# Markov Models for Health Economic Evaluation: The R Package heemod

**Antoine Filipović-Pierucci**
URC-Eco

**Kevin Zarca**
URC-Eco

**Isabelle Durand-Zaleski**
URC-Eco

**Abstract**

Health economic evaluation studies are widely used in public health to assess health strategies in terms of their cost-effectiveness and inform public policies. We developed an R package for Markov models implementing most of the modelling and reporting features described in reference textbooks and guidelines: deterministic and probabilistic sensitivity analysis, heterogeneity analysis, time dependency on state-time and model-time (semi-Markov and non-homogeneous Markov models), etc. In this paper we illustrate the features of **heemod** by building and analysing an example Markov model. We then explain the design and the underlying implementation of the package.

*Keywords*: health economic evaluation, Markov models, R.

## 1. Introduction

Health economic evaluation studies are widely used in public health to assess healthcare strategies in terms of their cost-effectiveness and inform public policies (Russell *et al.* 1996). In order to account for the long-term consequences of healthcare strategies, models are needed to extrapolate results to a longer time frame (Sonnenberg and Beck 1993; Eddy *et al.* 2012).[1]

Models compute the health status of a population over time (e.g. healthy, sick, dead) and how their health is affected by different strategies. Costs (e.g. medical or drug costs) and outcomes (e.g. life years or quality of life) are attached to each distinct health status, allowing to estimate the cost and effectiveness expected for every studied strategy.

By representing health status as states and health changes over time as transitions probabilities between states this process can be modelled using Markov chains, called a Markov model. Transition probabilities between states can be described by a square 2-dimensional transition matrix $T$, where element $i, j$ is the transition probability between state $i$ and $j$. The probability of being in a given state at time $t$ is given by:

$$X \times T^t \tag{1}$$

[1] Usually the entire target population lifetime.

Where $X$ is a vector[2] giving the probability of being in a given state at the start of the model, and $T^t$ is the product of multiplying $t$ matrices $T$. The use of Markov models in health economic evaluation have been thoroughly described in Beck and Pauker (1983), Sonnenberg and Beck (1993) and Briggs and Sculpher (1998).

In order to best inform the decision process Markov models should incorporate a wide range of information to account for all the available evidence at a given time (Briggs and Sculpher 1998). Results from various sources can be combined, such as estimated drug efficacy from clinical trials, disease evolution from epidemiological cohorts, quality of life values from population-level studies, transition probabilities from life-tables, etc. An implementation of Markov models should be flexible enough to receive all these sources of data.

Most Markov models are built using basic spreadsheet software such as Microsoft Excel (Microsoft 2016), which has drawbacks: analyses are hard to reproduce and lack transparency, errors are difficult to spot, track and correct, and graphic capabilities are lacking (Williams *et al.* 2016a). The R language (R Core Team 2016) can overcome these issues through script-based approaches: there is a written record of what was done, the calculations are transparent, modification can be easily applied to the model, traceability is guaranteed,[3] and the code just needs to be run again to reproduce the analysis (Williams *et al.* 2016a). But R has not been much used because no package implementing Markov models in health economic evaluation exists so far, and programming Markov models from scratch is not trivial.

Our objective was to develop an R package for Markov models implementing most of the modelling and reporting features described in reference textbooks (Drummond *et al.* 2005; Briggs *et al.* 2006) and guidelines (Eddy *et al.* 2012; Husereau *et al.* 2013). We named the package **heemod**, standing for Health Economic Evaluation MODelling. The package is available from the Comprehensive R Archive Network (CRAN) at `http://CRAN.R-project.org/package=heemod`.

In Section 2 we illustrate the features of **heemod** by building and analysing an example Markov model. Then in Section 3 we explain the design and the underlying implementation of the package.

## 2. Building and analysing a model: an example

In this section we use a simplified example to illustrate how and why Markov models are used in health economic evaluation studies. We introduce theoretical concepts and methods along as they are encountered, present the production of results with the **heemod** package, and their interpretation. The entire workflow is then summarised in a chart presented in Figure 8.

### 2.1. Description of the question

For this example we will model the imaginary disease called *shame*, which is still a terminal disease in some parts of the Galaxy (Adams 1979). At the onset the disease is asymptomatic: patients are quite unashamed, they do not feel sick, have a good quality of life, and are not likely to die of shame. But patients are at risk of being ashamed: that marks the entry into

[2] Of length equal to the number of states.

[3] Especially with the help of version control software such as git (Torvalds and Hamano 2017) that are particularly well suited to script files.

the symptomatic phase of the disease, with frequent hospital stays, deteriorated quality of life and a high risk of dying of shame. The probability to revert to the asymptomatic unashamed state is unfortunately quite low, and there is no cure known to work reliably: to be effective shame therapy should thus be provided during the initial asymptomatic state, to prevent being ashamed in the first place.

From the description of the disease we can define 3 states:

**Asymptomatic state (`pre`):** Before the symptomatic state, when treatment can still be provided.

**Symptomatic state (`symp`):** Symptomatic disease, after being ashamed. With degraded health, high hospital costs and increased probability of dying of shame.

**Death (`death`):** Death by natural causes or because of shame.

Three strategies are proposed to prevent being ashamed:

**Base strategy (`base`):** Do nothing, this is the natural evolution of the disease.

**Medical treatment (`med`):** Patients with asymptomatic disease take a drug all their life to lower the risk of being ashamed.

**Surgical treatment (`surg`):** Patients with asymptomatic disease undergo a surgical procedure that lowers the risk of being ashamed (shamectomy), the procedure needs to be performed only once.

Medical treatment is effective in preventing symptomatic disease, but the drug is expensive. Surgical treatment is a one-time cost, but its effect decreases with time. The increased probability of dying of shame once in the symptomatic disease state does not depend on the treatment used before, when the disease was asymptomatic.

In this model we will use a cycle duration of 1 year: we must take care that the transitions probabilities and the values attached to states are calculated on this time-frame.

We can now start to build the model.

```
R> library(heemod)
```

## 2.2. Transitions

We start by defining a transition matrix for the `base` strategy with the `define_transition()` function.

```
R> mat_base <- define_transition(
R+   state_names = c("pre", "symp", "death"),
R+
R+   C,       p_disease_base, p_death_all,
R+   p_cured, C,              p_death_symp,
R+   0,       0,              1)
```

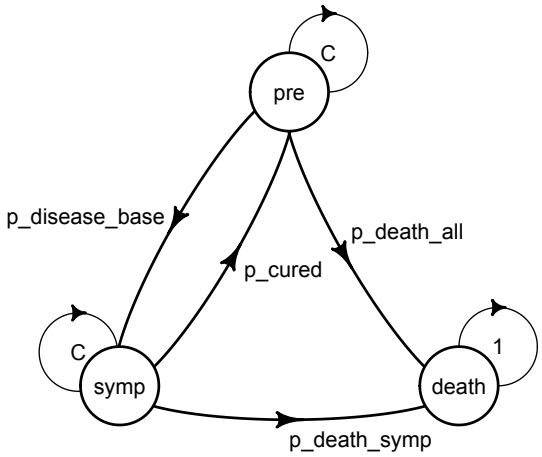


Figure 1: Transition diagram for the `base` strategy.

```
R> mat_base
```

```
A transition matrix, 3 states.

      pre     symp           death
pre   C       p_disease_base p_death_all
symp  p_cured C              p_death_symp
death                        1
```

`p_disease_base` is the probability of being ashamed in the `base` strategy, `p_death_all` the all cause probability of death (not caused by the disease) and `p_death_symp` the death probability in the symptomatic state (greater than `p_death_all`). `p_cured` is the unlikely probability to revert to the asymptomatic unashamed state. `p_cured`, `p_death_symp` and `p_death_all` do not depend on the strategy. The value of these parameters will be defined later. `C` is an alias for the probability complement, 1 minus the sum of probabilities in a given row. Once dead we assumed coming back to another state was impossible: the backward probabilities were set to 0, and the probability to stay dead to 1. The resulting transition diagram is presented in Figure 1.[4]

Similarly, transitions can be defined for the other 2 strategies. In our case only the name of the probabilities change: `p_disease_base` becomes `p_disease_med` or `p_disease_surg` (those parameters will also be defined later).

```
R> mat_med <- define_transition(
R+   state_names = c("pre", "symp", "death"),
R+
R+   C,       p_disease_med, p_death_all,
R+   p_cured, C,             p_death_symp,
R+   0,       0,             1)
```

[4]Generated by `plot(mat_base)`.

```
R> mat_surg <- define_transition(
R+   state_names = c("pre", "symp", "death"),
R+
R+   C,       p_disease_surg, p_death_all,
R+   p_cured, C,              p_death_symp,
R+   0,       0,              1)
```

### 2.3. State values

Next we define the values associated with states using the `define_state` function. An arbitrary number of values can be attached to a state, here we define: `cost_treat` the treatment cost (drug costs for the `med` strategy or surgery costs for the `surg` strategy, there is no treatment in the `base` strategy), `cost_hospit` the hospitalization costs, `cost_total` the total cost, and `qaly` the health-related quality-adjusted life years (QALY), where 1 stands for one year in perfect health and 0 stands for death (Torrance and Feeny 1989). In the following code we define the state `pre`:

```
R> state_pre <- define_state(
R+   cost_treat  = dispatch_strategy(
R+     base = 0,
R+     med  = cost_med,
R+     surg = cost_surg_cycle),
R+   cost_hospit = 0,
R+   cost_total  = discount(cost_treat + cost_hospit, r = dr),
R+   qaly        = 1)
```

To dispatch the cost of treatment according to the strategy we used the `dispatch_strategy()` function. Another approach would have been to define 3 versions of `state_pre`, one per strategy, and in the next section use the corresponding version in each distinct `define_strategy()` call.

The total cost is discounted with the `discount()` function at a given rate (`dr`). The QALY attached to 1 year in this state are set to 1, corresponding to 1 year in perfect health. The variables `dr`, `cost_med` and `cost_surg_cycle` will be defined later.

The other 2 states are defined similarly:

```
R> state_symp <- define_state(
R+   cost_treat  = 0,
R+   cost_hospit = cost_hospit_cycle,
R+   cost_total  = discount(cost_treat + cost_hospit, r = dr),
R+   qaly        = qaly_disease)

R> state_death <- define_state(
R+   cost_treat  = 0,
R+   cost_hospit = 0,
R+   cost_total  = discount(cost_treat + cost_hospit, r = dr),
R+   qaly        = 0)
```

Patients have a degraded quality of life and are hospitalized during the symptomatic disease phase, we need to define specific QALYs (`qaly_disease`) and hospital costs (`cost_hospit`) for this state. These variables will be defined later. Finally dead patients have QALYs at 0, and they do not cost anything to the healthcare system.

### 2.4. Strategies

All the information (states and transitions) is now available to define the strategies. For this purpose we use the `define_strategy()` function. Only the transition objects differ between `strat_base`, `strat_med` and `strat_surg`.

```
R> strat_base <- define_strategy(
R+   transition = mat_base,
R+
R+   pre   = state_pre,
R+   symp  = state_symp,
R+   death = state_death)

R> strat_med <- define_strategy(
R+   transition = mat_med,
R+
R+   pre   = state_pre,
R+   symp  = state_symp,
R+   death = state_death)

R> strat_surg <- define_strategy(
R+   transition = mat_surg,
R+
R+   pre   = state_pre,
R+   symp  = state_symp,
R+   death = state_death)
```

### 2.5. Model parameters

We referenced parameters (such as `qaly_disease` or `p_disease_base`) in the transitions and the state definitions. It is now necessary to define the parameter values in an object. Because we said in the disease description that some probabilities and values vary with time, we need to introduce some concepts regarding time-dependency before we can define the parameters.

Transition probabilities or state values may change with time (e.g. the protecting effect of surgery may decrease with time after the procedure, probability of all-causes death may increase as the population gets older, hospital costs may change with disease evolution). It is thus important to account for time-dependency in order to build accurate models. In Markov models values may depend on 2 distinct measurements of time (Hawkins *et al.* 2005): time elapsed since the start of the model (called *model time*), and time spent in a given state (called *state time*). Both situations can co-exist in a same model.

In **heemod** time-dependency is specified with 2 variables: `model_time` and `state_time`. These package-reserved names return sequential values starting from 1, corresponding to time spent

in the model for *model time* and time spent in a given state for *state time*. They can be used in any user-defined expression or function.

In our case the probability of all-cause death depends on age. Because age increases with time spent since the beginning of the model, all-cause death probability is *model time* dependent. On the other hand the probability of dying of shame depends on time elapsed after being ashamed (i.e. time spent in the `symp` state): this probability is *state time* dependent. The probability of being ashamed after surgery, the cost of surgery, and the hospital costs in the symptomatic state also depend on the time spent in their respective state.

We can now create the global parameters with `define_parameters()`:

```
R> par_mod <- define_parameters(
R+   age_base  = 20,
R+   age_cycle = model_time + age_base)
```

The age of individuals for a given cycle `age_cycle` is the age at the beginning of the model (`age_base`), plus the time the model has run (`model_time`).

```
R> par_mod <- modify(
R+   par_mod,
R+
R+   sex_indiv   = "MLE",
R+   p_death_all = get_who_mr(
R+     age     = age_cycle,
R+     sex     = sex_indiv,
R+     country = "GBR",
R+     local   = TRUE))
```

The death probability `p_death_all`, as a function of age and sex, is fetched from the World Health Organisation database with `get_who_mr()`,[5] here for a British population.[6]

```
R> par_mod <- modify(
R+   par_mod,
R+
R+   p_death_disease = get_probs_from_surv(
R+     fit_death_disease,
R+     cycle    = state_time,
R+     km_limit = 5))
```

The probability of dying of shame when the disease is symptomatic `p_death_disease` is extracted with the `get_probs_from_surv()` function from `fit_death_disease`, a model fitted with the **flexsurv** package (Jackson 2016). Because this probability depends on time spent in the `disease` state the `state_time` model variable is used to specify time. Here we use non-parametric Kaplan-Meier estimates for the first 5 years instead of model-fitted values with `km_limit = 5`.

[5] Relying on the **rgho** package (Filipovic-Pierucci 2017).

[6] We specify the use of local data cached in **heemod** with `local = TRUE` to avoid adding overhead query time.

The parametric survival model `fit_death_disease` used to compute `p_death_disease` is fitted with the following code:

```
R> fit_death_disease <- flexsurv::flexsurvreg(
R+   survival::Surv(time, status) ~ 1,
R+   dist = "weibull",
R+   data = tab_surv)
```

Where `tab_surv` is a data-frame containing survival data.

```
R> par_mod <- modify(
R+   par_mod,
R+
R+   p_death_symp = combine_probs(
R+     p_death_all,
R+     p_death_disease))
```

The death probability in the symptomatic state `p_death_symp` is the probability to die either from old age (`p_death_all`) or from the disease (`p_death_symp`). Assuming those probabilities are independent, we use the `combine_probs()` to combine them with the formula $P(A \cup B) = 1 - \big(1 - P(A)\big) \times \big(1 - P(B)\big)$.

```
R> par_mod <- modify(
R+   par_mod,
R+
R+   p_disease_base = 0.25,
R+   med_effect     = 0.5,
R+   p_disease_med  = p_disease_base * med_effect)
```

The probability of disease under medical treatment `p_disease_med` is the base probability of disease `p_disease_base` times the protecting effect of the treatment, `med_effect`.

```
R> par_mod <- modify(
R+   par_mod,
R+
R+   shape = 1.5, # We will see later why we need
R+   scale = 5,   # to define these 2 parameters here.
R+   p_disease_surg = get_probs_from_surv(
R+     define_survival(
R+       distribution = "weibull",
R+       shape        = shape,
R+       scale        = scale),
R+     cycle = state_time))
```

The probability of disease after surgery is extracted with `get_probs_from_surv()` from a parametric Weibull survival model defined with `define_survival()`. For reason explained in Section 2.8 parameters `scale` and `shape` are not written in the `define_survival()` call, but defined separately.

```
R> par_mod <- modify(
R+   par_mod,
R+
R+   cost_surg       = 20000,
R+   cost_surg_cycle = ifelse(state_time == 1, cost_surg, 0))
```

Because surgery is only performed once at the beginning of the `pre` state, the time-dependant variable `state_time` was used to limit surgery costs to the first cycle in the `pre` state.

```
R> par_mod <- modify(
R+   par_mod,
R+
R+   cost_hospit_start = 11000,
R+   cost_hospit_end   = 9000,
R+   n_years           = 9,
R+   cost_hospit_cycle = ifelse(
R+     state_time < n_years,
R+     cost_hospit_start,
R+     cost_hospit_end))
```

After `n_years` in the symptomatic state the symptoms become milder and hospital costs decrease (from `cost_hospit_start` to `cost_hospit_end`). We used the time-dependant variable `state_time` to condition the hospital costs `cost_hospit` on `n_years`.

```
R> par_mod <- modify(
R+   par_mod,
R+
R+   p_cured      = 0.001,
R+   cost_med     = 5000,
R+   dr           = 0.05,
R+   qaly_disease = 0.5)
```

Finally we define `p_cured` the probability to spontaneously revert to the asymptomatic unashamed state, `cost_med` the drug costs, `dr` the discount rate and `qaly_disease` the QALY for one year in the symptomatic state.

## 2.6. Running the model

The model can then be run with `run_model()`:

```
R> res_mod <- run_model(
R+   parameters = par_mod,
R+
R+   base = strat_base,
R+   med = strat_med,
R+   surg = strat_surg,
R+
R+   cycles = 10,
R+
R+   cost = cost_total,
R+   effect = qaly,
R+
R+   method = "life-table")

base: detected use of 'state_time', expanding states: pre, symp.

Fetching mortality data from package cached data.

Using cached data from year 2015.

med: detected use of 'state_time', expanding states: pre, symp.

surg: detected use of 'state_time', expanding states: pre, symp.
```

We define `cost_total` and `qaly` as the respective cost and effectiveness result. The model is run for 10 cycles (i.e. 10 years), and state membership counts are corrected using the life-table method (Barendregt 2009).

## 2.7. Results interpretation

*How do the strategies compare to each other with regard to their relative cost and effectiveness?*

The answer is given by calculating the total expected cost and effectiveness of all strategies, and then computing the incremental cost-effectiveness ratio (ICER) between them (Drummond *et al.* 2005). The ICER between strategies $A$ and $B$ is defined as:

$$\frac{C_B - C_A}{E_B - E_A}$$

Where $C$ is the total expected cost of a strategy and $E$ its total expected effect (e.g. sum of life-years of the population). Thus the ICER is the cost of an incremental unit of effectiveness. The most cost-effective strategy is (1) the most effective strategy (2) among the strategies having an ICER no higher than a threshold. This ICER threshold is the maximal willingness to pay for an additional unit of effectiveness: it is a political choice that depends on multiple factors (Claxton *et al.* 2015).

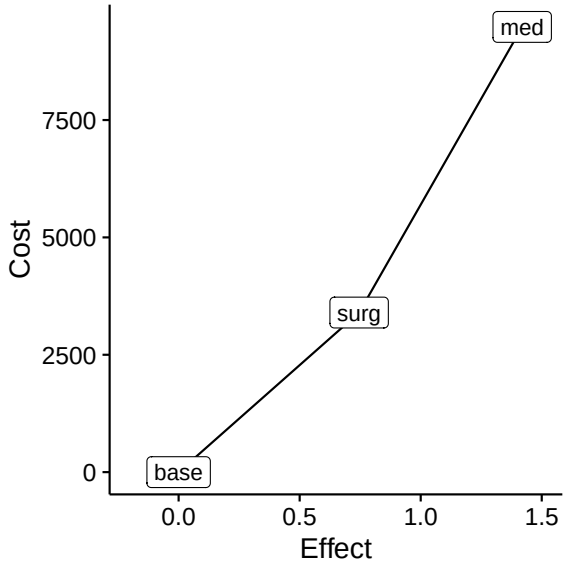


Figure 2: Incremental cost and effect of strategies on the cost-effectiveness plane, taking the `base` strategy as a reference.

The strategies can be presented on a cost-effectiveness plane (Figure 2),[7] were we see that both `med` and `surg` are more effective than `base`, but more costly.

```
R> summary(res_mod, threshold = c(1000, 5000, 15000))

3 strategies run for 10 cycles.

Initial state counts:

         N
pre   1000
symp     0
death    0

Counting method: 'life-table'.

Values:

     cost_treat cost_hospit cost_total     qaly
base          0    54614289   42809363 5811.957
med    27619118    37320805   52292756 7231.178
surg   10074542    47502867   46208450 6557.615

Net monetary benefit:
```

[7] Generated by `plot(res_mod, type = "ce")`.

```
          1000      5000    15000
base -36997.41 -13749.58 44369.99
surg -39650.84 -13420.38 52155.77
med  -45061.58 -16136.87 56174.92

Efficiency frontier:

base -> surg -> med

Differences:

     Cost Diff. Effect Diff.     ICER Ref.
surg   3399.088    0.7456579 4558.508 base
med    6084.306    0.6735634 9033.012 surg
```

From the printed model output presented above we see in the `ICER` column of the `Differences` section that `surg` is more cost-effective than `base` if one is willing to pay 4,559 more per QALY gained. Furthermore `med` is more cost-effective than `surg` if one is willing to pay 9,033 more per QALY gained.

A net monetary benefit analysis (Stinnett and Mullahy 1998) is run by specifying thresholds ICER values in the `summary()` function with the `threshold` argument. We see in the `Net monetary benefit` section that at a threshold ICER of 1,000 the strategy with the highest net monetary benefit is `base`, at 5,000 `surg` and at 15,000 `med`.

Figure 3 gives us more information about what happens in our model: the effect of surgery seems to wear down with time compared to the medical treatment.[8] Surgery delays the outcome, reporting degraded health status and hospital costs further in time. After a few years the hospital costs of the surgery strategy reach similar levels to the base strategy. Nevertheless these increased hospital costs do not outweigh the important treatment costs associated with the medical therapy.

## 2.8. Uncertainty analysis

*What is the uncertainty of these results? What strategy is probably the most cost-effective?*

Uncertainty of the results originate from uncertainty regarding the true value of input parameters (e.g. treatment effect, cost of hospital stays, quality of life with the disease, survival probabilities). The effect of this uncertainty can be assessed by varying parameter values and computing the model results with these new inputs. While multiple methods exist to study uncertainty (Briggs *et al.* 1994), deterministic and probabilistic sensitivity analysis (DSA and PSA) are the most widely used (Briggs *et al.* 2006).

In DSA parameter values are changed one by one, usually to a low and high value. Model results are plotted on a *tornado plot* to display how a change in the value of one parameter impacts the model results. DSA gives a good sense of the relative impact of each parameter on uncertainty, but does not account for total uncertainty over all the parameters, for skewed or complex parameter distribution, nor for correlation of error between parameters (Briggs *et al.* 1994).

[8] Generated by `plot(res_mod, type = "counts")` and `plot(res_mod, type = "values")`.

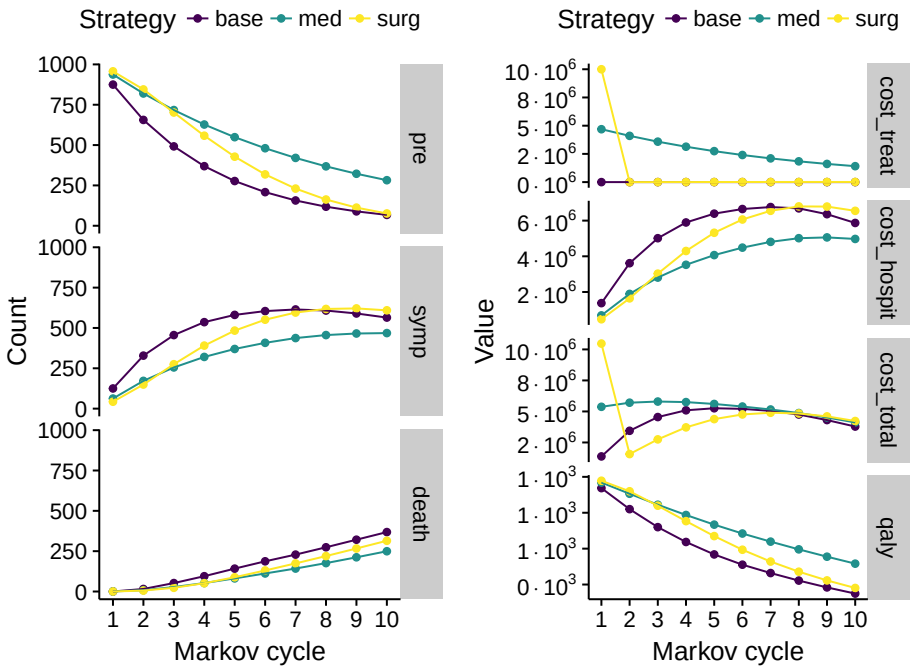


Figure 3: Evolution over time of counts by state (A) and of values (B).

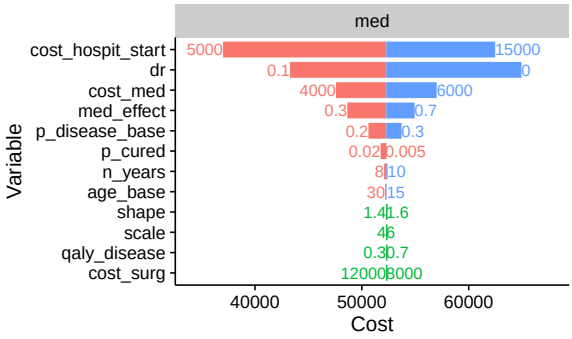


Figure 4: Tornado plot presenting uncertainty of the cost for the `med` strategy. Each line shows how setting the parameter to its low and high value impacts costs. The bars are centred around baseline costs.

We define the DSA with `define_dsa()` by specifying a lower and upper bound for each parameter of interest.

```
R> def_dsa <- define_dsa(
R+   age_base,           15,    30,
R+   p_disease_base,     0.2,   0.3,
R+   p_cured,           0.005, 0.02,
R+   med_effect,        0.3,   0.7,
R+   shape,             1.4,   1.6,
R+   scale,             4,     6,
R+   cost_med,          4000,  6000,
R+   cost_surg,         8000,  12000,
R+   cost_hospit_start, 5000,  15000,
R+   dr,                0,     0.1,
R+   qaly_disease,      0.3,   0.7,
R+   n_years,           8,     10)

R> res_dsa <- run_dsa(res_mod, dsa = def_dsa)

Running DSA on strategy 'base'...

Running DSA on strategy 'med'...

Running DSA on strategy 'surg'...
```

Only parameters defined with `define_parameters()` can be modified in a DSA (or a PSA), this is why most of the costs and probabilities of the model were defined in this step and not hard-written in the transition or state objects. This is also why the `shape` and `scale` parameters were defined separately, and not written in the `define_survival()` call. The analysis is run with `run_dsa()`.

In Figure 4 we represent how parameter variations influenced the total cost for strategy `med`.[9] We see that the discount rate, hospital costs, and the cost and effect of the drug have a more important impact than other parameters. Unsurprisingly parameters used only in the surgery strategy, or parameters unrelated to costs, have no effect here.

In a PSA the values of all parameters are re-sampled from pre-defined distributions and the model is run again with these new parameters as inputs, allowing us to obtain a distribution of model outputs (Critchfield *et al.* 1986).

We define the parameter distributions with `define_psa()`, and optionally their correlation structure with `define_correlation()`.

```
R> def_psa <- define_psa(
R+   age_base          ~ normal(mean = 20, sd = 5),
R+   p_disease_base    ~ binomial(prob = 0.25, size = 500),
R+   p_cured           ~ binomial(prob = 0.001, size = 500),
```

[9] Generated by `plot(res_dsa, result = "cost", strategy = "med")`.

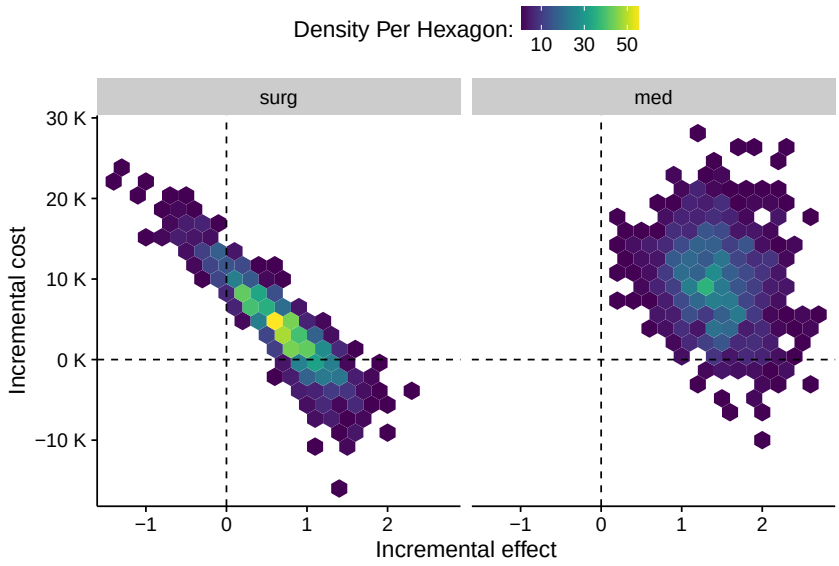


Figure 5: Uncertainty of the incremental cost and effect of strategies on the cost-effectiveness plane, taking the `base` strategy as a reference.

```
R+   med_effect        ~ lognormal(mean = 0.5, sd = 0.1),
R+   shape             ~ normal(mean = 1.5, sd = 0.2),
R+   scale             ~ normal(mean = 5, sd = 1),
R+   cost_med          ~ gamma(mean = 5000, sd = 1000),
R+   cost_surg         ~ gamma(mean = 20000, sd = 3000),
R+   cost_hospit_start ~ gamma(mean = 11000, sd = 2000),
R+   dr                ~ binomial(prob = 0.05, size = 100),
R+   qaly_disease      ~ normal(mean = 0.5, sd = 0.1),
R+   n_years           ~ poisson(mean = 9),
R+
R+   correlation = define_correlation(
R+     shape,    scale,          -0.5,
R+     age_base, p_disease_base,  0.3))

R> res_psa <- run_psa(res_mod, psa = def_psa, N = 1000)

Resampling strategy 'base'...

Resampling strategy 'med'...

Resampling strategy 'surg'...
```

We then run the PSA with `run_psa()`, here for 1,000 re-samplings. The results can be plotted as uncertainty clouds on the cost-effectiveness plane (Figure 5).[10]

[10] A similar plot can be generated with `plot(res_psa, type = "ce")`.

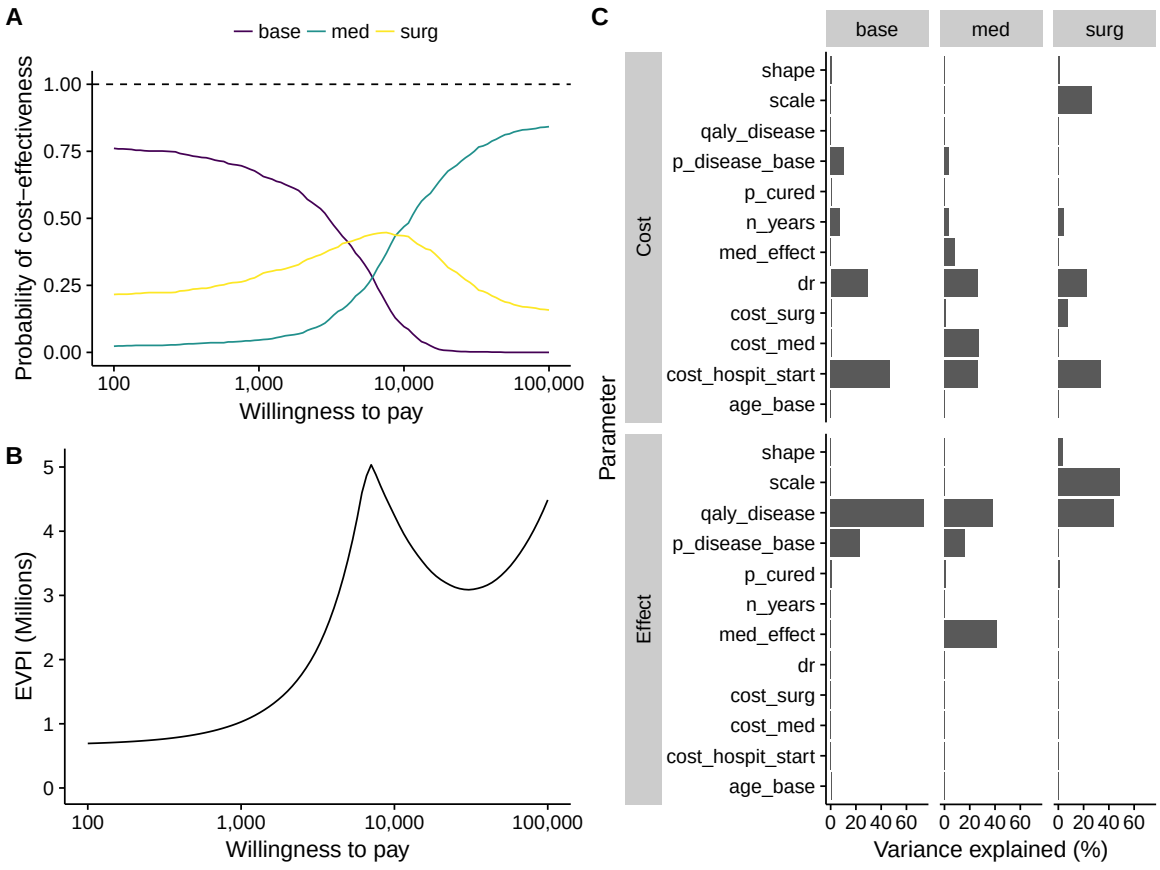


Figure 6: A: Cost-effectiveness acceptability curve; B: Expected value of perfect information; C: Covariance analysis of PSA results.

The probability of a strategy being cost effective can be plotted for various willingness to pay on a cost-effectiveness acceptability curve (Van Hout *et al.* 1994; Fenwick and Byford 2005). In Figure 6A we see that with a threshold ICER below 1,000 the `base` strategy is probably the most cost-effective.[11] Above 50,000 `med` is probably the most cost-effective strategy. Between those 2 values the decision is less clear.

It is possible to compute the expected value of perfect information (EVPI) depending on the willingness to pay (Claxton and Posnett 1996; Felli and Hazen 1997). This is a quantification of the cost of potentially choosing the wrong strategy, and thus conversely the price one is ready to pay to get more information in order to reduce that risk (e.g. conduct more studies). In Figure 6B we see the EVPI peaks between 1,000 and 10,000, where the uncertainty is high.[12] It also increases for higher willingness to pay, because even though the uncertainty is not as high, the costs of a wrong decision become higher.

The EVPI can indicate whether conducting more research is cost-effective. But it does not inform on the value of getting more information on particular parameters (Briggs *et al.* 2006). The expected value of perfect information for parameters (EVPPI) is very similar to the EVPI, but returns values by parameters (Ades *et al.* 2004). Unfortunately its computation is not trivial. PSA results can be exported to compute EVPPI with the Sheffield Accelerated Value of Information SAVI software (Strong *et al.* 2014) with `export_savi()`.

The individual contribution of parameter uncertainty on the overall uncertainty (Briggs *et al.* 2006) is illustrated by Figure 6C.[13] We can see that depending on the strategy different parameters generate the uncertainty on costs and effect. In all cases `dr`, `cost_hospit_start` and `qaly_disease` explain a high part of variability for all strategies. Unsurprisingly the effect of `scale` (the scale of the post-surgery Weibull survival function), `med_effect` and `cost_med` are limited to the `surg` or `med` strategies.

In addition average model values can be computed on the results and presented in a summary similar to the `run_model()` output. Because of non-linearities in Markov models, averages over the PSA output distribution are more accurate than point estimates (Briggs *et al.* 2006). In our case the ICERs changed from 4,559 to 6,322 and 9,033 to 7,095 for the `surg` and `med` strategies respectively.

## 2.9. Heterogeneity analysis

*How does the cost-effectiveness of strategies vary depending on the characteristics of the population?*

If population characteristics are available, model results can be computed on the different subpopulations to study the heterogeneity of the resulting model outputs (Briggs *et al.* 2006). Furthermore average population-level results can be computed from these distributions.

The model we ran in Section 2.7 computed results for a cohort of males aged 20. To assess how population characteristics affect model results we can run a heterogeneity analysis. We use the `update()` function to run the model on a table containing population data.[14]

[11] Generated by `plot(res_psa, type = "ac")`.

[12] Generated by `plot(res_psa, type = "evpi")`.

[13] Generated by `plot(res_psa, type = "cov")`. We could also perform the same analysis on the difference between strategies with the option `diff = TRUE`.

[14] In this example we use a table with population characteristics, here named `tab_pop`, with an optional column `.weights` giving the relative population weight of each strata.

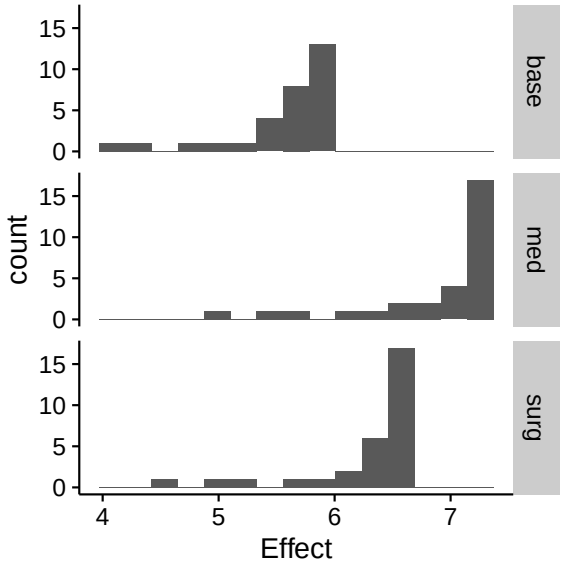


Figure 7: Heterogeneity of strategy effects in the population.

```
R> head(tab_pop)
```

```
  age_base sex_indiv   .weights
1       10       MLE 0.04242889
2       10      FMLE 0.86696571
3       15       MLE 0.69960873
4       15      FMLE 0.51253057
5       20       MLE 0.91723545
6       20      FMLE 0.09685623
```

```
R> pop_mod <- update(res_mod, newdata = tab_pop)
```

```
Updating strategy 'base'...

Updating strategy 'med'...

Updating strategy 'surg'...
```

The summary of the updated model gives the distribution of the values of interest in the population, and the average model values over the entire population. Here the average ICERs in the population are 4,896 and 9,195 for the `surg` and `med` strategies respectively, quite similar to the values computed in Section 2.7 (4,559 and 9,033). We can also plot the distribution of model results (e.g. the intervention effects in Figure 7).[15]

[15] Generated by `plot(pop_mod, result = "effect", bins = 15)`.

## 2.10. Budget impact analysis

*What would be the total cost of a strategy for the health system?*

So far we mostly worked on model results at the scale of the individual (e.g. cost per person). If we want to implement a strategy at a health system level we also need to know the total cost over a given time horizon, to assess whether the strategy is sustainable. This is called a budget impact analysis (BIA). The main differences with the classic model are (1) the patients counts at the model start should reflect the population statistics, and (2) additional patients may enter the model every year (new disease cases).

We use the `init` and `inflow` arguments of `run_model()` to implement BIA, here for the `med` strategy. The inflow of new patients is defined with `define_inflow()`. Inflow counts can depend on *model time* (*state time* dependency is meaningless in this context).

```
R> res_bia <- run_model(
R+   parameters = par_mod,
R+
R+   med = strat_med,
R+
R+   cycles = 10,
R+
R+   cost = cost_total,
R+   effect = qaly,
R+
R+   method = "life-table",
R+
R+   init =   c(
R+     pre   = 25000,
R+     symp  = 5000,
R+     death = 0),
R+   inflow = define_inflow(
R+     pre   = 8000,
R+     symp  = 0,
R+     death = 0))
```

```
med: detected use of 'state_time', expanding states: pre, symp.
```

At the start of the model there is 25,000 patients with asymptomatic shame and 5,000 with a symptomatic form of the disease in the population. Every year 8,000 additional cases of shame are added to the model, starting the disease in the asymptomatic state.

```
R> summary(res_bia)
```

```
1 strategy run for 10 cycles.

Initial state counts:

          N
```

```
pre   25000
symp   5000
death     0

Counting method: 'life-table'.

Values:

    cost_treat cost_hospit cost_total     qaly
med 1942345506  2564136553 3479424515 508363.6
```

The total cost of strategy `med` over a 10-year time horizon will be 3.5 billions.

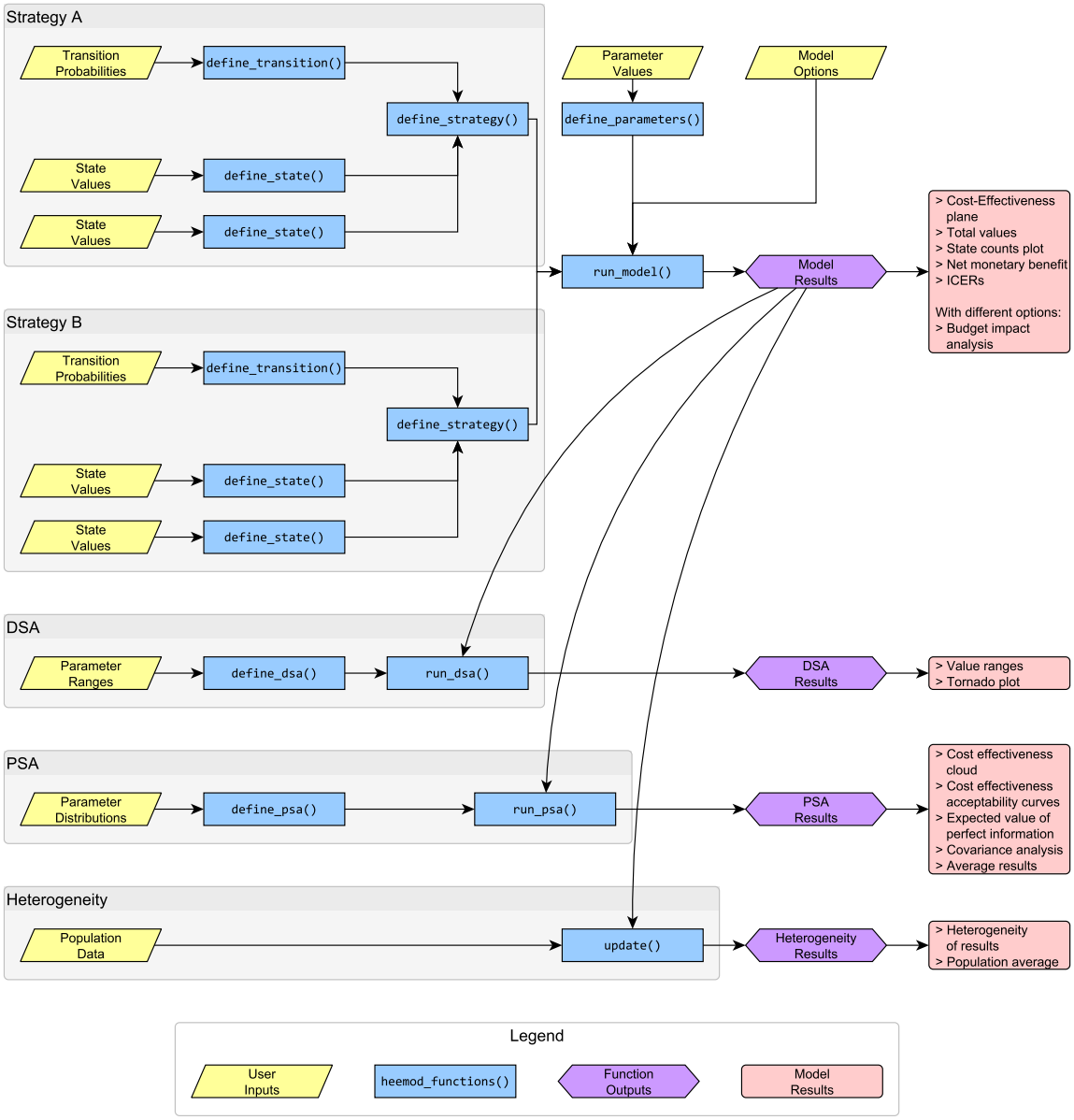


Figure 8: **heemod** package workflow. Drawn with yEd (yWorks 2016).

# 3. Package design and underlying implementation

In this section we explain the ideas underlying the design of the package, the features and extensions that were not presented in the previous example, and the validation process. We then present the package back-end and how most of the features were actually implemented.

## 3.1. Package design and workflow

The package was focused on reproducibility of analyses and ease of use. Both those objectives could be reached by making the functions unambiguous and easily readable by humans. We tried to rely on Hadley Wickham's *tidy manifesto* to design our package in that direction (Wickham 2017).

We divided health economic evaluation modelling into distinct and sequential tasks, and wrote simple *verb* functions corresponding to the most common tasks (e.g. `define_*`, `run_*`). We tried to keep functions as simple as possible: each function should do one thing, each task should have its own function. By simplifying the options we hoped to simplify how the user thinks about modelling, hence making it easier not only to build models, but more importantly for another user to read and understand someone else's model. This last point is of particular importance if we want more research transparency and reproducibility in health economic evaluation studies.

To paraphrase Hal Abelson, we think that models must be written for people to read, and only incidentally for machines to execute.

## 3.2. Other features and extensions

Even though the main focus of **heemod** is to compute Markov models, other methods that model state changes can be used: only the `transition` argument of `define_strategy()` needs to be changed. For example partitioned survival models (Williams *et al.* 2016b) can be computed by passing an object defined by `define_part_surv()` to `transition`. In theory most modelling method that returns state counts over time could be integrated to **heemod**, e.g. dynamic models for infectious diseases (Keeling and Rohani 2011; Snedecor 2012).

For the same objective of reproducibility and ease of use we implemented convenience functions to perform some of the most common calculations needed in health economic evaluation studies (e.g converting incidence rates, odds ratios, or relative risks to transition probabilities with `rate_to_prob()`, `or_to_prob()`, or `rr_to_prob()` respectively). Probabilities and discount rates can be rescaled to fit different time frames (generally the duration of a cycle) with `rescale_prob()` and `rescale_discount_rate()`.

PSA and heterogeneity analyses can become time-consuming since they consist in iteratively re-running the model with new parameter inputs. Because this workload is *embarrassingly parallel* (Herlihy and Shavit 2012), i.e. there is no dependency or need for communication between the parallel tasks, it can easily be run on a cluster relying on the **parallel** (R Core Team 2016) package. This is done by calling the `use_cluster()` function. This function can either take as an argument:

1. A number: a local cluster with the given number of cores will be created.
2. A cluster object defined with the `makeCluster()` function from **parallel**: the user-defined cluster will be used (e.g. to use more complex clusters with non-local hosts).

To facilitate the use of **heemod** by users not familiar with R we developed a **shiny** (Chang *et al.* 2017) graphical user interface. Similarly, for users that require the use of spreadsheet models (such as health regulatory agencies), a model can be specified in spreadsheet files and run by **heemod**. To keep the traceability, transparency and reproducibility advantages provided by written source code it is possible to export models built from these interfaces to R source code files.

These alternative interfaces are needed in a context where (1) Markov models are already widely used and implemented on spreadsheet software, (2) a significant proportion of the modellers are not R users, and (3) health regulatory agencies from several countries require models to be in spreadsheet format. We believe that to gain acceptance in a field such as health economic evaluation where habits are already ingrained one must adapt to existing user requirements, as long as the final outcome is to help develop transparency and reproducibility in the domain.

### 3.3. Validation

We validated the package by reproducing the exact result of 2 analyses described in the reference textbook by Briggs *et al.* (2006): the HIV therapy and the total hip replacement model. In both case we found identical results (total values, patient counts and ICERs).

To ensure the results remain correct when the package is updated or when a dependency is upgraded multiple tests were written with the **testthat** package (Wickham 2011). We verify that functions produce the expected output, that incorrect inputs generate errors, and that results from published models are reproduced. The tests are run as soon as a modification is made to the code: when a change introduces a bug a warning is raised and remains until the issue is fixed.

### 3.4. Back-end

The **heemod** package syntax relies heavily on the non-standard evaluation features offered by **lazyeval** (Wickham 2016). This results in a more legible syntax, similar to the **dplyr** package (Wickham and Francois 2016), by allowing the user to write expressions (e.g. in `define_parameters()`) that will be evaluated later, while keeping namespace collisions in check.[16]

More generally, most of `heemod` core functions rely on the **dplyr** package: the objects, data, and results are stored as `tbl_df` objects, the **dplyr** implementation of data frames. This is another principle of the *tidy manifesto* (Wickham 2017): reuse existing data structures. This allows the use of powerful base functions, efficient computation by limiting copy creation, and iterative model re-computation for PSA or heterogeneity analyses with the `dplyr::do()` function.

Relying on another package instead of writing package-specific functions has the drawback that slightly ill-fitting data structures may sometimes be used. But we think this drawback is outweighed by the multiple advantages of *piggybacking* a widely used package such as `dplyr`. We benefit from the code quality control and the constantly improving features of a popular package, letting us focus our development time on actually implementing Markov models. Even more importantly our internal code is easier to understand by anyone familiar with

[16] A usual pitfall of non-standard evaluation in R.

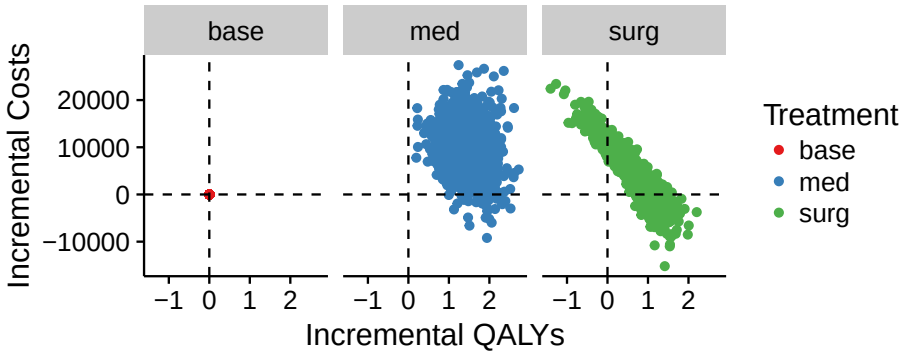


Figure 9: Example of plot customization.

`dplyr`. This last point lowers the bar for including new contributors for bug fixes or new features, and would make a hypothetical maintainer change easier.

The plotting functions rely on the **ggplot2** package (Wickham 2009), and can be easily customized using the `+` operator. The following code is used to produce Figure 9:

```
R> library(ggplot2)
R+
R+ plot(res_psa, type = "ce") +
R+   scale_color_brewer(name = "Treatment", palette = "Set1") +
R+   facet_wrap(~ .strategy_names) +
R+   xlab("Incremental QALYs") + ylab("Incremental Costs") +
R+   geom_hline(yintercept = 0, linetype = "dashed") +
R+   geom_vline(xintercept = 0, linetype = "dashed")
```

The plotting of transition matrices as directed diagrams is performed by the **diagram** package (Soetaert 2014), the covariance analysis of PSA relies on the **mgcv** package (Wood 2011), and the weighted summary of the results is computed with the **Hmisc** package (Harrell and Dupont 2016).

The running time of some functions is not negligible (e.g. `get_who_mr()`). While half a second is not an issue when a function is run only once, it becomes a major hurdle during re-sampling where the function may be called thousands of times. We used the memoisation features of the **memoise** package (Wickham *et al.* 2016) to shorten execution time: when a function is called the result is kept in memory alongside the values of the calling arguments. If the function is called again with the same argument values then the function body is not evaluated, but the memoised result is instantly returned instead. The use of memoisation is particularly efficient in re-sampling because in most cases the values of the arguments of most functions remain identical, resulting in the same outputs.

### 3.5. Parameter correlation in PSA

Correlation of parameters in PSA was implemented with the following steps:

1. A correlation structure is define with `define_correlation()` (see Section 2.8).

2. Values are sampled from a multi-normal distribution having the required correlation structure with **mvnfast** (Fasiolo 2016).

3. The sampled values are then mapped to the target distributions on a quantile by quantile basis.

This approach described in Briggs *et al.* (2006) is an approximation that allows to define correlations between arbitrary distributions. The final Pearson correlation coefficients between the target distributions may differ slightly from the ones initially defined by the user. That issue is mostly true if the target distributions are too dissimilar, e.g. a gamma and a binomial distribution.

### 3.6. Time-dependency implementation

A throughout description of time-dependency in Markov models is given by Hawkins *et al.* (2005), with solutions for the computation of both non-homogeneous and semi-Markov models. These methods were implemented in the **heemod** package. Markov models with *model time* dependency are usually called non-homogeneous Markov models, and models with *state time* dependency are called semi-Markov models.

*Model time* (non-homogeneous Markov models) was implemented by using a 3-dimensional transition matrix $U$. As in the 2-dimensional matrix $T$ described in Section 1 the indices of the first 2 dimensions $i, j$ encode the transition probability between state $i$ and $j$. In addition the third dimension index $k$ corresponds to the number of cycles the model has run so far, so that element $i, j, k$ of matrix $U$ corresponds to the transition probability between state $i$ and $j$ at time $k$. The probability of being in a given state at time $t$ is given by a simple extension of Equation 1:

$$X \times \prod_{k=1}^{t} U_k \tag{2}$$

Where $X$ is a vector[17] giving the probability of being in a given state at the start of the model, $\prod$ stands for matrix multiplication, and $U_k$ is a 2-dimensional slice of the 3-dimensional transition matrix $U$, giving the transition probabilities at time $k$.

*State time* (semi-Markov models) was implemented with the tunnel-state method, described in Hawkins *et al.* (2005). A tunnel state is a state that can be occupied for only 1 cycle, it represents at the same time the health state a person is in and the number of cycles previously spent in this state. A state $A$ with state-time dependency is expanded in $t$ tunnel states $A_1, A_2, \ldots, A_t$ (where $t$ is the total number of cycles). For example consider the following transition matrix:

$$\begin{bmatrix} P(A \to A) = f(s) & P(A \to B) = C \\ P(B \to A) & P(B \to B) \end{bmatrix} \tag{3}$$

Where $P(A \to B)$ is the transition probability between state $A$ and $B$, $s$ the number of cycles spent in state $A$, $f$ an arbitrary function returning a transition probability, and $C$ the probability complement (1 minus the sum of probabilities in a given row). $P(B \to A)$ and $P(B \to B)$ are arbitrary probabilities that do not depend on state time.

[17] Of length equal to the number of states.

The matrix in Equation 3 can be expanded to the following matrix when the model is run for $t$ cycles:

$$\begin{bmatrix} 0 & P(A_1 \to A_2)=f(1) & 0 & \cdots & 0 & 0 & P(A_1 \to B)=C \\ 0 & 0 & P(A_2 \to A_3)=f(2) & \cdots & 0 & 0 & P(A_2 \to B)=C \\ 0 & 0 & 0 & \cdots & 0 & 0 & P(A_3 \to B)=C \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & P(A_{t-1} \to A_t)=f(t-1) & P(A_{t-1} \to B)=C \\ 0 & 0 & 0 & \cdots & 0 & P(A_t \to A_t)=f(t) & P(A_t \to B)=C \\ P(B \to A) & 0 & 0 & \cdots & 0 & 0 & P(B \to B) \end{bmatrix} \tag{4}$$

The semi-Markov model described in Equation 3 is now rearranged as a classic Markov model. It can be noticed that if we were to run this model for more than $t$ cycles then $P(A \to A)$ would remain constant after time $t$, at a value of $f(t)$. This property is useful in situations where $f(s)$ become roughly constant when $s \geq t$: in that case we can stop the state expansion at $t$ tunnel states in order to limit the final matrix size, and thus the computational burden. This approximation is implemented in **heemod** with the `state_cycle_limit` option of `run_model()`.

In practice in **heemod** any state where *state time* dependency is detected is implicitly converted internally to a sequence of tunnel states. Counts and values are internally computed for each tunnel state, and then internally re-aggregated before being returned to the user as a single state. The transformation to a sequence of tunnel states is thus invisible for the user, except for a message informing of the implicit state expansion.

## 3.7. Implementing budget impact analysis

The main technical difficulty to implement budget impact analyses is that new individuals may enter the model at any point in time. The classic Markov model computation described in Equation 1 cannot be used any more. Instead the probability of being in a given state at time $k$ is given by the following sequence:

$$\begin{aligned} a_0 &= Y \\ a_k &= a_{k-1} \times T + Z \end{aligned} \tag{5}$$

Where $Y$ is a vector of the number of individuals in a given state at the start of the model, $T$ a 2-dimensional transition matrix, and $Z$ a vector[18] giving the number of new individuals entering the model at each new cycle.

Equation 5 can be adapted to allow (1) for *model time* dependency of transition probabilities as described in Equation 2 and (2) for time-dependency of the number of individual entering the model at each new cycle, in this way:

$$\begin{aligned} a_0 &= Y \\ a_k &= a_{k-1} \times U_k + Z_k \end{aligned} \tag{6}$$

Where $U_k$ is a 2-dimensional slice of the 3-dimensional transition matrix $U$, and $Z_k$ a vector giving the number of new individuals entering the model at cycle $k$.

Finally, *state time* dependency by tunnel state expansion can be integrated without any change to Equation 6.

[18] Of length equal to the number of states.

## Acknowledgements

Thanks to Matthew Wiener, Zdenek Kabat and Vojtech Filipec for their contributions to this package, to Marina Filipović-Pierucci for her help with mathematical notation and for proof-reading, and to Rana Maroun, Chloé Le Cossec and Guillaume Pressiat for proof-reading this paper.

## References

Adams D (1979). *The Hitchhiker's Guide to the Galaxy*. Del Rey.

Ades AE, Lu G, Claxton K (2004). "Expected Value of Sample Information Calculations in Medical Decision Modeling." *Medical Decision Making*, **24**(2), 207–227. doi:10.1177/0272989X04263162. PMID: 15090106, URL http://dx.doi.org/10.1177/0272989X04263162.

Barendregt JJ (2009). "The Half-Cycle Correction: Banish Rather Than Explain It." *Medical Decision Making*, **29**(4), 500–502. doi:10.1177/0272989X09340585. URL http://dx.doi.org/10.1177/0272989X09340585.

Beck J, Pauker SG (1983). "The Markov Process in Medical Prognosis." *Medical Decision Making*, **3**(4), 419–458. doi:10.1177/0272989X8300300403. URL http://dx.doi.org/10.1177/0272989X8300300403.

Briggs A, Sculpher M (1998). "An Introduction to Markov Modelling for Economic Evaluation." *PharmacoEconomics*, **13**(4), 397–409. doi:10.2165/00019053-199813040-00003. URL http://dx.doi.org/10.2165/00019053-199813040-00003.

Briggs A, Sculpher M, Buxton M (1994). "Uncertainty in the Economic Evaluation of Health Care Technologies: The Role of Sensitivity Analysis." *Health Economics*, **3**(2), 95–104. ISSN 1099-1050. doi:10.1002/hec.4730030206. URL http://dx.doi.org/10.1002/hec.4730030206.

Briggs A, Sculpher M, Claxton K (2006). *Decision Modelling for Health Economic Evaluation (Handbooks in Health Economic Evaluation)*. Oxford University Press, USA. ISBN 9780198526629.

Chang W, Cheng J, Allaire J, Xie Y, McPherson J (2017). ***shiny**: Web Application Framework for R*. R package version 1.0.0, URL https://CRAN.R-project.org/package=shiny.

Claxton K, Martin S, Soares M, Rice N, Spackman E, Hinde S, Devlin N, Smith PC, Sculpher M (2015). "Methods for the Estimation of the National Institute for Health and Care Excellence Cost-Effectiveness Threshold." *Health Technology Assessment*, **19**(14), 1–504. doi:10.3310/hta19140. URL https://doi.org/10.3310%2Fhta19140.

Claxton K, Posnett J (1996). "An Economic Approach to Clinical Trial Design and Research Priority-Setting." *Health Economics*, **5**(6), 513–524. ISSN 1099-1050. doi:10.1002/(SICI)1099-1050(199611)5:6<513::AID-HEC237>3.0.CO;2-9. URL http://dx.doi.org/10.1002/(SICI)1099-1050(199611)5:6<513::AID-HEC237>3.0.CO;2-9.

Critchfield GC, Willard KE, Connelly DP (1986). "Probabilistic Sensitivity Analysis Methods for General Decision Models." *Computers and Biomedical Research*, **19**(3), 254 – 265. ISSN 0010-4809. doi:http://dx.doi.org/10.1016/0010-4809(86)90020-0. URL http://www.sciencedirect.com/science/article/pii/0010480986900200.

Drummond MF, Sculpher MJ, Claxton K, Stoddart GL, Torrance GW (2005). *Methods for the Economic Evaluation of Health Care Programmes, Fourth Edition*. Oxford University Press. ISBN 9780198529453.

Eddy DM, Hollingworth W, Caro JJ, Tsevat J, McDonald KM, Wong JB (2012). "Model Transparency and Validation." *Medical Decision Making*, **32**(5), 733–743. doi:10.1177/0272989X12454579. URL http://dx.doi.org/10.1177/0272989X12454579.

Fasiolo M (2016). *An introduction to **mvnfast**. R package version 0.1.6*. URL https://CRAN.R-project.org/package=mvnfast.

Felli JC, Hazen GB (1997). "Sensitivity Analysis And Expected Value Of Perfect Information." *Medical Decision Making*, **18**, 95–109.

Fenwick E, Byford S (2005). "A Guide to Cost-Effectiveness Acceptability Curves." *The British Journal of Psychiatry*, **187**(2), 106–108. ISSN 0007-1250. doi:10.1192/bjp.187.2.106. URL http://bjp.rcpsych.org/content/187/2/106.

Filipovic-Pierucci A (2017). ***rgho**: Access WHO Global Health Observatory Data from R*. R package version 1.0.1, URL https://CRAN.R-project.org/package=rgho.

Harrell FE, Dupont C (2016). ***Hmisc**: Harrell Miscellaneous*. R package version 4.0-2, URL https://CRAN.R-project.org/package=Hmisc.

Hawkins N, Sculpher M, Epstein D (2005). "Cost-Effectiveness Analysis of Treatments for Chronic Disease: Using R to Incorporate Time Dependency of Treatment Response." *Medical Decision Making*, **25**(5), 511–519. doi:10.1177/0272989X05280562. URL http://dx.doi.org/10.1177/0272989X05280562.

Herlihy M, Shavit N (2012). *The Art of Multiprocessor Programming, Revised Reprint*. 1st edition. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. ISBN 9780123973375, 9780123977953.

Husereau D, Drummond M, Petrou S, Carswell C, Moher D, Greenberg D, Augustovski F, Briggs AH, Mauskopf J, Loder E (2013). "Consolidated Health Economic Evaluation Reporting Standards (CHEERS) statement." *BMJ*, **346**. doi:10.1136/bmj.f1049. URL http://www.bmj.com/content/346/bmj.f1049.

Jackson C (2016). "**flexsurv**: A Platform for Parametric Survival Modeling in R." *Journal of Statistical Software*, **70**(8), 1–33. doi:10.18637/jss.v070.i08.

Keeling MJ, Rohani P (2011). *Modeling Infectious Diseases in Humans and Animals*. Princeton University Press. ISBN 978-0691116174.

Microsoft (2016). *Microsoft Excel*. URL http://office.microsoft.com/en-us/excel/.

R Core Team (2016). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Russell L, Gold M, Siegel J, Daniels N, Weinstein M (1996). "The Role of Cost-Effectiveness Analysis in Health and Medicine." *JAMA*, **276**(14), 1172–1177. doi:10.1001/jama.1996.03540140060028. URL http://dx.doi.org/10.1001/jama.1996.03540140060028.

Snedecor SJ (2012). "Understanding and Use of Dynamic Models in Health Economic Analyses." *ISPOR Connections*. URL https://www.ispor.org/news/articles/Nov-Dec12/understanding-use-dynamic-models.asp.

Soetaert K (2014). ***diagram**: Functions for visualising simple graphs (networks), plotting flow diagrams*. R package version 1.6.3, URL https://CRAN.R-project.org/package=diagram.

Sonnenberg FA, Beck JR (1993). "Markov Models in Medical Decision Making." *Medical Decision Making*, **13**(4), 322–338. doi:10.1177/0272989X9301300409. PMID: 8246705, URL http://dx.doi.org/10.1177/0272989X9301300409.

Stinnett AA, Mullahy J (1998). "Net Health Benefits: A New Framework for the Analysis of Uncertainty in Cost-Effectiveness Analysis." *NBER Technical Working Papers 0227*, National Bureau of Economic Research, Inc. URL https://ideas.repec.org/p/nbr/nberte/0227.html.

Strong M, Oakley JE, Brennan A (2014). "Estimating Multiparameter Partial Expected Value of Perfect Information from a Probabilistic Sensitivity Analysis Sample." *Medical Decision Making*, **34**(3), 311–326. doi:10.1177/0272989X13505910. URL http://dx.doi.org/10.1177/0272989X13505910.

Torrance GW, Feeny D (1989). "Utilities and quality-adjusted life years." *International journal of technology assessment in health care*, **5**(04), 559–575.

Torvalds L, Hamano J (2017). *git - The Stupid Content Tracker*. URL https://git-scm.com/.

Van Hout BA, Al MJ, Gordon GS, Rutten FFH (1994). "Costs, effects and C/E-ratios alongside a clinical trial." *Health Economics*, **3**(5), 309–319. ISSN 1099-1050. doi:10.1002/hec.4730030505. URL http://dx.doi.org/10.1002/hec.4730030505.

Wickham H (2009). ***ggplot2**: Elegant Graphics for Data Analysis*. Springer-Verlag New York. ISBN 978-0-387-98140-6. URL http://ggplot2.org.

Wickham H (2011). "**testthat**: Get Started with Testing." *The R Journal*, **3**, 5–10. URL http://journal.r-project.org/archive/2011-1/RJournal_2011-1_Wickham.pdf.

Wickham H (2016). ***lazyeval**: Lazy (Non-Standard) Evaluation*. R package version 0.2.0, URL https://CRAN.R-project.org/package=lazyeval.

Wickham H (2017). ***tidyverse**: Easily Install and Load 'Tidyverse' Packages*. R package version 1.1.1, URL https://CRAN.R-project.org/package=tidyverse.

Wickham H, Francois R (2016). ***dplyr**: A Grammar of Data Manipulation*. R package version 0.5.0, URL https://CRAN.R-project.org/package=dplyr.

Wickham H, Hester J, Müller K (2016). ***memoise**: Memoisation of Functions*. R package version 1.0.0, URL https://CRAN.R-project.org/package=memoise.

Williams C, Lewsey JD, Briggs AH, Mackay DF (2016a). "Cost-effectiveness analysis in R using a multi-state modelling survival analysis framework: a tutorial." *Medical Decision Making*. doi:10.1177/0272989X16651869. URL http://eprints.gla.ac.uk/116523/.

Williams C, Lewsey JD, Mackay DF, Briggs AH (2016b). "Estimation of Survival Probabilities for Use in Cost-Effectiveness Analyses." *Medical Decision Making*, **0**(0), 0272989X16670617. doi:10.1177/0272989X16670617. PMID: 27698003, URL http://dx.doi.org/10.1177/0272989X16670617.

Wood SN (2011). "Fast Stable Restricted Maximum Likelihood and Marginal Likelihood Estimation of Semiparametric Generalized Linear Models." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **73**(1), 3–36. ISSN 1467-9868. doi:10.1111/j.1467-9868.2010.00749.x. URL http://dx.doi.org/10.1111/j.1467-9868.2010.00749.x.

yWorks (2016). *yEd Graph Editor*. URL http://www.yworks.com/products/yed.

## Affiliation:

Antoine Filipović-Pierucci
Unité de recherche clinique en économie de la santé d'Île-de-France
Hôtel-Dieu
Assistance publique-Hôpitaux de Paris
Parvis de Notre-Dame, Paris, France
E-mail: antoine.filipovic-pierucci@urc-eco.fr
URL: https://pierucci.org