Control",TRUE~"Treatment")
  )
```

First, we create a function `disc`, which applies a default 3.5% discount rate to the HRQL measurements at the various time points (see Section 4.1.1). Then we transform the original dataset from "wide" to "long" format — one in which each row contains a time measurement and thus individuals are replicated as many times as there are measurements. Then we group by individual and first create the variables `du` and `delta` (same as in Equation 5.1), to represent the sum of two consecutive HRQL measurements and the duration of time across them. With that, we compute the "area under the curve" value (see Figure 5.1) and then the full QALYs for each individual in the dataset.

The final command merges the computed QALYs back into the original dataset, adding an extra column with the resulting values.

Figure 5.2 show histograms for the distributions of QALYs and total costs grouped by treatment arm. As is possible to see, the QALYs are highly skewed to the left; this is sensible, because we have a time horizon of 2 years and given the underlying disease is not life-threatening, it is reasonable to imagine that most individuals would be in a state of good health, with quite a few even in "perfect health", resulting in very high QALYs over the two year range.

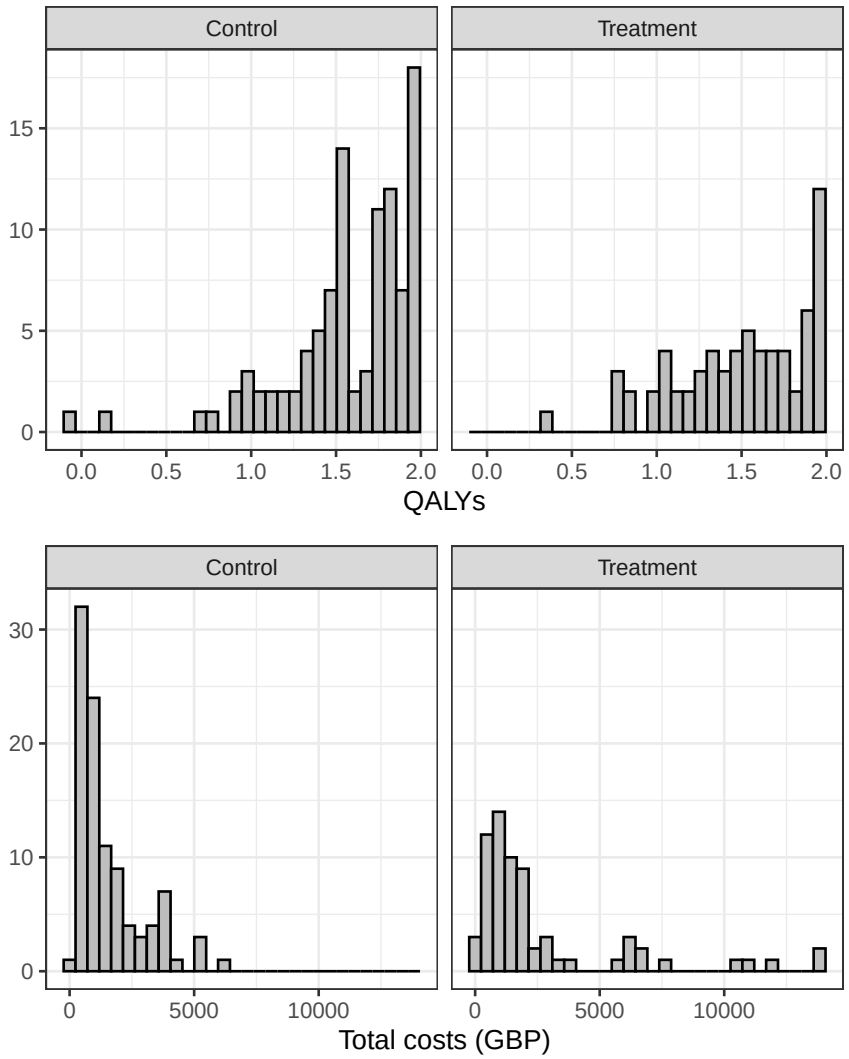


Figure 5.2: Empirical distribution for QALYs and total costs in the two treatment arms of the 10TT

5.2 Modelling ILD in HTA

5.2.1 Normal/Normal independent model

The "standard" and most basic statistical modelling (often implicitly) assume normality and linearity and models *independently* individual benefits/HRQLs and total costs by controlling for the *centered* baseline values, e.g. $u_i^* = (u_i - \bar{u})$ and $c_i^* = (c_i - \bar{c})$

$$e_i = \alpha_{e0} + \alpha_{e1} u_{0i}^* + \alpha_{e2} \text{Trt}_i + \varepsilon_{ei} [+ \ldots], \qquad \varepsilon_{ei} \sim \text{Normal}(0, \sigma_e) \tag{5.2}$$

$$c_i = \alpha_{c0} + \alpha_{c1} c^*_{0i} + \alpha_{c2} \text{Trt}_i + \varepsilon_{ci} \, [+ \ldots], \qquad \varepsilon_{ci} \sim \text{Normal}(0, \sigma_c), \tag{5.3}$$

where the notation $[+\ldots]$ indicates that we may want to include additional covariates in the model, for instance some demographic measurements on the individuals.

Statistics vs econometrics

Equation 5.2 and Equation 5.3 describe two linear regression models, in a form that is perhaps more familiar in Econometrics than it is in Statistics, where the observable variable is expressed in terms of a "fixed" component (the linear predictor) and some random noise (the error terms ε_{ei} and ε_{ci}).
If you prefer to speak Statistics than Econometrics[1], a more common description of this model is probably to consider

$$\begin{aligned} e_i &\sim \text{Normal}(\mu_{ei}, \varepsilon_{ei}) \\ \mu_{ei} &= \alpha_{e0} + \alpha_{e1} u^*_{0i} + \alpha_{e2} \text{Trt}_i \, [+ \ldots] \end{aligned}$$

(of course, this extends obviously to c_i).
The two models are identical as they encode the same assumptions, in slightly different semantic ways.

Even with underlying randomised data, it is usually a good idea to control for the baseline values of utility and costs. The reason for this is that, most often, randomisation is performed with respect to the main clinical outcome y, rather than to the economic ones (e, c) and thus it is still possible that the baseline measurements for (e, c) may be unbalanced among the different treatment or intervention arms. Using a centered version of these covariates typically simplifies the interpretation of the results (and we come back to this in Section 5.2.2).

The models in Equation 5.2 and Equation 5.3 can produce directly an estimate of the population average effectiveness and cost differentials — under these specifications, these are directly $\Delta_e = \alpha_{e2}$ and $\Delta_c = \alpha_{c2}$. In a fully frequentist/maximum likelihood analysis, we can use basic functions such as `lm`, in `R` to obtain the estimates for the model parameters and then quantify the impact of uncertainty in model parameters on the decision making process using resampling methods, such as the bootstrap.

Example 5.2: *10TT (continued): Normal/Normal independent model.* We start by modelling the 10TT data using the model in Equation 5.2 and Equation 5.3. For each of the $i = 1, \ldots, N$ observations we model $e_i \sim \text{Normal}(\phi_{ei}, \sigma_{et})$, where $t = 1, 2$ are the two intervention arm ($t = 1$ indicates the standard of care, while $t = 2$ is the active intervention). The linear predictor for the location is defined as

$$\phi_{ei} = \alpha_0 + \alpha_1 (\text{Trt}_i - 1) + \alpha_2 (u_{0i} - \bar{u}_0)$$

— note that because for each individual we code in the data the treatment indicator as $\text{Trt}_i = 1, 2$, we use the rescaled version $(\text{Trt}_i - 1)$ in the linear predictor, so that when $\text{Trt}_i = 1$ (the control intervention), then the term disappears from the linear regression model and thus α_1 indicates the effect of the active intervention. In line with Equation 5.2, we control for the scaled (centered) version of the baseline HRQL, $u^*_{0i} = (u_{0i} - \bar{u}_0)$, whose effect is measured by the coefficient α_2. Consequently, we can define $\mu_{et} = \alpha_0 + \alpha_1 (t - 1)$, for $t = 1, 2$, so that, in the control arm ($t = 1$), the effect of the treatment disappears.

[1] Unsurprisingly, the author does prefer to speak Statistics.

As for the costs, we model $c_i \sim \text{Normal}(\phi_{ci}, \sigma_{ct})$, in a similar way to what we have done for the effects

$$\phi_{ci} = \beta_0 + \beta_2(\text{Trt}_i - 1).$$

Here, β_0 is the value of the location parameter ϕ_{ci} when the treatment arm is $\text{Trt}_i = 1$ (i.e. the standard of care). Thus, we estimate of the population average costs as $\mu_{ct} = \beta_0 + \beta_2(t-1)$, for $t = 1, 2$. Because we only have data on the total costs over the course of the study, unlike for the cost, we do not control for the background level. Note that we use the notation β_2 to describe the treatment effect on the costs, for consistence with the model in Example 5.3.

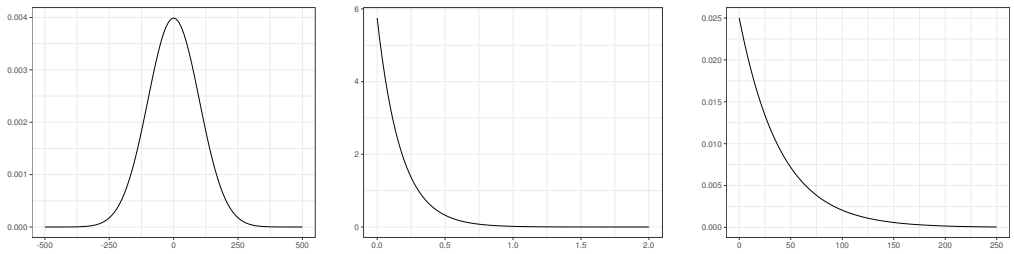
Finally, we define the prior distributions. For the regression coefficients we use minimally informative Normal priors

$$\alpha_0, \alpha_1, \alpha_2, \beta_0, \beta_2 \overset{iid}{\sim} \text{Normal}(\text{mean} = 0, \text{sd} = 100)$$

(this means of course a precision of $100^{-2} = 0.0001$). Note that we could probably do much better than this — given the scale of the observed data, it is unlikely that α_0 and α_1 have such a large range. However, given we have a reasonably large sample size, this very vague prior will probably not impact on the convergence of the model (see the results below).

For the standard deviations for the observed costs and effects, we consider a PC prior (see Section 2.2.3 and Example 2.5 in particular). Given knowledge of the scale of the two outcomes, we assume that $\Pr(\sigma_{et} > 0.8) \approx 0.01$, leading to an Exponential distribution with rate $-\frac{\log(0.01)}{0.8} \approx 5.75$; and $= \Pr(\sigma_{ct} > 100) \approx 0.1$, which implies an Exponential distribution with rate $-\frac{\log(0.1)}{100} \approx 0.025$.

Figure 5.3 shows these prior distributions. As is possible to see, the regression coefficients are widely spread and the data standard deviations have decreasingly small mass for higher values. In all cases, we do not rule out *a priori* the possibility that the evidence would change these distributions to different shapes.



(a) $\alpha_0, \alpha_1, \alpha_2, \beta_0, \beta_2$ (b) σ_e (c) σ_c

Figure 5.3: The prior distributions for the parameters of the 10TT model

The full modelling assumptions under a Bayesian framework can be written into suitable `JAGS` code. We do so by creating an `R` function (as in Note 4 in Example 3.1).

```
# Normal/Normal independent model - JAGS code
nn_indep=function(){
  for (i in 1:N) {
    # Model for the effects
    e[i] ~ dnorm(phi.e[i], tau.e[Trt[i]])
    phi.e[i] <- alpha0 + alpha1*(Trt[i]-1) + alpha2*u0star[i]
```

```
    # Model for the costs
    c[i] ~ dnorm(phi.c[i], tau.c[Trt[i]])
    phi.c[i] <- beta0 + beta2*(Trt[i]-1)
  }
  # Rescales the main economic parameters
  for (t in 1:2) {
    mu.e[t] <- alpha0 + alpha1*(t-1)
    mu.c[t] <- beta0 + beta2*(t-1)
  }
  # Minimally informative priors on the regression coefficients
  alpha0 ~ dnorm(0,0.0001)
  alpha1 ~ dnorm(0,0.0001)
  alpha2 ~ dnorm(0,0.0001)
  beta0 ~ dnorm(0,0.0001)
  beta2 ~ dnorm(0,0.0001)
  for (t in 1:2) {
  # PC-prior on the sd with Pr(sigma_e>0.8) \approx 0.01
    sigma.e[t] ~ dexp(5.75)
    tau.e[t] <- pow(sigma.e[t],-2)
  # PC-prior on the sd with Pr(sigma_c>100) \approx 0.1
    sigma.c[t] ~ dexp(0.025)
    tau.c[t] <- pow(sigma.c[t],-2)
  }
}
```

Note that BUGS/JAGS index the Normal distribution in terms of the precision, which we define as $\tau_{et} = \sigma_{et}^{-2}$, to encode the assumption that each treatment arm has a specific measure of variance. Note also that we use the observed treatment indicator t to construct the index to assign to the precision. This is consistent with the prior distributions definition — the block

```
for (t in 1:2) {
   sigma.e[t] ~ dexp(5.75)
   tau.e[t] <- pow(sigma.e[t],-2)
   ...
```

specifies a separate prior for each treatment arm on the standard deviation scale and then maps it on the precision scale, which is required for BUGS/JAGS to run the model.

We now run the model using JAGS and R2jags, with the following R commands. Once the model has completed the run, we can visulise the summary statistics using the print method for the resulting JAGS object.

```
# Defines the data list
data=list(e=e,c=c,Trt=Trt,u0star=u0star,N=N)

# Initialises the object 'model' as a list to store all the results
model=list()

# Runs JAGS in the background
library(R2jags)
# Stores the BUGS output as an element named 'nn_indep' in the object 'model'
```

```
model$nn_indep=jags(
  data=data,
  parameters.to.save=c(
    "mu.e","mu.c","alpha0","alpha1","alpha2","beta0","beta2","sigma.e","sigma.c"
  ),
  inits=NULL,n.chains=2,n.iter=5000,n.burnin=3000,n.thin=1,DIC=TRUE,
  # This specifies the model code as the function 'nn_indep'
  model.file=nn_indep
)

# Shows the summary statistics from the posterior distributions.
print(model$nn_indep,digits=3,interval=c(0.025,0.5,0.975))
```

```
Inference for Bugs model at "/tmp/Rtmpa7bxAs/model42da61e0475de.txt",
 2 chains, each with 5000 iterations (first 3000 discarded)
 n.sims = 4000 iterations saved. Running time = 1.241 secs
           mu.vect sd.vect     2.5%      50%    97.5%  Rhat n.eff
alpha0       1.551   0.021    1.510    1.551    1.594 1.001  4000
alpha1      -0.006   0.039   -0.083   -0.006    0.071 1.001  4000
alpha2       1.279   0.072    1.142    1.279    1.420 1.001  4000
beta0      638.594  88.794  462.841  638.371  811.984 1.001  4000
beta2      175.685  94.570   -8.773  176.336  363.125 1.001  4000
mu.c[1]    638.594  88.794  462.841  638.371  811.984 1.001  4000
mu.c[2]    814.279 125.809  562.616  813.342 1062.826 1.001  4000
mu.e[1]      1.551   0.021    1.510    1.551    1.594 1.001  4000
mu.e[2]      1.545   0.033    1.482    1.546    1.611 1.001  4000
sigma.c[1] 1409.250  88.264 1244.271 1408.408 1583.078 1.001  2900
sigma.c[2] 2656.446 142.146 2397.957 2653.220 2940.305 1.020   110
sigma.e[1]    0.208   0.015    0.180    0.207    0.240 1.001  4000
sigma.e[2]    0.267   0.023    0.227    0.265    0.317 1.001  4000
deviance   3063.636  12.067 3040.748 3063.029 3087.780 1.004   540

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: pV = var(deviance)/2)
pV = 72.7 and DIC = 3136.3
DIC is an estimate of expected predictive error (lower deviance is better).
```

The weird path for the model file (in this case `/tmp/Rtmpa7bxAs/model42da61e0475de.txt` depends on the fact that we are coding up the model as a function. `JAGS` will write data and model files onto a temporary folder).

As is possible to see from the summary statistics, convergence seems to be reached for all the model parameters and, according to the ESS (see Section 2.3.2.2) is reasonably large for almost all the parameters, indicating that there are no crucial issues with autocorrelation. We can use `bmhe` and other packages to produce other diagnostics visualisations and summaries, as in Section 2.3.2.

The intervention seems to produce slightly lower QALYs than the standard of care (on average, 1.545 against 1.551). These differences do not seem substantial, though, with much of the underlying distributions crossing each other. The difference in population average

costs seem to be more marked, with average values of £814.279 against £638.594, although, again, the whole distributions do not seem too distant. As mentioned above, these relative differences are equivalent to the observed posterior values for the regression parameters α_1 and β_1.

The posterior summaries for the parameters σ_{et} and σ_{ct} indicate that the evidence from the data perhaps is stronger than implied in the prior, with the former shrunk towards smaller values and the latter pushed to higher ones. Then again, the PC priors are able to not force too much information and can be modified if a strong signal is observed in the data.

The first important issue with the basic structure described above is that it fails to account for the potential correlation between costs and clinical benefits. In general, this may materialise as a positive correlation, since effective treatments are innovative and result from intensive and lengthy research and are thus associated with higher unit costs. However, it may also arise as negative correlation, because more effective treatments may end up reducing total care pathway costs e.g. by reducing hospitalisations, side effects, etc.

Either way, modelling costs and benefits separately and (more or less implicitly) assuming independent error terms in the regression models in Equation 5.2 and Equation 5.3 simply fails to allow for the possibility of correlation, potentially leading to biased estimates.

In a frequentist/maximum likelihood setting, one possible way around this limitation is to use so-called "seemingly unrelated regression" (SUR) models, originally proposed by Zellner (1962). SUR models essentially imply a correlation structure between the error terms $(\varepsilon_{ei}, \varepsilon_{ci})$ — this is effectively the same model as in Equation 5.2 and Equation 5.3, but with the added assumption that

$$\begin{pmatrix} \varepsilon_{ei} \\ \varepsilon_{ci} \end{pmatrix} \sim \text{Normal}\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \sigma_e^2 & \rho\sigma_e\sigma_c \\ & \sigma_c^2 \end{pmatrix} \right).$$

Early advice on statistical modelling for HTA did consider this — see for instance Willan and Briggs (2006).

A second potential issue is that joint or marginal normality is generally not a realistic assumption. This is because costs are usually skewed and certainly bounded in $[0; \infty)$, while benefits may be bounded in an interval $[a; b]$, for some problem-specific constants, e.g. when representing QALYs — see Figure 5.2.

One obvious solution to this would be to use transformation, for instance to model $\log c_i$; while this is reasonably easy to do, care is needed when back transforming to the natural scale — recall that the objectives of the economic evaluation are the population *average* (mean) cost and benefits, because it is them we need to feed to the decision analysis (see Chapter 4). We return to this problem in Example 5.4 and in Section 5.2.3.

5.2.2 Marginal/conditional factorisation model

To overcome the issues highlighted above, we can take a set of countermeasures, which allow us to build models capable of producing estimates of the population average cost and benefit accounting for potential correlation as well as for the underlying nature of the observed data. This is of course an important objective — but it does mean using a much more structured modelling framework, as we explain below.

The main objective is to consider as target for our inference the *joint* probability distribution $p(e, c \mid \boldsymbol{\theta})$, for some vector of model parameters $\boldsymbol{\theta}$. A useful strategy that allows us to move away from the normality assumption without too many complications makes use of the fundamental properties of probability, under which we can always factorise a joint distribution into the product of a (univariate) marginal and a (univariate) conditional:

$$p(e, c) = p(e)p(c \mid e) = p(c)p(e \mid c) \tag{5.4}$$

— note that we temporarily drop the dependence to the model parameters, for simplicity. We label this modelling strategy as "marginal/conditional factorisation" (MCF) model.

From a purely probabilistic point of view, the two different factorisations are equivalent and repesent fully the target joint distribution. Pragmatically, we may have some underlying causal construct, whereby, for instance, it is more appropriate to think of a DGP in which the effects have an impact on the costs (and thus we may favour the first factorisation in Equation 5.4).

Figure 5.4 shows a graphical representation of this structured modelling framework — indeed, we assume that the joint is factorised as $p(e, c \mid \boldsymbol{\theta}) = p(e \mid \boldsymbol{\theta}_e)p(c \mid e, \boldsymbol{\theta}_c)$, for some specific subset of parameters $\boldsymbol{\theta} = (\boldsymbol{\theta}_e, \boldsymbol{\theta}_c)$. The blue dashed box indicates the marginal model for the benefits $e_i \sim p(e \mid \phi_{ei}, \boldsymbol{\xi}_e)$. Here, $\boldsymbol{\theta}_e = (\phi_{ei}, \boldsymbol{\xi}_e)$; ϕ_{ei} is a *location* parameter indicating a measure of central tendency for the underlying distribution, while $\boldsymbol{\xi}_e$ are a set of *ancillary* parameters, which may indicate a measure of shape, or variance — these may or may not be present, depending on the form of the distribution $p(\cdot)$ that we choose.

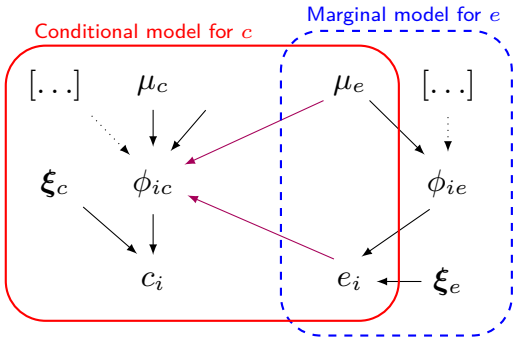


Figure 5.4: Some caption

Generally speaking, we can model the location parameter using some generalised linear regression

$$g_e(\phi_{ei}) = \alpha_0 \, [+ \ldots]. \tag{5.5}$$

The red box in Figure 5.4 shows the conditional model for the cost, given the effects. We describe $c_i \sim p_c(c \mid e, \phi_{ci}, \boldsymbol{\xi}_c)$, where, again, ϕ_{ci} is a location and $\boldsymbol{\xi}_c$ is a vector of ancillary parameters. We include the dependence between costs and effects through the location parameter, which is again modelled using a generalised linear regression

$$g_c(\phi_{ci}) = \beta_0 + \beta_1 \, (e_i - \mu_e) \, [+ \ldots], \tag{5.6}$$

where this time we definitely have one (centered) covariate, i.e. the observed individual benefits, scaled by the estimated population average (which we get from the previous module).

The coefficient β_1 quantifies the level of correlation between e and c — in the simplest case of a Normal/Normal model (which we describe in more details in Example 5.3), we can prove analytically the relationship

$$\beta_1 = \rho\frac{\sigma_c}{\sigma_e} \qquad \Rightarrow \qquad \rho = \beta_1\frac{\sigma_e}{\sigma_c}, \tag{5.7}$$

where $\rho \in (-1, 1)$ is the correlation coefficient between costs and benefits (see Example 5.3).

It is very much possible that the evidence in the data suggests that, in fact, there is no substantial correlation among the two outcomes and thus the estimate for its value would be close to 0 — in a frequentist setup, this would be "non-significantly different from 0". On the other hand, the model does allow for the possibility that the two outcomes are indeed correlated and if the evidence supported this assumption, then β_1 would be estimated with a value away from 0.

In addition, the modelling structure encoded in Figure 5.4 do make the simplifying assumption that the correlation between costs and benefits only happens through the location parameter. However, this is by no means a requirement and it would be possible to add even further structure and build another generalised linear regression component in which $\boldsymbol{\xi}_c$ does depend probabilistically on e_i.

If no further covariates are included in the two generalised linear regressions of Equation 5.5 and Equation 5.6, the intercept α_0 represents the population average benefits, on the scale defined by $g_e(\cdot)$, while the intercept β_0 is the population average costs on the scale defined by $g_c(\cdot)$ and computed for an individual whose benefits are set to the observed average. This is because, if that is the case, then $(e_i - \mu_e) = 0$. Consequently, we can compute $\mu_e = g_e^{-1}(\alpha_0)$ and $\mu_c = g_c^{-1}(\beta_0)$.

If we do consider additional covariates in Equation 5.5 and/or Equation 5.6, we can still use the rescaled intercepts to estimate the population average benefits and costs. However, in this case, the models in Equation 5.5 and Equation 5.6 target directly two *conditional* estimates, given the "profile" of covariates and so a little care is needed.

For example, consider the case

$$g_e(\phi_{ei}) = \alpha_0 + \alpha_1(u_{0i} - \bar{u}_0) + \alpha_2 \text{Sex}_i,$$

where $(u_{0i} - \bar{u}_0)$ is the centered baseline HRQL continuous measurement and Sex_i is a discrete variable taking values $(0, 1)$ for males and females, respectively. Under this specification, $g_e^{-1}(\alpha_0)$ = population average benefits for a male with average baseline HRQL measurement, while $g_e^{-1}(\alpha_0 + \alpha_2)$ = population average benefits for a female with average baseline HRQL measurement. We can approximate the *marginal* population average benefit as

$$\mu_e \approx p_M g_e^{-1}(\alpha_0) + p_F g_e^{-1}(\alpha_0 + \alpha_2),$$

where p_M and p_F are the frequencies with which males and females are observed in the population of interest. Note that marginalisation of continuous covariates is automatic with centering (because it makes sense to determine the "average" profile, with respect to these covariates). Conversely, for discrete covariates we need to effectively compute a weighted average over their levels — we return to the distinction between marginal and conditional estimates in Chapter 10.

⚠ To be or not to be... (Bayesian)?

While there is nothing inherently Bayesian about the modelling framework described in Figure 5.4, there are several advantages to using a Bayesian approach.

1. As the model becomes more and more realistic and its structure more and more complex, to account for skeweness and correlation, the computational advantages of using maximum likelihood estimations become increasingly smaller. This is because writing and optimising the likelihood function becomes more complex analytically and even numerically and might require the use of simulation althorithms. A Bayesian approach, conversely, can scale up with minimal changes — whether we consider the simple Normal-Normal model or more convoluted distributional assumptions, MCMC methods effectively require almost no modification. The computation may be more expensive, but the marginal computational cost is in fact diminishing.
2. As mentioned above, it is very possible that realistic models be based on highly non-linear transformations; from a Bayesian perspective, this does not pose a substantial problem, because once we obtain simulations from the posterior distribution for α_0 and β_0, it is possible to simply rescale them to obtain samples from the posterior distribution of μ_e and μ_c. This in turn allows us to fully characterise and propagate the uncertainty in the fundamental economic parameters to the rest of the decision analysis with essentially no extra computational cost. A frequentist/maximum likelihood approach would resort to resampling methods, such as the bootstrap to effectively produce an *approximation* to the joint posterior distribution for all the model parameters $\boldsymbol{\theta}$ and any function thereof.
3. Prior information can help stabilise inference (especially with sparse data, as shown in Example 1.2). Most often, we do have contextual information to mitigate limited evidence from the data and stabilise the evidence — for instance, we may be confident in thinking that odds ratio for drug-related treatment effects are rarely very large (say, larger than 1.5 or 2); similarly, when looking at time-to-event data, we know for instance that cancer patients are unlikely to outlive the general population and thus we may "anchor" estimates for sick individuals using population life tables. Using a Bayesian approach allows us to use this contextual information in a principled way.

Example 5.3: *10TT (continued): Normal/Normal MCF model.* The simplest version of the modelling framework of Figure 5.4, which is equivalent to a standard implementation of the SUR model, is:

$$\begin{aligned} e_i &\sim \text{Normal}(\phi_{ei}, \tau_{et}) & \phi_{ei} &= \alpha_0 + \alpha_1(\text{Trt}_i - 1) + \alpha_2 u^*_{0i} & \mu_{et} &= \alpha_0 + \alpha_1(t-1) \\ c_i \mid e_i &\sim \text{Normal}(\phi_{ci}, \tau_{ct}) & \phi_{ci} &= \beta_0 + \beta_1(e_i - \mu_e) + \beta_2(\text{Trt}_i - 1) & \mu_{ct} &= \beta_0 + \beta_2(t-1), \end{aligned}$$

where ϕ_{ei} and τ_e are the benefits *marginal* mean and precision, respectively, while ϕ_{ci} and τ_c are the costs *conditional* mean and precision, respectively. In this case both $g_e(\cdot)$ and $g_c(\cdot)$ are the identity function and so the two intercepts represent directly the population average parameters.

This formulation is indeed fairly similar to that of Equation 5.2 and Equation 5.3, with the notable improvement that we no longer assume independence across the outcomes, because of the inclusion of the term $\beta_2(\text{Trt}_i - 1)$ in the linear predictor for the location parameter of the conditional cost distribution.

While perhaps less intuitive, *at face value*, than a bivariate distribution, the modelling framework of Figure 5.4 has the advantage that the two components in which we factorise the joint distribution have lower dimension — we do not need to model a bivariate distribution, but two univariate ones.

What's so special about the Normal distribution?...

One of the nice features of the Normal distribution is that, given a joint, multivariate Normal for a K-dimensional vector $\boldsymbol{y} = (y_1, \ldots, y_K) \sim \text{Normal}_K(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, then we can prove that all the resulting marginal and conditional distributions remain Normal. Specifically, in the bivariate case of interest in HTA

$$\begin{pmatrix} e \\ c \end{pmatrix} \sim \text{Normal}_2 \left(\begin{pmatrix} \mu_e \\ \mu_c \end{pmatrix}, \begin{pmatrix} \sigma_e^2 & \rho\sigma_e\sigma_c \\ & \sigma_c^2 \end{pmatrix} \right)$$

we can prove that

$$e \sim \text{Normal}(\mu_e, \sigma_e) \quad \text{and} \quad c \mid e \sim \text{Normal}\left(\eta_c, \lambda_c\right),$$

where

$$\eta_c = \mu_c + \rho\frac{\sigma_c}{\sigma_e}(e - \mu_e)$$

is the *conditional mean* and

$$\lambda_c = \sqrt{(1-\rho^2)\sigma_c^2}, \tag{5.8}$$

is the *conditional standard deviation* for the costs, both defined as functions of the marginal means μ_e, μ_c, the marginal standard deviations σ_e, σ_c and the correlation coefficient ρ. Of course we can derive the other factorisation with a marginal for c and a conditional $e \mid c$, accordingly.
This property, specific to the Normal distribution, implies that the Normal/Normal MCF model is indeed equivalent to the SUR model.

The modelling assumptions detailed above can be encoded in the JAGS code below.

```
# Normal/Normal MCF - JAGS code
nn_mcf=function(){
  for (i in 1:N) {
    # Marginal model for the effects
    e[i] ~ dnorm(phi.e[i], tau.e[Trt[i]])
    phi.e[i] <- alpha0 + alpha1*(Trt[i]-1) + alpha2*u0star[i]
    # *Conditional* model for the costs
    c[i] ~ dnorm(phi.c[i], tau.c[Trt[i]])
    phi.c[i] <- beta0 + beta1*(e[i]-mu.e[Trt[i]]) + beta2*(Trt[i]-1)
  }
  # Rescales the main economic parameters
  for (t in 1:2) {
    mu.e[t] <- alpha0 + alpha1*(t-1)
    mu.c[t] <- beta0 + beta2*(t-1)
```

```
  }
  # Minimally informative priors on the regression coefficients
  alpha0 ~ dnorm(0,0.0001)
  alpha1 ~ dnorm(0,0.0001)
  alpha2 ~ dnorm(0,0.0001)
  beta0 ~ dnorm(0,0.0001)
  beta1 ~ dnorm(0,0.0001)
  beta2 ~ dnorm(0,0.0001)
  for (t in 1:2) {
  # PC-prior on the *marginal* sd with Pr(sigma_e>0.8) \approx 0.01
    sigma.e[t] ~ dexp(5.75)
    tau.e[t] <- pow(sigma.e[t],-2)
  # PC-prior on the *conditional* sd with Pr(lambda_c>100) \approx 0.1
    lambda.c[t] ~ dexp(0.025)
    tau.c[t] <- pow(lambda.c[t],-2)
  # Retrieves the correlation coefficients
    rho[t] <- beta1*sigma.e[t]/sigma.c[t]
  # And the *marginal* standard deviation for the cost
    sigma.c[t] <- sqrt(pow(lambda.c[t],2) + pow(sigma.e[t],2)*pow(beta1,2))
  }
}
```

Because of the underlying assumption of joint Normality, we can write down analytically and monitor the posterior distributions for the correlation coefficient `rho[t]` and the marginal cost standard deviation `sigma.c[t]`, obtained using Equation 5.7 and combining it with Equation 5.8.

In reality, defining and monitoring `rho[t]` and `sigma.c[t]` is not essential — we can do it analytically in the bivariate Normal case and thus we show how to handle this in the code above. But, realistically, for the purposes of the HTA analysis, we do not care massively about them and could dispense with this extra computation.

The model is run using the following code.

```
# Runs JAGS in the background and stores the output in the element 'nn_mcf'
model$nn_mcf=jags(
  data=data,
  parameters.to.save=c(
    "mu.e","mu.c","alpha0","alpha1","alpha2","beta0","beta1","beta2",
    "sigma.e","sigma.c","lambda.c","rho"
  ),
  inits=NULL,n.chains=2,n.iter=5000,n.burnin=3000,n.thin=1,DIC=TRUE,
  # This specifies the model code as the function 'nn_mcf'
  model.file=nn_mcf
)

# Shows the summary statistics from the posterior distributions.
print(model$nn_mcf,digits=3,interval=c(0.025,0.5,0.975))
```

```
Inference for Bugs model at "/tmp/RtmpVHriMa/model415f71bb6e0e3.txt",
 2 chains, each with 5000 iterations (first 3000 discarded)
 n.sims = 4000 iterations saved. Running time = 1.691 secs
```

```
             mu.vect sd.vect     2.5%      50%    97.5%  Rhat n.eff
alpha0         1.554   0.020    1.514    1.554    1.594 1.001  4000
alpha1        -0.006   0.039   -0.080   -0.006    0.070 1.002  1300
alpha2         1.280   0.072    1.137    1.282    1.419 1.001  4000
beta0        645.145  89.713  476.544  643.876  818.636 1.001  4000
beta1       -102.446  96.540 -292.081 -102.709   86.175 1.001  3200
beta2        173.534  98.859  -21.773  174.862  365.761 1.007   250
lambda.c[1] 1400.050  90.945 1229.855 1396.864 1578.685 1.014  4000
lambda.c[2] 2664.867 152.051 2368.575 2664.838 2949.444 1.037    47
mu.c[1]      645.145  89.713  476.544  643.876  818.636 1.001  4000
mu.c[2]      818.679 129.750  560.908  819.177 1069.656 1.004   450
mu.e[1]        1.554   0.020    1.514    1.554    1.594 1.001  4000
mu.e[2]        1.548   0.033    1.484    1.548    1.612 1.002  1400
rho[1]        -0.015   0.014   -0.044   -0.015    0.013 1.001  2800
rho[2]        -0.010   0.010   -0.030   -0.010    0.009 1.001  4000
sigma.c[1]  1400.356  90.908 1230.352 1397.162 1579.041 1.014  4000
sigma.c[2]  2665.136 152.017 2368.634 2665.013 2949.507 1.037    47
sigma.e[1]     0.207   0.015    0.181    0.206    0.238 1.001  2500
sigma.e[2]     0.267   0.024    0.227    0.265    0.319 1.001  4000
deviance    3060.926  12.575 3037.550 3060.443 3087.413 1.007   220

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: pV = var(deviance)/2)
pV = 78.7 and DIC = 3139.7
DIC is an estimate of expected predictive error (lower deviance is better).
```

Looking at the summary table produced by JAGS, we can see that convergence and autocorrelation are not a problem. All $\hat{R}$ statistics are well below the rule-of-thumb threshold of 1.1 (see Section 2.3.2.1) and the effective sample size (Section 2.3.2.2) is very close to the nominal one, for almost all the monitored nodes.

(Non-essential) Computed parameters in BUGS/JAGS

In fact, we do not even need to include the definition for the nodes `rho[t]` and `sigma.c[t]` in the JAGS model, because these are simply deterministic functions of other model parameters (`beta1`, `sigma.e[t]`, and `lambda.c[t]`). Thus, if we run the model by monitoring these key nodes, we can actually construct `rho[t]` and `sigma.c[t]` in R using simple algebra.
Note that BUGS/JAGS effectively adds a dimension to each node. So the node `beta1` is defined in the model code as a scalar. But because when processed by BUGS/JAGS we record `n.sims` simulations, then the resulting object turns into a vector with `n.sims` rows. Similarly, in the model code `lambda.c` is a vector with two elements (one for each treatment arm) and therefore when processed by BUGS/JAGS the element `model$nn_mcf$sims.list$sigma.e` is in fact a matrix with `n.sims` rows and 2 columns.
For this reason, when defining in R the new nodes `sigma.c` and `rho` we need to set them to matrices — and for convenience we can use the same dimension as `lambda.c`, as shown in the code below.

```r
# Extracts the simulations from the BUGS object into R variables
lambda.c=model$nn_mcf$BUGSoutput$sims.list$lambda.c
sigma.e=model$nn_mcf$BUGSoutput$sims.list$sigma.e
beta1=model$nn_mcf$BUGSoutput$sims.list$beta1

# Defines the new variables --- note the dimensions!
sigma.c=rho=matrix(NA,nrow=nrow(lambda.c),ncol=ncol(lambda.c))

# Uses the simulated values to compute rho[t] and sigma.c[t]
for (t in 1:2) {
  sigma.c[,t] <- sqrt(lambda.c[,t]^2 + (sigma.e[,t]^2*beta1^2))
  rho[,t] <- beta1*sigma.e[,t]/sigma.c[,t]
}

# Summarises the results
colnames(sigma.c)=c("sigma.c[1]","sigma.c[2]")
colnames(rho)=c("rho[1]","rho[2]")
cbind(sigma.c,rho) |> bmhe::stats() |> round(digits=3)
```

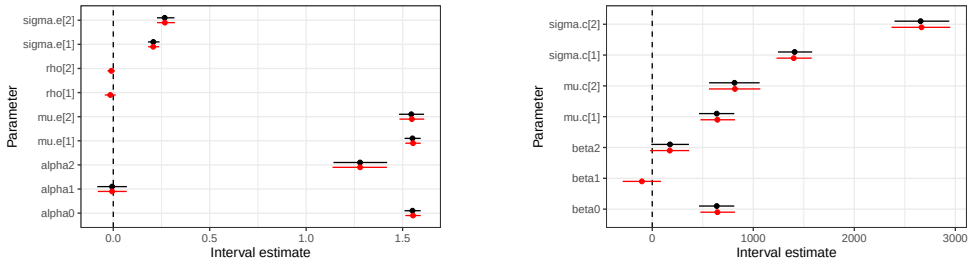
```
              mean      sd     2.5%   median    97.5%
sigma.c[1] 1400.356  90.908 1230.352 1397.162 1579.041
sigma.c[2] 2665.136 152.017 2368.634 2665.013 2949.507
rho[1]       -0.015   0.014   -0.044   -0.015    0.013
rho[2]       -0.010   0.010   -0.030   -0.010    0.009
```

As is possible to see in the summary table produced by R, the results are (of course!) identical with the ones reported directly by JAGS.
Most of the times, it does not really matter which way around we decide to make the algebraic calculation, However, it is perhaps good practice to leave all the non-essential non-MCMC computation to R, because BUGS/JAGS model scale up in computational complexity with the number of nodes defined. If the model is fairly quick to run, as in the present case, there really is not much difference; but for more structured models, it may be much more efficient to leave some of the computation back to R.

The correlation coefficients are estimated to be very small — in frequentist parlance they would be deemed to be "non-significant", because their posterior distribution crosses 0. This is consistent with the fact that the conditional and the marginal standard deviations for the costs (`lambda.c` and `sigma.c`) are effectively identical, because of the lack of correlation in the observed data.

This is a reassuring feature of the MCF model — if there is no evidence of correlation in the data, we do not lose anything in comparison to the independence model of @-10tt-nn-indep, because the resulting correlation is effectively set to 0, meaning that the two modules in Figure 5.4 become detached. If however, there is correlation in the data but we do not allow our model to pick up we may end inducing important bias in our estimates.

Figure 5.5 shows a comparison of the Normal/Normal independent and the Normal/Normal MCF models. As is possible to see, the estimates for the model parameters are effectively identical — this is consistent with the discussion above, because in the absence of substantial correlation, the two models coincide.



(a) (b)

Figure 5.5: Graphical summareis of the posterior distributions for the model parameters. The two panels collect parameters according to the range of their distributions, for better visualisation. The dots indicate the posterior means, while the lines expands to cover the 95% posterior intervals. Black dots and lines indicate the Normal/Normal independence model, while red dots and lines indicate the Normal/Normal MCF model

Using the structure described in Figure 5.4 it is possible to easily extend the Normal/Normal MCF model to virtually consider any combination of distributional assumptions for the costs and benefits. For example, we can assume a Gamma distribution to model skewed, positive costs, conditionally on a suitable measure of benefits; or a Beta distribution to model marginal benefits defined in terms of QALYs for a study where the time horizon is below 1 year — assuming there are no individuals with negative QALYs, implies that the range for the variable e would be between $(0, 1)$.

Example 5.4: *10TT (continued): Gamma/Gamma MCF model.* Figure 5.2 clearly shows that the empirical distributions for both costs and benefits are highly skewed. In addition, because the time horizon for the study is 24 months, the QALYs are naturally constrained to be below 2. For these reasons, it is a good idea to try and use non-Normal models.

One slight complication is that the QALYs show a rather large left skewness; in addition, a small number of individuals are associated with negative QALYs, indicating a very poor health state ("worse than death"). For these reasons, it is helpful to transform the original data, so that we can apply a simpler model. Following Gabrio et al. (2024), we define the transformation $e_i^* = (3 - e_i)$. This rescales the observed QALYs and turns the distribution into a right skewed, which means we can use a Gamma distribution to model the sampling variability.

With this, we can model:

$$\begin{aligned} e_i^* &\sim \text{Gamma}(\nu_e, \gamma_{ei}) & \log(\phi_{ei}) &= \alpha_0 + \alpha 1(\text{Trt}_i - 1) + \alpha_2 u_{0i}^* \\ c_i \mid e_i^* &\sim \text{Gamma}(\nu_c, \gamma_{ci}) & \log(\phi_{ci}) &= \beta_0 + \beta_1(e_i^* - \mu_e^*) + \beta_2(\text{Trt}_i - 1), \end{aligned}$$

where, because of the properties of the Gamma distribution, ϕ_{ei} indicates the marginal mean for the QALYs, while ν_e is the shape and $\gamma_{ei} = \nu_c/\phi_{ei}$ is the rate (see Section 1.3.3 and Equation 1.6, in particular). Similarly, ν_c is the shape and $\gamma_{ci} = \nu_c/\phi_{ci}$ is the rate of the conditional distribution for the costs given the benefits, with ϕ_{ci} representing the conditional mean.

The population average benefits and costs can be retrieved by back-transforming the linear predictors (using the exponential function).

$$\mu_e = \exp\left(\alpha_0 + \alpha_1(t-1)\right) \quad \text{and} \quad \mu_c^* = \exp\left(\beta_0 + \beta_2(t-1)\right).$$

In terms of prior distributions, we keep the very vague, minimally informative priors for the regression coefficients — again, we could probably do better here, considering that both ϕ_{ei} and ϕ_{ci} are defined on a log scale and thus we are implying a very large range in the priors. Sensitivity analysis may be very important to this aspect.

As for the shape parameters ν_{et} and ν_{ct}, we specify again a PC prior. Generally speaking, it is more difficult to encode genuine prior knowledge on this scale (see the discussion in Section 1.3.3) and thus we start by including a fairly vague specification, where we assume that $\Pr(\nu_{et} > 30) = \Pr(\nu_{ct} > 30) \approx 0.01$, which implies an Exponential with rate $-\frac{\log(0.01)}{30} \approx 0.15$.

The modelling assumptions are mapped onto the `JAGS` code below.

```
# Gamma/Gamma MCF - JAGS code
gg_mcf=function(){
  for (i in 1:N) {
   # Marginal model for the *rescaled* effects
   estar[i] ~ dgamma(nu.e[Trt[i]], gamma.e[Trt[i],i])
   gamma.e[Trt[i],i] <- nu.e[Trt[i]]/phi.e[i]
   log(phi.e[i]) <- alpha0 + alpha1*(Trt[i]-1) + alpha2*u0star[i]
   # Conditional model for the costs
   c[i] ~ dgamma(nu.c[Trt[i]], gamma.c[Trt[i],i])
   gamma.c[Trt[i],i] <- nu.c[Trt[i]]/phi.c[i]
   log(phi.c[i]) <- beta0 + beta1*(estar[i]-mustar.e[Trt[i]]) + beta2*(Trt[i]-1)
  }
  # Rescales the main economic parameters
  for (t in 1:2) {
   mustar.e[t] <- exp(alpha0 + alpha1*(t-1))
   mu.e[t] <- 3 - mustar.e[t]
   mu.c[t] <- exp(beta0 + beta1*(t-1))
  }
  # Minimally informative priors on the regression coefficients
  alpha0 ~ dnorm(0,0.0001)
  alpha1 ~ dnorm(0,0.0001)
  alpha2 ~ dnorm(0,0.0001)
  beta0 ~ dnorm(0,0.0001)
  beta1 ~ dnorm(0,0.0001)
  beta2 ~ dnorm(0,0.0001)
  # PC prior on the shape parameters
  # assume that Pr(nu.e>30) = Pr(nu.c>30) \approx 0.01
  for (t in 1:2) {
    nu.e[t] ~ dexp(0.15)
    nu.c[t] ~ dexp(0.15)
  }
}
```

Notice that in the code above we first define the variable `mustar.e` to indicate the population average QALYs *on the rescaled version* and then back-transform to the original scale by simply using the command `mu.e[t] <- 3 - mustar.e[t]`. This is possible because the transformation applied to the original data is linear and thus it holds for the expectations (or any other function, for that matter)

$$e^* = 3 - e \quad \Rightarrow \quad \mathrm{E}[e^*] = \mathrm{E}[3 - e] = 3 - \mathrm{E}[e] \quad \Rightarrow \quad \mathrm{E}[e] = 3 - \mathrm{E}[e^*].$$

We return to this point in Section 5.2.3 and then later in Chapter 10.

In order to run the model, we need to slightly modify the `data` list, to include the new variable `estar` (the rescaled QALYs).

```
# Defines the rescaled QALYs in the data list and remove the old version
data$estar=3-data$e

# Runs JAGS in the background and stores the output in the element 'gg_mcf'
model$gg_mcf=jags(
  data=data,
  parameters.to.save=c(
    "mu.e","mustar.e","mu.c","alpha0","alpha1","alpha2","beta0",
    "beta1","beta2","nu.e","nu.c"
  ),
  inits=NULL,n.chains=2,n.iter=5000, n.burnin=3000,n.thin=1,DIC=TRUE,
  # This specifies the model code as the function 'model.code'
  model.file=gg_mcf
)

# Shows the summary statistics from the posterior distributions.
print(model$gg_mcf,digits=3,interval=c(0.025,0.5,0.975))
```

```
Inference for Bugs model at "/tmp/RtmpVHriMa/model415f7270df293.txt",
 2 chains, each with 5000 iterations (first 3000 discarded)
 n.sims = 4000 iterations saved. Running time = 21.23 secs
             mu.vect sd.vect     2.5%      50%    97.5%  Rhat n.eff
alpha0         0.348   0.015    0.319    0.348    0.376 1.003   620
alpha1         0.011   0.028   -0.042    0.010    0.068 1.005   660
alpha2        -0.823   0.052   -0.925   -0.822   -0.724 1.001  4000
beta0          7.314   0.077    7.165    7.314    7.465 1.003   710
beta1          0.935   0.193    0.564    0.935    1.323 1.001  2500
beta2          0.428   0.147    0.146    0.428    0.720 1.002  1900
mu.c[1]     1506.222 115.834 1292.816 1501.580 1745.208 1.003   710
mu.c[2]     3913.959 847.158 2568.088 3811.682 5852.601 1.002  1300
mu.e[1]        1.584   0.021    1.543    1.583    1.625 1.003   600
mu.e[2]        1.568   0.033    1.501    1.569    1.630 1.001  2700
mustar.e[1]    1.416   0.021    1.375    1.417    1.457 1.003   610
mustar.e[2]    1.432   0.033    1.370    1.431    1.499 1.001  2400
nu.c[1]        1.830   0.240    1.404    1.820    2.337 1.001  4000
nu.c[2]        1.151   0.178    0.839    1.139    1.548 1.001  2200
nu.e[1]       47.063   6.625   35.070   46.843   60.582 1.004   500
nu.e[2]       29.583   5.113   20.366   29.329   40.314 1.002  4000
deviance    2787.409   5.145 2779.668 2786.726 2799.410 1.001  4000

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: pV = var(deviance)/2)
pV = 13.2 and DIC = 2800.6
DIC is an estimate of expected predictive error (lower deviance is better).
```

Again, convergence and autocorrelation seem under control from the summary statistics. In this case, however, the model accounting for skewness in the data suggest a much higher estimate for the average costs, which are almost doubled, in comparison to the two Normal/Normal models shown above. The "correct-scale" population average QALYs, i.e. the nodes `mu.e[1]` and `mu.e[2]`, show also some differences with respect to the Normal/Normal models, although they are more aligned.

The computational effort required to run the Gamma/Gamma MCF is, unsurprisingly, much larger, relatively speaking, than for the two Normal/Normal models — it takes about 21 seconds to run (not enough to go and make a coffee, but still more expensive in terms of running time than the simpler Normal models).

5.2.3 Non-linerity and g-computation

If we have available a set of simulations from the (posterior) distribution of a parameter ψ, say in the vector $\left(\psi^{(1)}, \ldots, \psi^{(S)}\right)$, then we can use these to retrieve a set of simulations from the (posterior) distribution of any function $\theta = g(\psi)$ simply applying the inverse transformation and compute a vector of simulations from the (posterior) distribution for θ as

$$\left(\theta^{(1)} = g^{-1}\left(\psi^{(1)}\right), \ldots, \theta^{(S)} = g^{-1}\left(\psi^{(S)}\right)\right).$$

This works because we propagate the full uncertainty in ϕ (through the S simulations) directly onto the distribution of θ and it is the basic principle of Monte Carlo simulation (see Section 1.3.4).

The problem is more complicated when we apply a non linear transformation to a variable for which we do not have directly simulations to characterise the underlying uncertainty. Consider for example an observed variable y, which we transform as $y^* = g(y)$, for some non-linear function $g(\cdot)$. Suppose also that $\mathrm{E}[Y^*] = \theta$. In this situation,

$$\mathrm{E}[Y] = \mathrm{E}[g^{-1}(Y^*)] \neq g^{-1}\mathrm{E}[Y^*] = g^{-1}(\theta),$$

i.e. if the function $g(\cdot)$ is not linear, we cannot take it out of the integral that defines the expected value.

To give a more tangible example, consider $y^* = \log(y)$, where y is a variable representing skewed and positive costs and we model $y^* \sim \mathrm{Normal}(\eta, \lambda)$ — this is equivalent to assuming

$$y \sim \text{log-Normal}(\eta, \lambda) \qquad \text{with} \qquad p(y \mid \eta, \lambda) = \frac{1}{y\sqrt{2\pi\lambda^2}} \exp\left(-\frac{(\log y - \eta)^2}{2\lambda^2}\right). \tag{5.9}$$

Suppose further that we have obtained a set of simulations from the posterior distribution of the mean $\eta \mid y^*$ and standard deviation $\lambda \mid y^*$ (both defined on the log scale), say $\left(\eta^{(1)}, \ldots, \eta^{(S)}\right)$ and $\left(\lambda^{(1)}, \ldots, \lambda^{(S)}\right)$.

In this case, however, because the transformation $g(\cdot) = \log(\cdot)$ is non-linear, simply rescaling the simulated values by applying $g^{-1}(\cdot) = \log^{-1}(\cdot) = \exp(\cdot)$ does not directly target the quantity of interest, i.e. the mean of the costs on the natural scale, as exemplified by the following chain of relationships

$$\mathrm{E}[Y] = \mathrm{E}[\exp(\log(Y))] = \mathrm{E}[\exp(Y^*)] \neq \exp\left(\mathrm{E}[Y^*]\right) = \exp\left(\mathrm{E}[\log(Y)]\right) = \mathrm{median}(Y).$$

To overcome this problem, we can resort to a clever technique, usually referred to as *g-computation*. In a nutshell, this amounts to obtaining a Monte Carlo sample for the variable in question, starting from simulated values of the model parameters on the transformed scale.

In the example above, we can use each of the S simulated values $\left(\eta^{(1)}, \ldots, \eta^{(S)}\right)$ and $\left(\lambda^{(1)}, \ldots, \lambda^{(S)}\right)$ to generate N samples from the (predictive) distribution of y^*, as

$$
\begin{aligned}
&y^{*(1,1)} \sim \text{log-Normal}\left(\eta^{(1)}, \lambda^{(1)}\right), \ldots, y^{*(1,N)} \sim \text{log-Normal}\left(\eta^{(1)}, \lambda^{(1)}\right) \\
&y^{*(2,1)} \sim \text{log-Normal}\left(\eta^{(2)}, \lambda^{(2)}\right), \ldots, y^{*(2,N)} \sim \text{log-Normal}\left(\eta^{(2)}, \lambda^{(2)}\right) \\
&\vdots \\
&y^{*(S,1)} \sim \text{log-Normal}\left(\eta^{(S)}, \lambda^{(S)}\right), \ldots, y^{*(S,N)} \sim \text{log-Normal}\left(\eta^{(S)}, \lambda^{(S)}\right).
\end{aligned}
$$

Now that we have a full characterisation of the uncertainty in y^* for all the simulated combinations of the parameters, we can use the samples from the predictive on y^* to back-transform to the original scale, by simply applying the inverse function $g^{-1}(\cdot)$

$$
\begin{aligned}
&y^{(1,1)} = \exp\left(y^{*(1,1)}\right), \ldots, y^{(1,N)} = \exp\left(y^{*(1,N)}\right) \\
&y^{(2,1)} = \exp\left(y^{*(2,1)}\right), \ldots, y^{(2,N)} = \exp\left(y^{*(2,N)}\right) \\
&\vdots \\
&y^{(S,1)} = \exp\left(y^{*(S,1)}\right), \ldots, y^{(S,N)} = \exp\left(y^{*(S,N)}\right).
\end{aligned}
$$

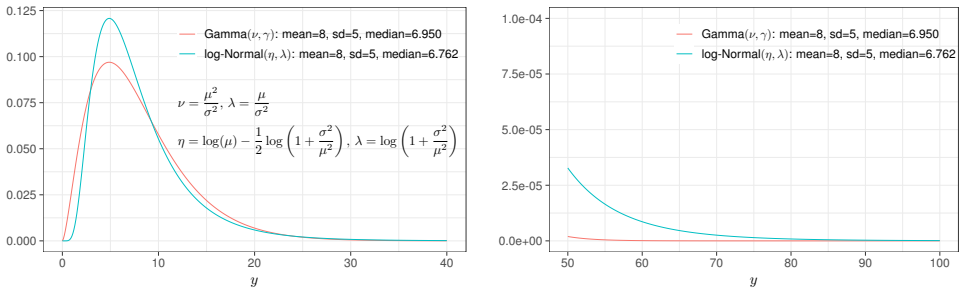
These, in turn, can be used to compute an empirical summary, for instance the mean as

$$
\begin{aligned}
&\mu^{(1)} = \frac{1}{N}\left(y^{*(1,1)} + y^{*(1,2)} + \ldots + y^{*(1,N)}\right) \\
&\mu^{(2)} = \frac{1}{N}\left(y^{*(2,1)} + y^{*(1,2)} + \ldots + y^{*(2,N)}\right) \\
&\vdots \\
&\mu^{(S)} = \frac{1}{N}\left(y^{*(S,1)} + y^{*(S,2)} + \ldots + y^{*(S,N)}\right)
\end{aligned}
$$

and the sample $\left(\mu^{(1)}, \ldots, \mu^{(S)}\right)$ now does characterise the distribution of the function of interest on the natural scale, in this case $\mathrm{E}[Y]$.

Gamma vs log-Normal distribution

The log-Normal distribution described above is another good candidate to describe the sampling variability or uncertainty for variables whose natural range is positive and skewed — typical examples in HTA include total resource costs, or, in some cases, HRQL-related measures.
While the shape of the Gamma and log-Normal distribution tends to be superficially similar, there is an important difference. By the mathematical properties of the two distributions, the log-Normal tends to have much heavier tails, meaning that it tends to assign higher probability to very large values of the underlying variable, as is obvious in Figure 5.6b.



(a) (b)

Figure 5.6: This plot shows the density for a log-Normal and a Gamma distributions with similar "natural" parameters. Panel (a) shows a larger range along both axes and from a cursory inspection, the two distributions look extremely close. In Panel (b), however, which focuses on the higher range along the x-axis, it is possible to see that the log-Normal has much fatter tails

This may be seen as an attracting feature, in case we want to safeguard from the theoretically justified possibility of having very large values (e.g. for the cost associated with a given treatment), even without having observed any in the data.
Generally speaking, however, this often implies that the Gamma is a safer option to model variables such as costs, limiting the potential impact of too large ranges in the estimates (Thompson and Nixon, 2005).

Figure 5.6a shows the formulae that can be used to relate the original-scale parameters (e.g. η, λ, for the log-Normal distribution) to the natural-scale parameters, i.e. the mean and standard deviation μ, σ on the scale of the data of interest — the relevant relationships for the Gamma distribution are presented also in Equation 1.6.

When we can write down these relationships analytically, then g-computation is not necessary — given simulations for η, λ, we can simply rescale them properly and obtain simulations for the relevant parameter. For instance, to retrieve samples from the distribution of the natural scale variable y we need to invert the relationship shown in Figure 5.6a and compute

$$\mu = \exp\left(\eta + \frac{1}{2}\lambda^2\right)$$

directly using the output of our model.

Example 5.5: *10TT (continued): Gamma/Gamma MCF model — g-computation..* We can compute the estimates for the population average benefits using g-computation and Monte Carlo simulation, using the following R code.

```
# Extracts the relevant parameters from the JAGS object
nu.e=model$gg_mcf$BUGSoutput$sims.list$nu.e
mustar.e=model$gg_mcf$BUGSoutput$sims.list$mustar.e

# Compute the "average" rate
rate=nu.e/mustar.e
```

```r
# g-computation for the mean on the original scale
# 1. defines the number of MC simulations for the outcome
N=4000
# 2. defines the dimension of the object 'mu' in which to store the means
mu=matrix(NA,nrow=nrow(rate),ncol=2)
# 3. loops over the simulations for 'nu.e' and 'rate'
for (i in 1:nrow(nu.e)) {
  for (t in 1:2) {
# Simulate 'mcsim' values of estar, for the current value of the parameters
    estar=rgamma(N,shape=nu.e[i,t],rate=rate[i,t])
# Computes the mean for the *inverse transformation*
    mu[i,t]=3-estar |> mean()
  }
}

# Computes the summary statistics
colnames(mu)=c("mu.e[1]","mu.e[2]")
mu |> bmhe::stats()
```

```
            mean         sd     2.5%   median    97.5%
mu.e[1] 1.583465 0.02132735 1.541709 1.582941 1.625392
mu.e[2] 1.567550 0.03354060 1.498750 1.568327 1.629786
```

As is possible to see, the g-computation version of this calculation aligns with the one provided directly in the JAGS model — but of course, in case of non-linear transformations for the outcomes, it would be necessary to either perform the back-transformation analytically, or by a similar numerical process as the one highlighted here.

5.3 Model selection

Why do we need model selection? Etc...

All models are wrong

Here talk about Box's mantra + make the point that perhaps Fisher et al may have (unintentionally) given the impression that stats was infallible and for any problem you'd have a model/test to use etc etc etc...
Bring the discussion on Eugenics too, when talking about Fisher and the founding fathers?...

5.3.1 Deviance Inflation Criterion (DIC)

One of the most popular tools to assess model fit and perform model selection in a Bayesian context is the *Deviance Information Criterion* (DIC, Spiegelhalter et al., 2002a). The DIC is based on the model deviance, defined as

$$D(\boldsymbol{\theta}) = -2 \log \mathcal{L}(\boldsymbol{\theta} \mid \boldsymbol{y}). \tag{5.10}$$

In isolation, the deviance is meaningless, because the scale on which the likelihood is defined varies with different models and parameterisations. Thus, reporting the value of a single model's deviance does not give information on its relative performance in terms of model selection. Comparatively, however, it can be used to assess how well a given model fit the observed data, with respect to an alternative one. Models associated with a lower deviance fit the observed data better.

Similarly to other information criteria, e.g. the *Akaike* Information Criterion (AIC, Akaike, 1974) or the *Bayesian* Information Criterion (BIC, Schwarz, 1978), the DIC is designed to safeguard against the phenomenon of *overfitting* (i.e. a model with a large number of parameters tends to do extremely well for the data at hand, but generally struggles to adapt to new/different data), by adding a penalty for model complexity.

Formally, in its original definition, the DIC is constructed as

$$\text{DIC} = p_D + \overline{D(\boldsymbol{\theta})} = D(\overline{\boldsymbol{\theta}}) + 2p_d, \tag{5.11}$$

where

$$\begin{aligned}\overline{D(\boldsymbol{\theta})} = \text{E}\left[D(\boldsymbol{\theta})\right] &= \int -2\log\mathcal{L}(\boldsymbol{\theta} \mid \boldsymbol{y})p(\boldsymbol{\theta} \mid \boldsymbol{y})d\boldsymbol{\theta} \\ &\approx \frac{1}{S}\sum_{s=1}^{S} -2\log\mathcal{L}(\boldsymbol{\theta}^{(s)})\end{aligned}$$

is the average model deviance (averaged over the posterior distribution of the model parameters, which can be approximated using a sample of S draws $\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(S)} \sim p(\boldsymbol{\theta} \mid \boldsymbol{y})$ obtained from the MCMC process);

$$\begin{aligned}D(\overline{\boldsymbol{\theta}}) = D(\text{E}\left[\boldsymbol{\theta} \mid \boldsymbol{y}\right]) &= -2\log\mathcal{L}\left(\int \boldsymbol{\theta}p(\boldsymbol{\theta} \mid \boldsymbol{y})d\boldsymbol{\theta}\right) \\ &\approx -2\log\mathcal{L}\left(\frac{1}{S}\sum_{s=1}^{S}\boldsymbol{\theta}^{(s)}\right)\end{aligned}$$

is the model deviance calculated in correspondence of the average value of the model parameters (which, again, can be approximated using the simulations from the MCMC process); and

$$p_D = \overline{D(\boldsymbol{\theta})} - D(\overline{\boldsymbol{\theta}}) \tag{5.12}$$

is the penalty for model complexity. This is the version used by `BUGS` when calculating the DIC — as evident by the note at the bottom of the outcome of the `print(mbugs,...)` instruction in Section 3.3. Equation 5.12 has an interesting direct interpretation as an estimate of the number of free parameters in the model, a quantity usually referred as "*effective number of model parameters*" — see Section 6.2 as well as Spiegelhalter et al. (2002a) and Lunn et al. (2013) for more details on this, in the context of hierarchical models.

HERE CRITICISM OF DIC AND HOW THE VERSION OF Plummer (2008) IS COMPUTED.

Gelman et al. (2013) suggest a slightly different definition for the penalty

$$p_V = \frac{1}{2}\text{Var}\left[D(\boldsymbol{\theta})\right], \tag{5.13}$$

which has the advantage of being invariant to re-parameterisations of the model and non-negative. This is the version used by JAGS, as reported at the bottom of the output for the `print(mjags,...)` command in Section 3.3 as well as in the outputs of all `print` commands in the examples above.

> **p_V vs p_D**
>
> Somewhat confusingly, earlier versions of the `R2jags` package still indicated the computed value for p_V with the general terminology `pD`. `R2jags` would highlight in the summary table produced by the `print` method that the computation was performed using Equation 5.13, but the naming convention was still confusing.
> This behaviour has changed in version `0.8-6`, since which `R2jags` explicitly distinguishes between `pV` (the default calculation) and `pD`, which is computed using the method described by Plummer (2008), a slightly different version than the original one defined by Spiegelhalter et al. (2002a).
> The computation of p_D performed by `rjags::dic.samples` is based on Plummer (2008), using an approach based on loss functions, a slightly different one to that of Spiegelhalter et al. (2002a), which usually provides near-identical results.

Equation 5.12 has the obvious advantage of being extremely easy to compute, once the deviance is monitored. In addition, in the case of weak prior information, it also approximates the effective number of model parameters. When the prior is not dominated by the likelihood, however, this results does not hold and the penalty can vary in ranges that are more complex to analyse (see Example 5.6). This implies that the scale of the penalty and thus of the DIC may be different when using BUGS (i.e. p_D), as opposed to the default specification of JAGS based on p_V.

5.3.2 Computing p_D manually

Irrespective of the software used to perform the MCMC analysis, we can still compute the penalty p_D by using the resulting simulations to estimate $\overline{D(\boldsymbol{\theta})}$ and $D(\overline{\boldsymbol{\theta}})$.

The first one is easy to obtain as the mean of the simulated values for the deviance. For instance, considering the model in Example 5.4, this can be easily extracted from the JAGS object, as we show below.

As for $D(\overline{\boldsymbol{\theta}})$, this is a bit more involved, because we first need to extract the mean value of all the model parameters and then compute the full likelihood function associated with these values.

In the case of the model in Example 5.4 this can be done using the following code. First we consider the part of the model that deals with the effects.

```
# 1. Model for the effects
# Extracts the shape parameter
nu.e=c(
  model$gg_mcf$BUGSoutput$summary["nu.e[1]","mean"],
  model$gg_mcf$BUGSoutput$summary["nu.e[2]","mean"]
)
# Extracts the regression coefficients
alpha0=model$gg_mcf$BUGSoutput$summary["alpha0","mean"]
alpha1=model$gg_mcf$BUGSoutput$summary["alpha1","mean"]
```

```
alpha2=model$gg_mcf$BUGSoutput$summary["alpha2","mean"]
# Computes the linear predictor
trt=data$Trt-1
u0=data$u0star
log.phi.e=alpha0 + alpha1*trt + alpha2*u0
# Computes the overall mean
mustar.e=numeric()
for (t in 1:2) {
  mustar.e[t] <- exp(alpha0 + alpha1*(t-1))
}
# Computes the rate parameter
gamma.e=numeric()
for (i in 1:data$N) {
  gamma.e[i]=nu.e[data$Trt[i]]/exp(log.phi.e[i])
}
# Computes the likelihood contribution from each observation
lik.e=-2*dgamma(data$estar,shape=nu.e[data$Trt],rate=gamma.e,log=TRUE)
```

The R code effectively replicates the constructions described for the JAGS model to use the mean value of the shape parameter nu.e and the linear predictor log.phi.e to derive the rate parameter gamma.e.

We then use the R built-in function dgamma to compute the likelihood given the observed data data$estar and the shape and rate parameters computed at the average value. Note that this code computes the full likelihood, including any constant that does not depend on the parameters — for this reason we can use the command dgamma, which in fact amounts to the sampling distribution of Equation 1.2 — and, incidentally, this is the same strategy used by BUGS when computing p_D. The option log=TRUE computes the likelihood on the log scale and by multiplying it by -2 we obtain the individual contribution for the deviance with respect to the effects.

We can replicate this computation for the part of the model that deals with the costs, as in the code below.

```
# 2. Model for the costs
# Extracts the shape parameter
nu.c=c(
  model$gg_mcf$BUGSoutput$summary["nu.c[1]","mean"],
  model$gg_mcf$BUGSoutput$summary["nu.c[2]","mean"]
)
# Extracts the regression coefficients
beta0=model$gg_mcf$BUGSoutput$summary["beta0","mean"]
beta1=model$gg_mcf$BUGSoutput$summary["beta1","mean"]
beta2=model$gg_mcf$BUGSoutput$summary["beta2","mean"]
# Computes the linear predictor
log.phi.c=beta0 + beta1*(data$estar-mustar.e[data$Trt]) + beta2*(trt)
# Computes the rate parameter
gamma.c=numeric()
for (i in 1:data$N) {
  gamma.c[i]=nu.c[data$Trt[i]]/exp(log.phi.c[i])
}
```

```
# Computes the likelihood contribution from each observation
lik.c=-2*dgamma(data$c,shape=nu.c[data$Trt],rate=gamma.c,log=TRUE)
```

The objects `lik.e` and `lik.c` are vectors of length 167 — one value for each individual in the data.

Once we have an estimate for these, we can simply sum them up to obtain an estimate for $D(\overline{\boldsymbol{\theta}})$, which we indicate as `dhat` in the code below. Together with the estimate for $\overline{D(\boldsymbol{\theta})}$, indicated as `dbar` and obtained by simply extracting the average for the posterior distribution of the deviance, we can simply compute p_D and the DIC, according to Equation 5.12 and Equation 5.11, using the code below.

```
# 3. Finally computes pD & DIC
# Computes the average model deviance
dbar=model$gg_mcf$BUGSoutput$summary["deviance","mean"]
# Computes the deviance at the average value for the parameters
dhat=sum(lik.e) + sum(lik.c)
# Computes pD
pD=dbar-dhat
# Computes DIC
DIC=dbar+pD
```

The computed values are $p_D = 10.04$ and DIC $= 2797.4$, which in this case are fairly similar to the estimates provided by `JAGS` (which can be found in the summary table in Example 5.4), although this is not necessarily always the case, as we show in Section 6.2.6.

While with a bit of coding and maths to obtain the relevant values for the deviance we can always post-pocess a `JAGS` object and obtain an estimate for p_D, we do not necessarily need to do so and instead obtain an approximation to the value of p_D as would be computed by `BUGS` (and thus a more direct estimate of the number of effective parameters) using the function `dic.samples` from the `rjags` package (which is automatically loaded when `R2jags` is).

```
rjags::dic.samples(object$model,n.iter=1000,type="pD")
```

where `object` is a `R2jags` object (e.g. `mjags` above) and `n.iter` is the number of iterations to be used to make the computation. When `type=pD` (which is the default), the element `penalty` in the output object is an approximation to p_D.

In addition, since version `0.8-6`, `R2jags` embeds this calculation in the call to the function `jags(...)`, where it is possible to specify an input `pD=TRUE`. If that is the case, then `R2jags` will first run the model and then `dic.samples()` to compute the penalty p_D, which is stored in the resulting `JAGS` object, alongside the default p_V, as we demonstrate below.

Example 5.6: *10TT (continued): model selection.* Comparing the models in Example 5.2 and in Example 5.3 we can see from the summary tables that they have basically identical DIC: 3136 and 3140, respectively, confirming the fact that they are effectively indistinguishable (as evident from Figure 5.5).

If we include in the mix the model in Example 5.4, the situation changes as the DIC for the Gamma/Gamma MCF version is substantially lower, with a computed value of 2801 — that is a difference of over 300 points with respect to the two versions of the Normal/Normal model, indicating a much better fit for the Gamma/Gamma MCF. In addition, this has

also a much lower value for p_V, computed at 13.2, which is over 6 times smaller than the value computed for the Normal/Normal models.

The main reason for this apparent large discrepancy is that, in Example 5.2 and Example 5.3, we have included some substantial information in the definition of the PC priors for the standard deviations, while all the model parameters have fairly vague priors in Example 5.4. Consequently, in the two Normal/Normal cases, the value for the penalty estimated by `JAGS` using the rule in Equation 5.13 cannot to be interpreted directly as a measure of the number of parameters in the model — in fact, the nominal number of fundamental parameters is 9 (`alpha0`, `alpha1`, `alpha2`, `beta0`, `beta1`, `sigma.e[1]`, `sigma.e[2]`, `sigma.c[1]` and `sigma.c[2]`) for the model in Example 5.2 and 10 (with `beta2` in addition to the ones above) for the model in Example 5.3, both much smaller numbers than the values computed for p_V, which are 72.7 and 78.7, respectively.

As mentioned above, we can directly compute p_D from `R2jags` by adding the option `pD=TRUE` to the call to the function `jags`. As all the models above have reached reasonable convergence, we can keep the other optional argument `n.iter.pd` to its default (1000 extra simulations) and re-run the three models. For instance, for the Normal/Normal independence model, the revised call to `jags` would be as in the following code.

```
# Re-runs the model adding the option pD=TRUE
model$nn_indep=jags(
  data=data,
  parameters.to.save=c(
    "mu.e","mu.c","alpha0","alpha1","alpha2","beta0","beta2","sigma.e","sigma.c"
  ),
  inits=NULL,n.chains=2,n.iter=5000,n.burnin=3000,n.thin=1,DIC=TRUE,
  # This specifies the model code as the function 'nn_indep'
  model.file=nn_indep,pD=TRUE
)

# Shows the summary table, including the estimate for pD
print(model$nn_indep,digits=3,interval=c(0.025,0.5,0.975))
```

```
Inference for Bugs model at "/tmp/RtmpVHriMa/model415f73a4774fa.txt",
 2 chains, each with 5000 iterations (first 3000 discarded)
 n.sims = 4000 iterations saved. Running time = 1.576 secs
            mu.vect sd.vect     2.5%      50%    97.5%  Rhat n.eff
alpha0        1.551   0.021    1.512    1.551    1.593 1.001  2900
alpha1       -0.006   0.039   -0.081   -0.006    0.073 1.001  3600
alpha2        1.280   0.073    1.135    1.280    1.425 1.001  4000
beta0       641.568  87.984  467.221  641.044  808.987 1.001  4000
beta2       169.641  95.622  -11.662  168.909  358.933 1.001  4000
mu.c[1]     641.568  87.984  467.221  641.044  808.987 1.001  4000
mu.c[2]     811.210 127.840  556.537  813.916 1067.923 1.001  4000
mu.e[1]       1.551   0.021    1.512    1.551    1.593 1.001  2900
mu.e[2]       1.545   0.033    1.481    1.545    1.612 1.001  4000
sigma.c[1] 1403.856  83.089 1255.225 1400.548 1574.894 1.002  1500
sigma.c[2] 2684.073 153.670 2400.495 2682.820 2989.463 1.020   130
sigma.e[1]    0.208   0.015    0.183    0.207    0.240 1.002  1900
sigma.e[2]    0.267   0.024    0.224    0.265    0.316 1.001  4000
deviance   3062.379  12.053 3040.533 3061.971 3086.932 1.006   280
```

```
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: pV = var(deviance)/2)
pV = 72.4 and DIC = 3134.8
DIC info (using the rule: pD = Dbar-Dhat, computed via 'rjags::dic.samples')
pD = 7.2 and DIC = 3071.9
DIC is an estimate of expected predictive error (lower deviance is better).
```

The summary table now includes at the bottom the values for p_D and the revised DIC — notice that, because `jags` needs to re-runs the model to compute p_D, the DIC associated with it is computed based on a new (possibly slightly different) distribution of the deviance. If the model is at convergence, the discrepancy is minimal. For the same reason, the reported values for p_V and DIC are also slightly different to the ones in Example 5.2, but these differences are only down to Monte Carlo error (due to the new MCMC run used to construct this table).

Table 5.3: Model selection via DIC, using the two versions — the first one based on p_V and the second based on p_D. The values reported for p_V are those obtained in Examples Example 5.2, Example 5.3 and Example 5.4, based on the original model runs

Model	p_V[a]	DIC[b]	p_D	DIC[d]
Normal/Normal independent	72.69	3136	7.19	3072
Normal/Normal MCF	78.73	3140	7.13	3069
Gamma/Gamma MCF	13.24	2801	10.76	2798

[a] Computed in 5.2, 5.3 and 5.4, respectively
[b] Computed as $p_V + \overline{D(\boldsymbol{\theta})}$
[d] Computed as $p_D + \overline{D(\boldsymbol{\theta})}$

We can do the same for all three models considered here. The resulting values for the model fitting statistics are presented in Table 5.3. As is possible to see, exactly as was happening before, the relative differences in the re-computed DICs are minimal between the two Normal/Normal models and very large against the Gamma/Gamma MCF model, confirming its better performance over the others.

However, the three values of the effective number of parameters are now closer — and closer to the nominal number of model parameters. For the Gamma/Gamma MCF, the value computed by `R2jags::jags()` is also close (but not identical, again due to simulation error) to the one we obtained manually in Section 5.3.2.

What if?...

Incidentally, if we included some substantial information in the prior for the shape of the effects, in the Gamma/Gamma MCF model of Example 5.4, e.g. $\nu_{et} \sim$ Exponential(6), to encode the assumption that $\Pr(\nu_{et} > 0.5) \approx 0.05$, then we would get estimates for the DIC of 3113, for p_V of 139.9 and for p_D of 10.8.

Not only would this change the scale of the estimate for p_V, but also the ordering of the three models — this new version of the Gamma/Gamma MCF would have worse performance than the two Normal/Normal models!

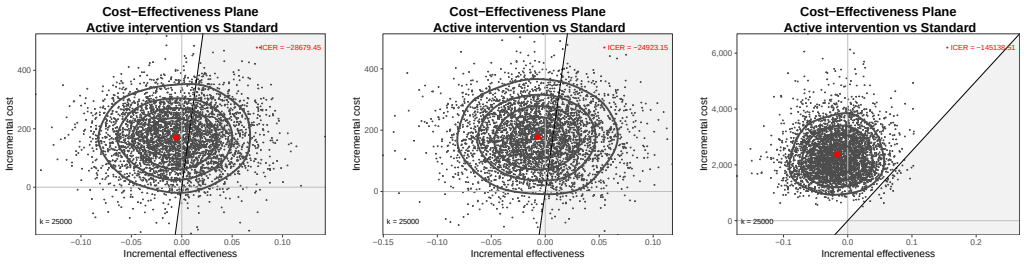
5.4 Cost-effectiveness modelling

We can post-process the outcome by accessing directly the simulations from the BUGS model, for instance stored in the object `model$sims.list`. For instance, we could construct the relevant variables (Δ_e, Δ_c) as

```
delta_e=model$nn_indep$BUGSoutput$sims.list$mu.e[,2]-
  model$BUGSoutput$sims.list$mu.e[,1]
delta_c=model$nn_indep$BUGSoutput$sims.list$mu.c[,2]-
  model$BUGSoutput$sims.list$mu.c[,1]
```

and then we could use these to, for instance, plot the Cost-Effectiveness plane. Alternatively, we can use the R package BCEA, which means we would only need to feed the simulations to the function `bcea` to then produce a range of standardised output for the economic evaluation. For instance, we could use the following R code.

Figure 5.7 shows the cost-effectiveness plane with overlaid a contour for the joint posterior distribution of (Δ_e, Δ_c).



(a) (b) (c)

Figure 5.7: The contour plot overlaid on the cost-effectiveness plane

5.5 Conclusions

Say something + what's next