APPLIED COST-EFFECTIVENESS MODELING WITH R

SMDM
October 13, 2020

Devin Incerti
Jeroen Jansen

Learning objectives

- Understand how R can be used to perform model-based cost-effectiveness analysis with existing packages;
- Develop own models in R by modifying existing code for commonly used model types;
- Understand how using R can improve reproducibility and transparency of model-based cost-effectiveness analysis

Agenda

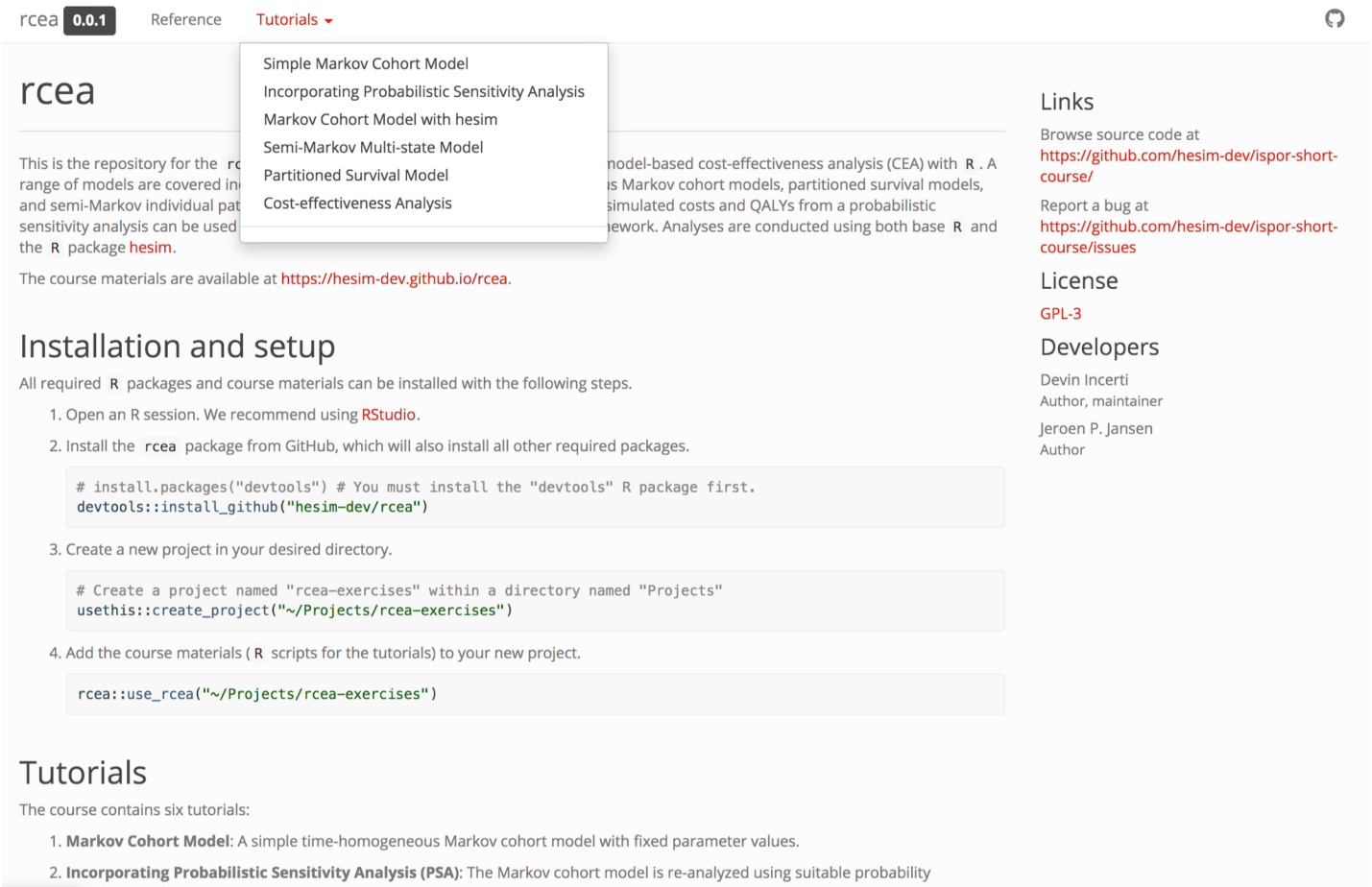
- Introduction
- Basic model taxonomy
- Simple Markov cohort model
- Semi-Markov multi-state model
- Cost-effectiveness analysis
- Summary

Structure

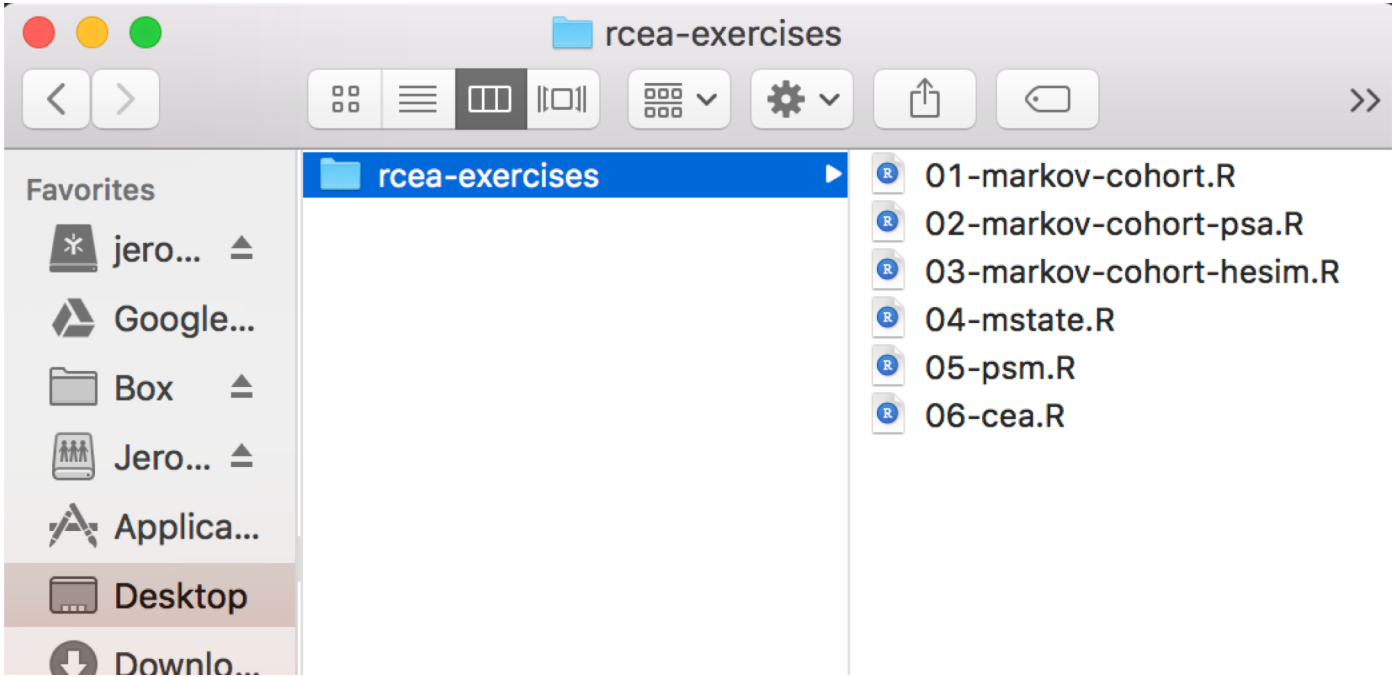
- Presentation
- Exercises using R

Online tutorial

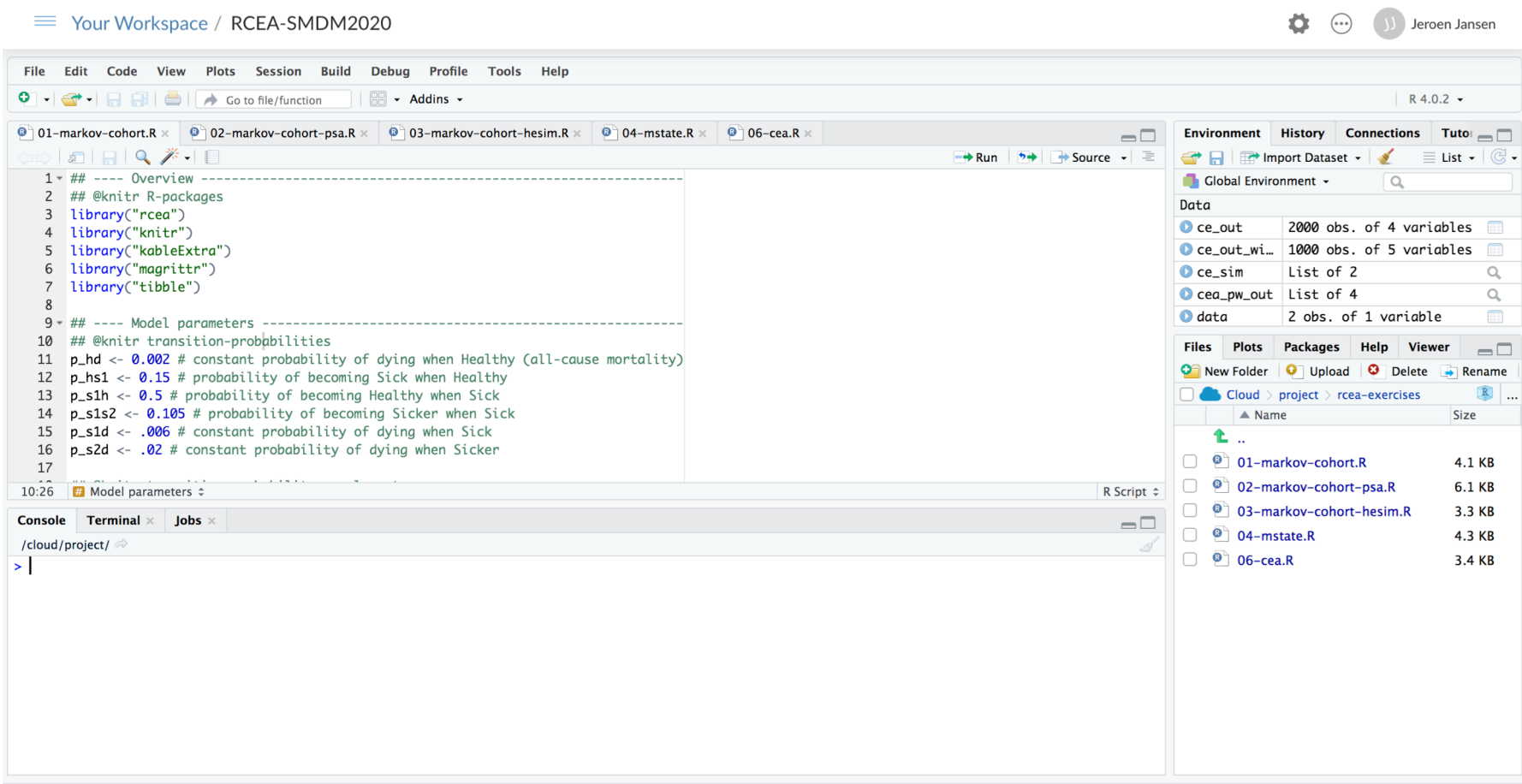
https://hesim-dev.github.io/rcea/



R scrips for exercises



RStudio Cloud

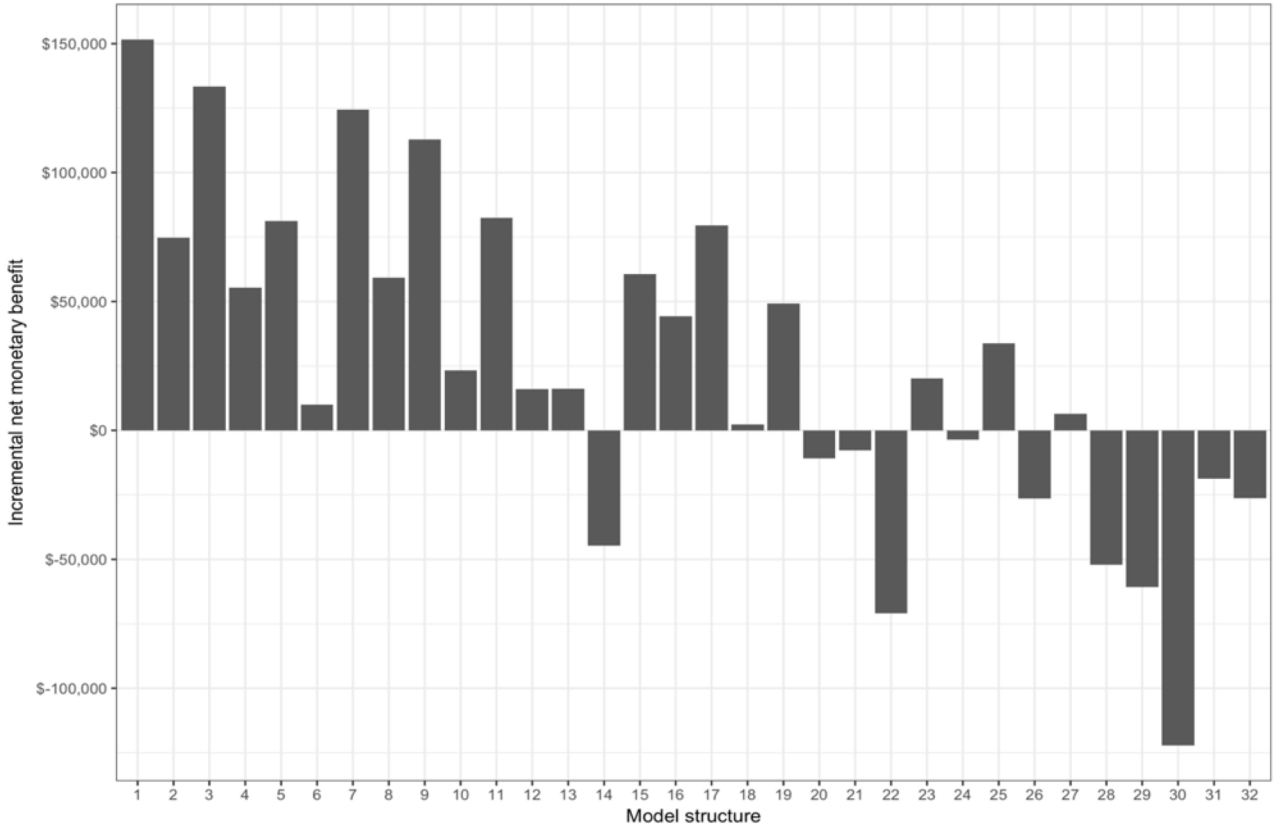


Introduction

Criteria that economic models should strive to meet

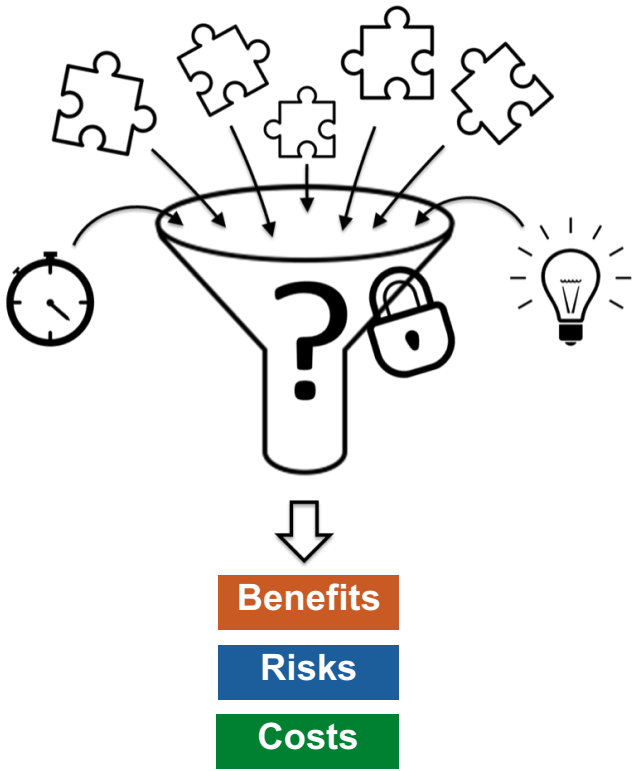
- Clinical realism
  - A model should reflect the state of evidence, the current understanding of the disease, and be accepted by clinical experts.
- Quantifying decision uncertainty
  - A model should be capable of quantifying decision uncertainty and informing prioritization of future research.
- Transparency and reproducibility
  - Resources should exist so that a model can be completely understood, reproduced, and pressure tested.
- Reusability and adaptability
  - It should be possible to easily update a model to reflect new clinical evidence or adapt it for a new market, indication, or intervention.

Structural uncertainty



Plugging in model input parameter estimates

Transparency & reproducibility



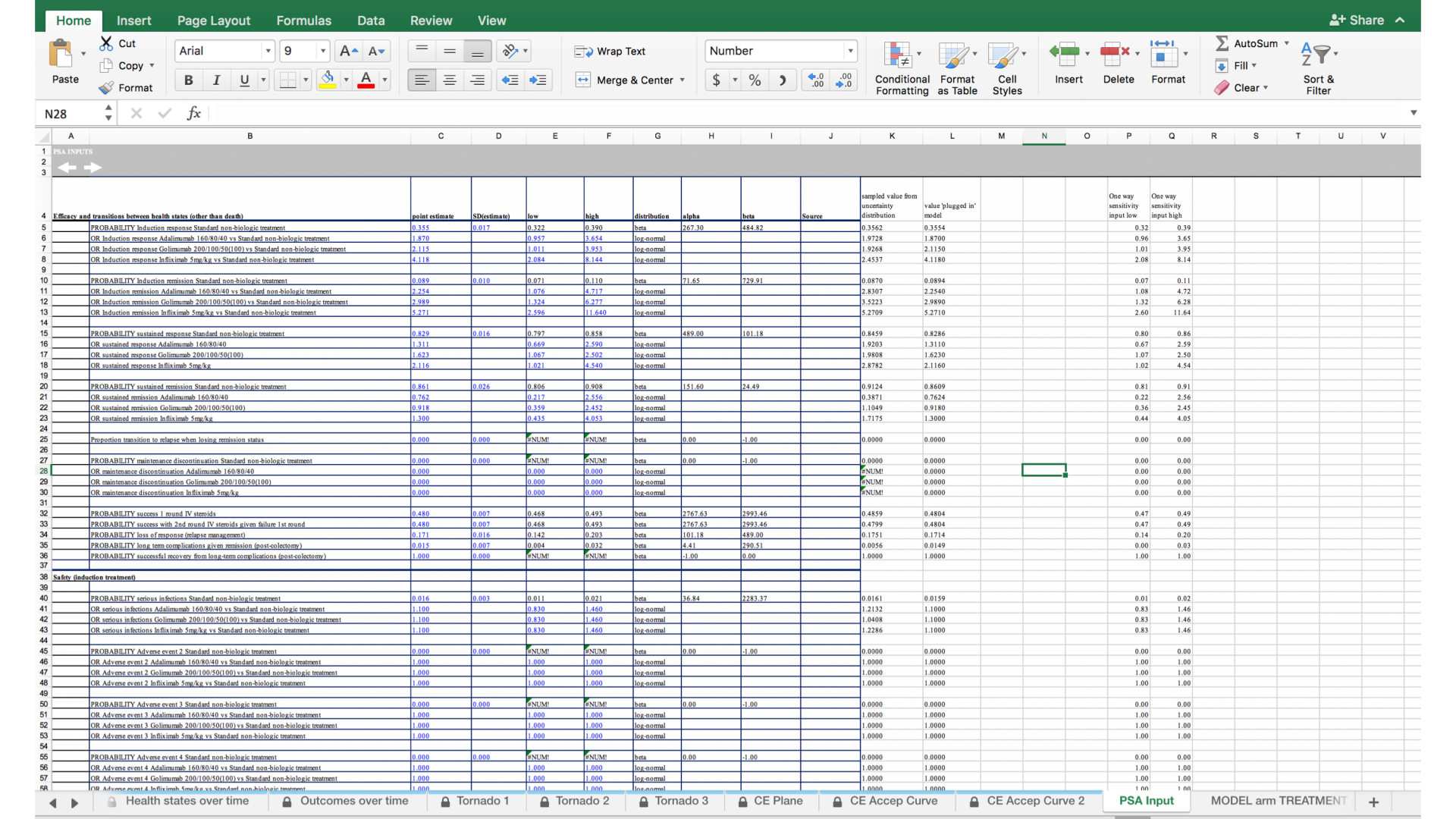
What do we mean with model transparency?

- Concept, math
- Face validity
- Implementation/programming
- Open-source, open-access

- *Familiarity with software?*

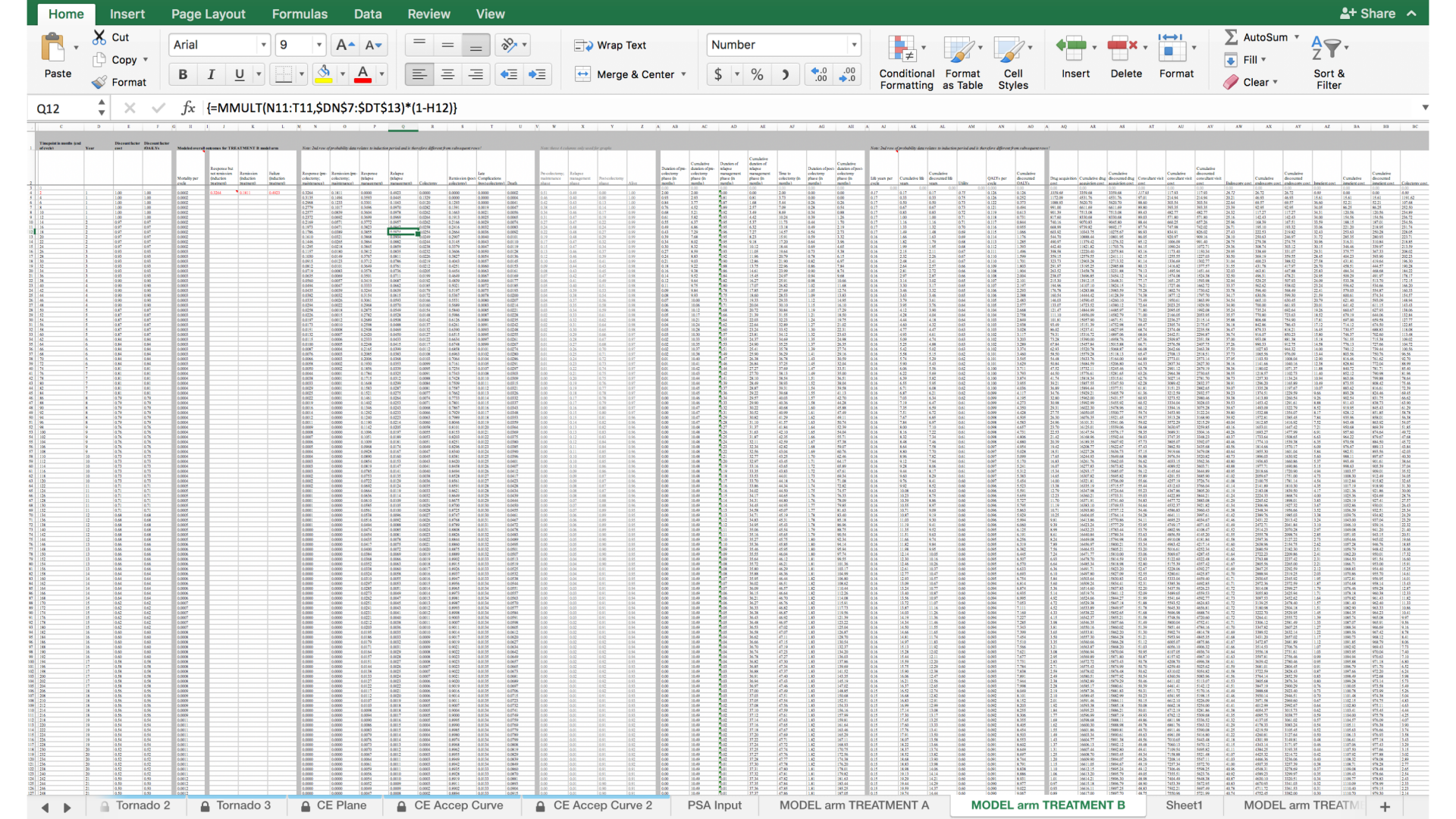
Modeling in Excel

- Excel has been dominant software platform used by HE modelers, especially for HTA submissions
- Reasons are not surprising
  - Practically everyone with a computer has access to Excel
  - Does not require that you learn a new programming language
- Many consider its “transparency” to be an attribute
- With models in Excel, you can follow calculations that are being performed in every single cell of every single worksheet

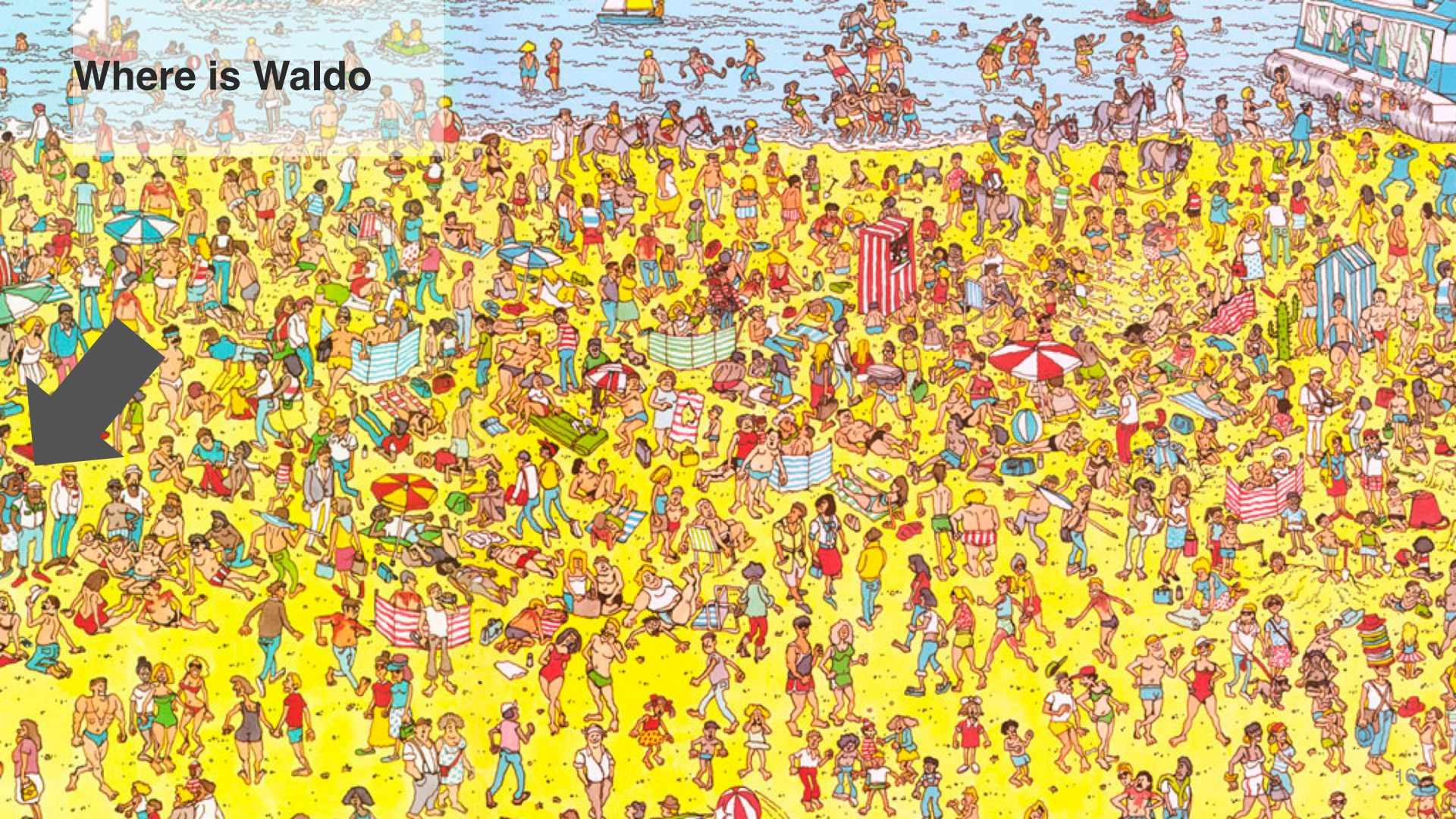


PSA INPUTS

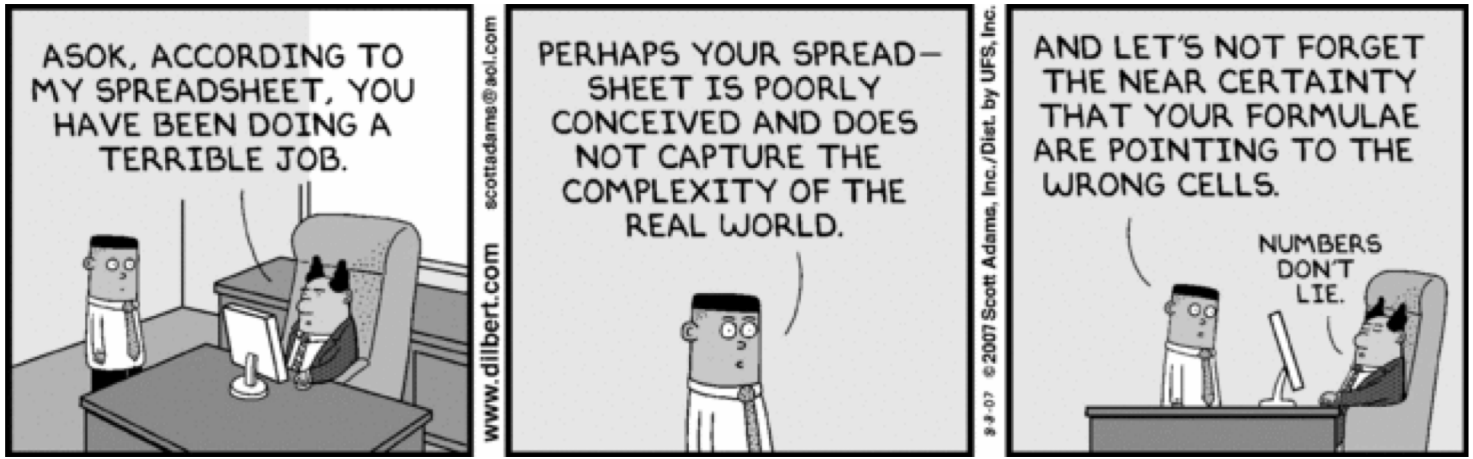
Efficacy and transitions between health states (other than death)	point estimate	SD(estimate)	low	high	distribution	alpha	beta	Source	sampled value from uncertainty distribution	value 'plugged in' model	One way sensitivity input low	One way sensitivity input high
PROBABILITY Induction response Standard non-biologic treatment	0.355	0.017	0.322	0.390	beta	267.30	484.82		0.3562	0.3554	0.32	0.39
OR Induction response Adalimumab 160/80/40 vs Standard non-biologic treatment	1.870		0.957	3.654	log-normal				1.9728	1.8700	0.96	3.65
OR Induction response Golimumab 200/100/50(100) vs Standard non-biologic treatment	2.115		1.011	3.953	log-normal				1.9268	2.1150	1.01	3.95
OR Induction response Infliximab 5mg/kg vs Standard non-biologic treatment	4.118		2.084	8.144	log-normal				2.4537	4.1180	2.08	8.14
PROBABILITY Induction remission Standard non-biologic treatment	0.089	0.010	0.071	0.110	beta	71.65	729.91		0.0870	0.0894	0.07	0.11
OR Induction remission Adalimumab 160/80/40 vs Standard non-biologic treatment	2.254		1.076	4.717	log-normal				2.8307	2.2540	1.08	4.72
OR Induction remission Golimumab 200/100/50(100) vs Standard non-biologic treatment	2.989		1.324	6.277	log-normal				3.5223	2.9890	1.32	6.28
OR Induction remission Infliximab 5mg/kg vs Standard non-biologic treatment	5.271		2.596	11.640	log-normal				5.2709	5.2710	2.60	11.64
PROBABILITY sustained response Standard non-biologic treatment	0.829	0.016	0.797	0.858	beta	489.00	101.18		0.8459	0.8286	0.80	0.86
OR sustained response Adalimumab 160/80/40	1.311		0.669	2.590	log-normal				1.9203	1.3110	0.67	2.59
OR sustained response Golimumab 200/100/50(100)	1.623		1.067	2.502	log-normal				1.9808	1.6230	1.07	2.50
OR sustained response Infliximab 5mg/kg	2.116		1.021	4.540	log-normal				2.8782	2.1160	1.02	4.54
PROBABILITY sustained remission Standard non-biologic treatment	0.861	0.026	0.806	0.908	beta	151.60	24.49		0.9124	0.8609	0.81	0.91
OR sustained remission Adalimumab 160/80/40	0.762		0.217	2.556	log-normal				0.3871	0.7624	0.22	2.56
OR sustained remission Golimumab 200/100/50(100)	0.918		0.359	2.452	log-normal				1.1049	0.9180	0.36	2.45
OR sustained remission Infliximab 5mg/kg	1.300		0.435	4.053	log-normal				1.7175	1.3000	0.44	4.05
Proportion transition to relapse when losing remission status	0.000	0.000	#NUM!	#NUM!	beta	0.00	-1.00		0.0000	0.0000	0.00	0.00
PROBABILITY maintenance discontinuation Standard non-biologic treatment	0.000	0.000	#NUM!	#NUM!	beta	0.00	-1.00		0.0000	0.0000	0.00	0.00
OR maintenance discontinuation Adalimumab 160/80/40	0.000		0.000	0.000	log-normal				#NUM!	0.0000	0.00	0.00
OR maintenance discontinuation Golimumab 200/100/50(100)	0.000		0.000	0.000	log-normal				#NUM!	0.0000	0.00	0.00
OR maintenance discontinuation Infliximab 5mg/kg	0.000		0.000	0.000	log-normal				#NUM!	0.0000	0.00	0.00
PROBABILITY success 1 round IV steroids	0.480	0.007	0.468	0.493	beta	2767.63	2993.46		0.4859	0.4804	0.47	0.49
PROBABILITY success with 2nd round IV steroids given failure 1st round	0.480	0.007	0.468	0.493	beta	2767.63	2993.46		0.4799	0.4804	0.47	0.49
PROBABILITY loss of response (relapse management)	0.171	0.016	0.142	0.203	beta	101.18	489.00		0.1751	0.1714	0.14	0.20
PROBABILITY long term complications given remission (post-colectomy)	0.015	0.007	0.004	0.032	beta	4.41	290.51		0.0056	0.0149	0.00	0.03
PROBABILITY successful recovery from long-term complications (post-colectomy)	1.000	0.000	#NUM!	#NUM!	beta	-1.00	0.00		1.0000	1.0000	1.00	1.00
Safety (induction treatment)												
PROBABILITY serious infections Standard non-biologic treatment	0.016	0.003	0.011	0.021	beta	36.84	2283.37		0.0161	0.0159	0.01	0.02
OR serious infections Adalimumab 160/80/40 vs Standard non-biologic treatment	1.100		0.830	1.460	log-normal				1.2132	1.1000	0.83	1.46
OR serious infections Golimumab 200/100/50(100) vs Standard non-biologic treatment	1.100		0.830	1.460	log-normal				1.0408	1.1000	0.83	1.46
OR serious infections Infliximab 5mg/kg vs Standard non-biologic treatment	1.100		0.830	1.460	log-normal				1.2286	1.1000	0.83	1.46
PROBABILITY Adverse event 2 Standard non-biologic treatment	0.000	0.000	#NUM!	#NUM!	beta	0.00	-1.00		0.0000	0.0000	0.00	0.00
OR Adverse event 2 Adalimumab 160/80/40 vs Standard non-biologic treatment	1.000		1.000	1.000	log-normal				1.0000	1.0000	1.00	1.00
OR Adverse event 2 Golimumab 200/100/50(100) vs Standard non-biologic treatment	1.000		1.000	1.000	log-normal				1.0000	1.0000	1.00	1.00
OR Adverse event 2 Infliximab 5mg/kg vs Standard non-biologic treatment	1.000		1.000	1.000	log-normal				1.0000	1.0000	1.00	1.00
PROBABILITY Adverse event 3 Standard non-biologic treatment	0.000	0.000	#NUM!	#NUM!	beta	0.00	-1.00		0.0000	0.0000	0.00	0.00
OR Adverse event 3 Adalimumab 160/80/40 vs Standard non-biologic treatment	1.000		1.000	1.000	log-normal				1.0000	1.0000	1.00	1.00
OR Adverse event 3 Golimumab 200/100/50(100) vs Standard non-biologic treatment	1.000		1.000	1.000	log-normal				1.0000	1.0000	1.00	1.00
OR Adverse event 3 Infliximab 5mg/kg vs Standard non-biologic treatment	1.000		1.000	1.000	log-normal				1.0000	1.0000	1.00	1.00
PROBABILITY Adverse event 4 Standard non-biologic treatment	0.000	0.000	#NUM!	#NUM!	beta	0.00	-1.00		0.0000	0.0000	0.00	0.00
OR Adverse event 4 Adalimumab 160/80/40 vs Standard non-biologic treatment	1.000		1.000	1.000	log-normal				1.0000	1.0000	1.00	1.00
OR Adverse event 4 Golimumab 200/100/50(100) vs Standard non-biologic treatment	1.000		1.000	1.000	log-normal				1.0000	1.0000	1.00	1.00
OR Adverse event 4 Infliximab 5mg/kg vs Standard non-biologic treatment	1.000		1.000	1.000	log-normal				1.0000	1.0000	1.00	1.00


Home
Insert
Page Layout
Formulas
Data
Review
View
Share
Cut
Copy
Format
Paste
Arial
9
Wrap Text
Number
Merge & Center
Conditional Formatting
Format as Table
Cell Styles
Insert
Delete
Format
AutoSum
Fill
Clear
Sort & Filter
Q12
{=MMULT(N11:T11,$DN$7:$DT$13)*(1-H12)}
Tornado 2
Tornado 3
CE Plane
CE Accep Curve
CE Accep Curve 2
PSA Input
MODEL arm TREATMENT A
MODEL arm TREATMENT B
Sheet1
MODEL arm TREATM

Where is Waldo



Time to change?



Alternative

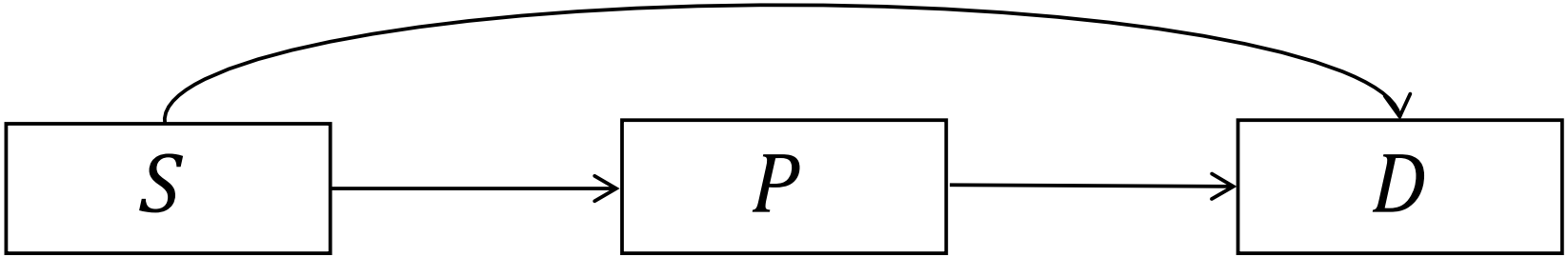
`BCEA` `heemod` `hesim` …

What is R

- Statistical programing language and environment for statistical computing
- Free to use (open source, user developed packages that are transparent)
- Very good for data management, statistical analysis, and visualization
- Scripts contain all steps to perform an analysis
- CEA models can be coded from ‘scratch’ using base R or via convenient and improving packages

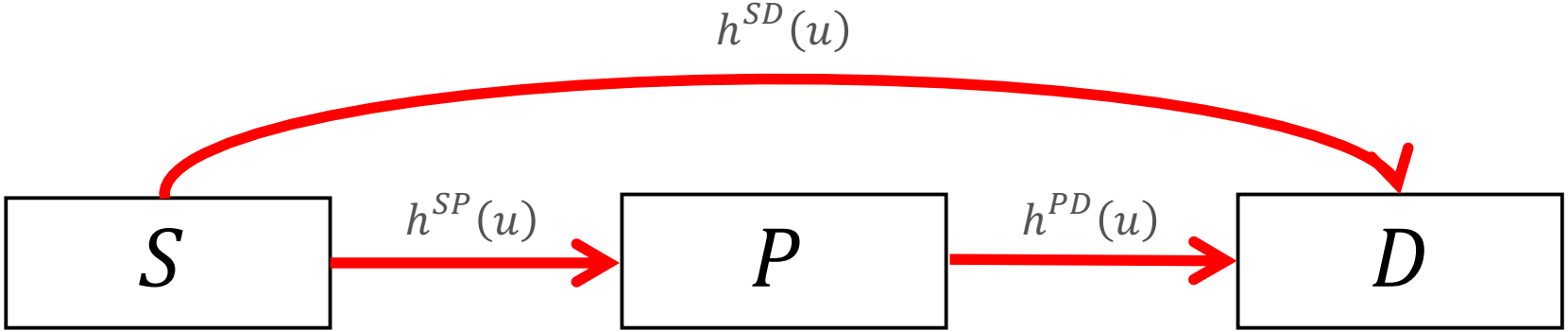
Basic model taxonomy

Health states describing course of disease over time



Markov model

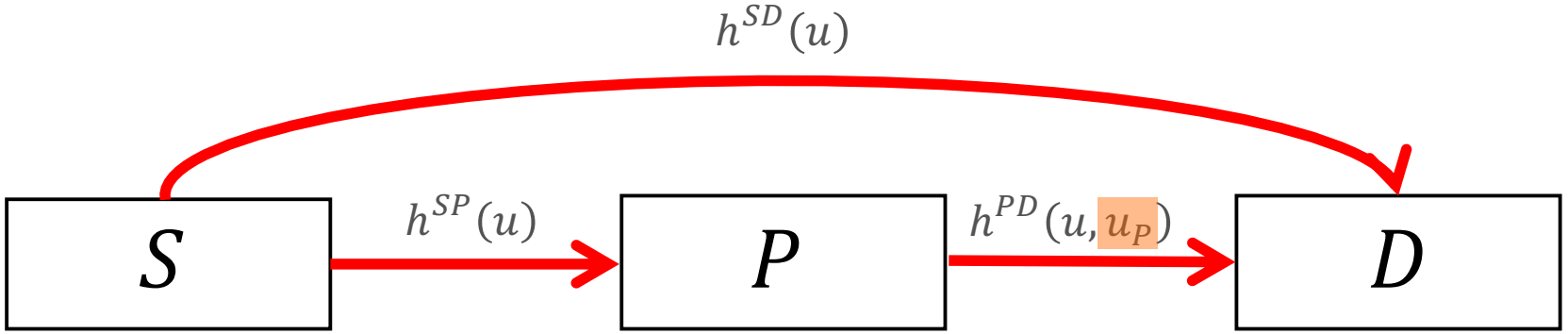
Clock forward



transition rates depend only on time in model

Semi-Markov model

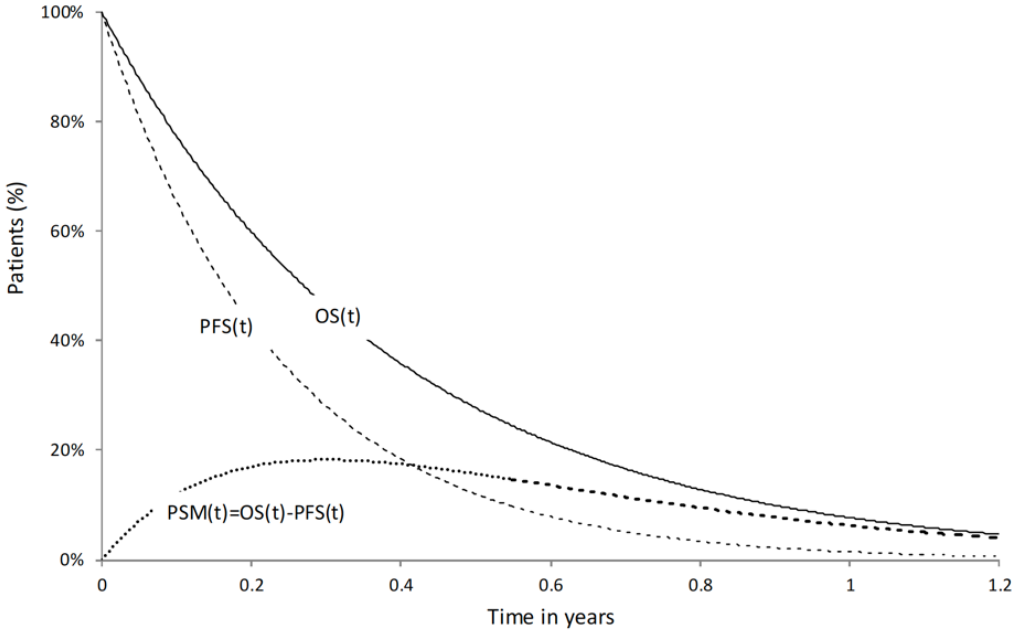
Clock reset

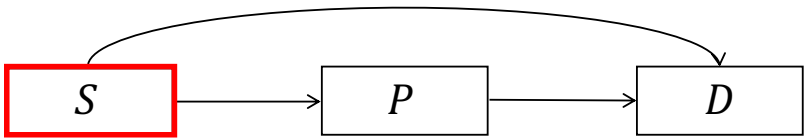


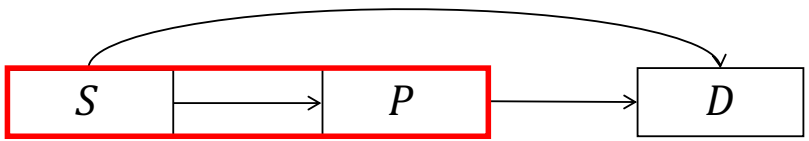
transition rates depend on time in model

some transitions depend on time in an intermediate health state

Partitioned survival model







Source: Woods B, Sideris E, Palmer S, Latimer N, Soares M. NICE DSU Technical Support Document 19. Partitioned Survival Analysis for Decision Modelling in Health Care: A Critical Review. 2017 [Available from http://www.nicedsu.org.uk]

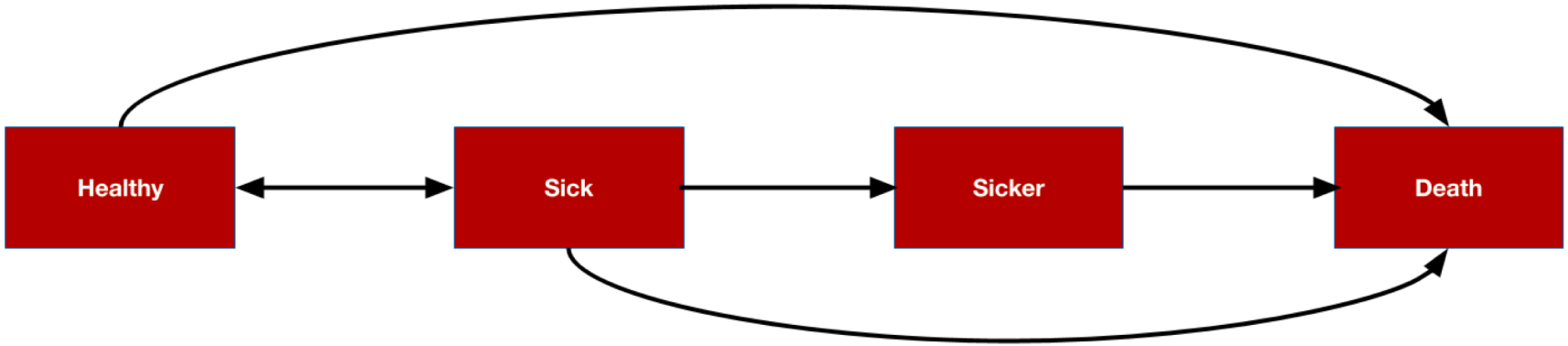
Overview

<table>
<tr><th colspan="3"></th><th>discrete time</th><th>continuous time</th></tr>
<tr><td rowspan="3">state transition models</td><td rowspan="2">Markov
(“Clock forward”)</td><td>Cohort</td><td>cohort discrete time state transition models (cDTSTM)
• time-homogeneous Markov models
• time-inhomogeneous Markov models</td><td>cohort continuous time state transition models</td></tr>
<tr><td>Individual-level</td><td>individual-level discrete time state transition models (iDTSTM)</td><td>individual-level continuous time state transition models (iCTSTM)</td></tr>
<tr><td>Semi-Markov
(“Clock reset”)*</td><td>Individual-level</td><td>iDTSTM</td><td>iCTSTM</td></tr>
<tr><td colspan="5">n-state partitioned survival models**</td></tr>
</table>

**tunnel states can be used in a cohort model to approximate a semi-Markov process*
***Online tutorial provides instructions for partitioned survival models*

Simple Markov cohort model

Model



Model

- Annual transition probabilities with SOC

	Healthy	Sick	Sicker	Death
Healthy	0.848	0.15	0	0.002
Sick	0.5	0.389	0.105	0.006
Sicker	0	0	0.98	0.02
Death	0	0	0	1.000

- Relative risk of progression to a worse health state with new intervention is 0.8

- Drug costs

	New	SOC
Drug costs	12000	2000

- Other annual costs and utility

	Healthy	Sick	Sicker	Death
Direct medical	2000	4000	15000	0
Utility	1	0.75	0.5	0

- Annual discount rates of 3% for costs and 3% for QALYs

SOC

	Healthy	Sick	Sicker	Death
Healthy	0.848	0.15	0	0.002
Sick	0.5	0.389	0.105	0.006
Sicker	0	0	0.98	0.02
Death	0	0	0	1

	Healthy	Sick	Sicker	Death
Utility	1	0.75	0.5	0
Direct medical	2000	4000	15000	0
Drug costs	2000			

Cycle	Healthy	Sick	Sicker	Death	discounted QALYs	discounted medical costs	discounted treatment costs	discounted total costs
0	1.00000	0.00000	0.00000	0.00000	1.00000	2000.000	2000	
1	0.84800	0.15000	0.00000	0.00200	0.93252	2229.126	1937.864	
2	0.79410	0.18555	0.01575	0.00460	0.88712	2419.321	1876.527	
3	0.76618	0.19129	0.03492	0.00761	0.84843	2581.884	1816.350	
4	0.74536	0.18934	0.05431	0.01099	0.81254	2721.140	1757.443	
5	0.72674	0.18546	0.07310	0.01470	0.77840	2839.541	1699.850	
6	0.70900	0.18115	0.09111	0.01873	0.74572	2938.972	1643.593	
80	0.11578	0.02960	0.29714	0.55748	0.02693	451.752	83.173	
81	0.11298	0.02888	0.29430	0.56384	0.02571	433.941	79.591	
82	0.11025	0.02818	0.29145	0.57012	0.02455	416.778	76.159	
83	0.10758	0.02750	0.28858	0.57634	0.02344	400.244	72.871	
84	0.10498	0.02683	0.28570	0.58249	0.02237	384.317	69.722	
85	0.10244	0.02619	0.28280	0.58858	0.02136	368.979	66.704	
Sum					**21.08**			**204,123**

Steps

- Define transition matrix SOC
- Define transition matrix New treatment
- Define utility and cost values by health state
- Calculate health state probabilities over time
- Calculate expected QALYs and costs
- Cost-effectiveness analysis

Define transition matrix with standard of care

```r
p_hd <- 0.002         # constant probability of dying when Healthy (all-cause mortality)
p_hs1 <- 0.15         # probability of becoming Sick when Healthy
p_s1h <- 0.5          # probability of becoming Healthy when Sick
p_s1s2 <- 0.105       # probability of becoming Sicker when Sick
p_s1d <- 0.006        # constant probability of dying when Sick
p_s2d <- 0.02         # constant probability of dying when Sicker


p_hh <- 1  - p_hs1 - p_hd
p_s1s1 <- 1 - p_s1h - p_s1s2 - p_s1d
p_s2s2 <- 1 - p_s2d
```

```r
p_soc <- matrix(
  c(p_hh,  p_hs1,  0,      p_hd,
    p_s1h, p_s1s1, p_s1s2, p_s1d,
    0,     0,      p_s2s2, p_s2d,
    0,     0,      0,      1),
  byrow = TRUE,
  nrow = 4, ncol = 4
)
state_names <- c("H", "S1", "S2", "D")
colnames(p_soc) <- rownames(p_soc) <- state_names

print(p_soc)
```

```
       H    S1    S2     D
H  0.848 0.150 0.000 0.002
S1 0.500 0.389 0.105 0.006
S2 0.000 0.000 0.980 0.020
D  0.000 0.000 0.000 1.000
```

Relative risk and transition matrix with New treatment

```r
apply_rr <- function(p, rr = .8){

  p["H", "S1"] <- p["H", "S1"] * rr
  p["H", "S2"] <- p["H", "S2"] * rr
  p["H", "D"] <- p["H", "S2"] * rr
  p["H", "H"] <- 1 - sum(p["H", -1])

  p["S1", "S2"] <- p["S1", "S2"] * rr
  p["S1", "D"] <- p["S1", "D"] * rr
  p["S1", "S1"] <- 1 - sum(p["S1", -2])

  p["S2", "D"] <- p["S2", "D"] * rr
  p["S2", "S2"] <- 1 - sum(p["S2", -3])

  return(p)
}

p_new <- apply_rr(p_soc, rr = .8)
```

`p` Transition probability matrix SOC
`rr` Relative risk with new treatment (default is 0.8)

```
        H      S1    S2       D
H  0.8784 0.1200 0.000 0.0016
S1 0.5000 0.4112 0.084 0.0048
S2 0.0000 0.0000 0.984 0.0160
D  0.0000 0.0000 0.000 1.0000
```

Utility and costs

```
utility <- c(1, .75, .5, 0)
costs_medical <- c(2000, 4000, 15000, 0)
costs_treat_soc <- c(rep(2000, 3), 0)
costs_treat_new <- c(rep(12000, 3), 0)
```

```
> utility
[1] 1.000 0.75 0.500 0.000

> costs_medical
[1]  2000  4000 15000     0

> costs_treat_soc
[1] 2000 2000 2000    0

> costs_treat_new
[1] 12000 12000 12000     0
```

Simulation – health state probabilities

Matrix multiplication

```
x_init <- c(1, 0, 0, 0)

x_init %*% p_soc
```

```
         H    S1 S2     D
[1,] 0.848 0.15  0 0.002
```

```
x_init %*% p_soc %*% p_soc
```

```
             H      S1      S2        D
[1,] 0.794104 0.18555 0.01575 0.004596
```

Simulation – health state probabilities with a function

`x0`	The state vector at model cycle 0 (i.e., the initial state vector)
`p`	The transition probability matrix
`n_cycles`	The number of model cycles. (Default is 85)

```
sim_markov_chain <- function(x0, p, n_cycles = 85){

  x <- matrix(NA, ncol = length(x0), nrow = n_cycles) # Initialize Markov trace
  x <- rbind(x0, x)                                   # Markov trace at cycle 0 is initial state vector
  colnames(x) <- colnames(p)                          # Columns are the health states
  rownames(x) <- 0:n_cycles                           # Rows are the model cycles

  for (t in 1:n_cycles){                    # Simulating state vectors at each cycle with for loop
    x[t + 1, ] <- x[t, ] %*% p
  }

return(x)

}
```

Simulation – health state probabilities with a function

```
x_soc <- sim_markov_chain(x_init, p_soc)
```

```
 H         S1         S2          D
0  1.0000000 0.00000000 0.00000000 0.000000000
1  0.8480000 0.15000000 0.00000000 0.002000000
2  0.7941040 0.18555000 0.01575000 0.004596000
3  0.7661752 0.19129455 0.03491775 0.007612508
4  0.7453638 0.18933986 0.05430532 0.010990981
5  0.7267385 0.18545778 0.07309990 0.014703854
6  0.7090031 0.18115385 0.09111097 0.018732076
7  0.6918116 0.17681931 0.10830990 0.023059224
8  0.6750659 0.17255445 0.12470973 0.027669961
9  0.6587331 0.16838356 0.14033376 0.032549614
---
79 0.1186504 0.03032935 0.29995426 0.551065992
80 0.1157802 0.02959568 0.29713976 0.557484354
81 0.1129795 0.02887975 0.29430451 0.563836283
82 0.1102465 0.02818114 0.29145079 0.570121611
83 0.1075796 0.02749943 0.28858079 0.576340207
84 0.1049772 0.02683421 0.28569662 0.582491978
85 0.1024378 0.02618509 0.28280028 0.588576870
```

```
x_new <- sim_markov_chain(x_init, p_new)
```

```
H         S1         S2          D
0  1.0000000 0.00000000 0.00000000 0.000000000
1  0.8784000 0.12000000 0.00000000 0.001600000
2  0.8315866 0.15475200 0.01008000 0.003581440
3  0.8078416 0.16342441 0.02291789 0.005816068
4  0.7913203 0.16414111 0.03627885 0.008259738
5  0.7771663 0.16245326 0.04948624 0.010894190
6  0.7638895 0.16006074 0.06234054 0.013709211
7  0.7510309 0.15748372 0.07478819 0.016697175
8  0.7384474 0.15488101 0.08682021 0.019851357
9  0.7260927 0.15230076 0.09844109 0.023165425
---
79 0.2230518 0.04678750 0.31880122 0.411359506
80 0.2193224 0.04600523 0.31763055 0.417041788
81 0.2156554 0.04523604 0.31641290 0.422695618
82 0.2120498 0.04447971 0.31515012 0.428320406
83 0.2085044 0.04373603 0.31384402 0.433915590
84 0.2050182 0.04300478 0.31249634 0.439480634
85 0.2015904 0.04228575 0.31110880 0.445015028
```

Computing a present value with a function

```
pv <- function(z, dr, t) {

  z/(1 + dr)^t

}
```

`z` A numeric quantity
`dr` Discount rate
`t` Vector of times to compute the present value

```
pv(1000, dr = .03, t = 0:4)
```

```
[1] 1000.0000  970.8738  942.5959  915.1417  888.4870
```

QALYs after 1st cycle

```
x_soc[2, ] # State occupancy probabilities after 1st cycle
```

```
    H    S1    S2     D
0.848 0.150 0.000 0.002
```

```
sum(x_soc[2, 1:3]) # Expected life-years after 1st cycle
```

```
[1] 0.998
```

```
sum(x_soc[2, ] * utility) # Expected utility after 1st cycle
```

```
[1] 0.9605
```

```
sum(pv(x_soc[2, ] * utility, .03, 1)) # Expected discounted utility after 1st cycle
```

```
[1] 0.9325
```

Discounted QALYs for each cycle

```r
compute_qalys <- function(x, utility, dr = .03){

  n_cycles <- nrow(x) - 1
  pv(x %*% utility, dr, 0:n_cycles)

}
```

```r
dqalys_soc <- compute_qalys(x_soc, utility = utility)
dqalys_new <- compute_qalys(x_new, utility = utility)
```

```r
head(dqalys_soc)
```

```
        [,1]
0 1.0000000
1 0.9325243
2 0.8871161
3 0.8484324
4 0.8125404
5 0.7784024
```

```r
head(dqalys_new)
```

```
        [,1]
0 1.0000000
1 0.9401942
2 0.8980022
3 0.8619435
4 0.8285724
5 0.7968343
```

Discounted cost for each cycle

```
compute_costs <- function(x, costs_medical, costs_treat, dr = .03){

  n_cycles <- nrow(x) - 1
  costs <- cbind(pv(x %*% costs_medical, dr, 0:n_cycles),
                 pv(x %*% costs_treat, dr, 0:n_cycles)
  )
  colnames(costs) <- c("medical", "treatment")
  return(costs)
}

dcosts_soc <- compute_costs(x_soc, costs_medical, costs_treat_soc)
dcosts_new <- compute_costs(x_new, costs_medical, costs_treat_new)
```

```
head(dcosts_soc)
```

```
   medical treatment
0 2000.000  2000.000
1 2229.126  1937.864
2 2419.321  1876.527
3 2581.884  1816.350
4 2721.140  1757.443
5 2839.541  1699.850
```

```
head(dcosts_new)
```

```
   medical treatment
0 2000.000  12000.00
1 2171.650  11631.84
2 2293.695  11270.64
3 2391.402  10917.83
4 2473.004  10573.78
5 2541.624  10238.54
```

Cost-effectiveness

```
(sum(dcosts_new[-1, ]) - sum(dcosts_soc[-1, ])) /
(sum(dqalys_new[-1, ]) - sum(dqalys_soc[-1, ]))
```

```
[1] 122946.8
```

```
format_costs <- function(x) formatC(x, format = "d", big.mark = ",")
format_qalys <- function(x) formatC(x, format = "f", digits = 2)

make_icer_tbl <- function(costs0, costs1, qalys0, qalys1){

  # Computations
  total_costs0 <- sum(costs0)
  total_costs1 <- sum(costs1)
  total_qalys0 <- sum(qalys0)
  total_qalys1 <- sum(qalys1)
  incr_total_costs <- total_costs1 - total_costs0
  inc_total_qalys <- total_qalys1 - total_qalys0
  icer <- incr_total_costs/inc_total_qalys


  # Make table
  tibble(
    `Costs` = c(total_costs0, total_costs1) %>%
    `Strategy` = c("SOC", "New"),
      format_costs(),
    `QALYs` = c(total_qalys0, total_qalys1) %>%
      format_qalys(),
    `Incremental costs` = c("--", incr_total_costs %>% format_costs()),
    `Incremental QALYs` = c("--", inc_total_qalys %>% format_qalys()),
    `ICER` = c("--", icer %>% format_costs())
) %>%
  kable() %>%
  kable_styling() %>%
  footnote(general = "Costs and QALYs are discounted at 3% per annum.", footnote_as_chunk = TRUE)
}
```

Cost-effectiveness

```
make_icer_tbl(costs0 = dcosts_soc[-1, ], costs1 = dcosts_new[-1, ],
              qalys0 = dqalys_soc[-1, ], qalys1 = dqalys_new[-1, ])
```

Strategy	Costs	QALYs	Incremental costs	Incremental QALYs	ICER
SOC	204,123	21.08	--	--	--
New	464,390	23.19	260,266	2.12	122,946

Note: Costs and QALYs are discounted at 3% per annum.

Steps

- Define transition matrix SOC
- Define transition matrix New treatment `apply_rr(p_soc, rr)`
- Define utility and cost values by health state
- Calculate health state probabilities over time `sim_markov_chain(x0, p, n_cycles)`
- Calculate expected QALYs and costs

```
pv(z, dr, t)
compute_qalys(x, utility, dr)
compute_costs(x, costs_medical, costs_treat, dr)
```

- Cost-effectiveness analysis `make_icer_tbl(costs0, costs1, qalys0, qalys1)`

Complete R script

01-markov-cohort.R

Source on Save | Run | Source

```r
## ---- Overview ------------------------------------------------------------
## @knitr R-packages
library("rcea")
library("knitr")
library("kableExtra")
library("magrittr")
library("tibble")

## ---- Model parameters ----------------------------------------------------
## @knitr transition-probabilities
p_hd <- 0.002 # constant probability of dying when Healthy (all-cause mortality)
p_hs1 <- 0.15 # probability of becoming Sick when Healthy
p_s1h <- 0.5 # probability of becoming Healthy when Sick
p_s1s2 <- 0.105 # probability of becoming Sicker when Sick
p_s1d <- .006 # constant probability of dying when Sick
p_s2d <- .02 # constant probability of dying when Sicker

## @knitr transition-probability-complements
p_hh <- 1  - p_hs1 - p_hd
p_s1s1 <- 1 - p_s1h - p_s1s2 - p_s1d
p_s2s2 <- 1 - p_s2d

## @knitr tpmatrix
p_soc <- matrix(
  c(p_hh,  p_hs1,  0,      p_hd,
    p_s1h, p_s1s1, p_s1s2, p_s1d,
    0,     0,      p_s2s2, p_s2d,
    0,     0,      0,      1),
  byrow = TRUE,
  nrow = 4, ncol = 4
)
state_names <- c("H", "S1", "S2", "D")
colnames(p_soc) <- rownames(p_soc) <- state_names
print(p_soc)

## @knitr apply_rr
apply_rr <- function(p, rr = .8){
  p["H", "S1"] <- p["H", "S1"] * rr
  p["H", "S2"] <- p["H", "S2"] * rr
  p["H", "D"] <- p["H", "D"] * rr
  p["H", "H"] <- 1 - sum(p["H", -1])

  p["S1", "S2"] <- p["S1", "S2"] * rr
  p["S1", "D"] <- p["S1", "D"] * rr
  p["S1", "S1"] <- 1 - sum(p["S1", -2])
```

Tutorial

rcea 0.0.1 Reference Tutorials

Simple Markov Cohort Model

2020-10-11

Source: `vignettes/01-markov-cohort.Rmd`

Contents

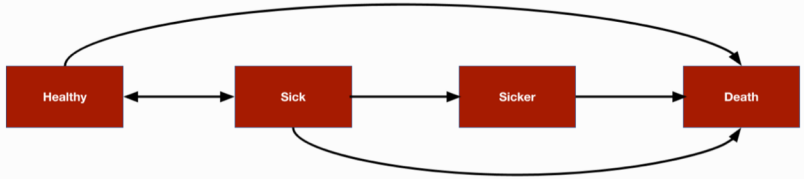
1 Overview

The most commonly used model for cost-effectiveness analysis (CEA) is the cohort discrete time state transition model (cDTSTM), commonly

https://hesim-dev.github.io/rcea/articles/01-markov-cohort.html

```
library("rcea")
library("knitr")
library("kableExtra")
library("magrittr")
library("tibble")
```

As an example, we will consider the 4-state sick-sicker model that has been described in more detail by Alarid-Escudero et al. The model will be used to compare two treatment strategies, a "New" treatment and the existing "standard of care (SOC)". The model consists 4 health states. Ordered from worst to best to worst, they are: Healthy (*H*), Sick (*S1*), Sicker (*S2*), and Death (*D*). Possible transitions from each state are displayed in the figure below.



2 Theory

Exercise 1: Simple Markov Cohort model

- *Modify R-script "01-markov-cohort-R"*
  - *Change relative risk*
  - *Change drug costs*
  - *Change utilities*
  - *Change follow-up time (i.e. number of cycles)*

```
sim_markov_chain(x0, p, n_cycles)
```

- *Run modified script*

Simple Markov cohort model – Incorporating probabilistic sensitivity analysis

Probabilistic sensitivity analysis

- The standard methodology for quantifying the impact of parameter uncertainty is probabilistic sensitivity analysis (PSA)
- Propagating uncertainty in the input parameters throughout the model by randomly sampling sets of input values from suitable probability distributions
- Probability distributions are determined according to the distributional properties of the statistical estimates, which, in turn, depend on the statistical techniques used and the distributions of the underlying data

- R is a natural programming language for performing PSA
- Random samples can be drawn from almost any probability distribution

Transition probabilities for SOC

- We assume that summary level data is available on transitions from the Healthy state (n = 1000), Sick state (n = 1000), and Sicker state (n = 800).

```
transitions_soc <- matrix(
  c(848, 150, 0,   2,
    500, 389, 105,  6,
    0,   0,   784, 16,
    0,   0,   0,   23),
  nrow = 4, byrow = TRUE)

state_names <- c("H", "S1", "S2", "D")
colnames(transitions_soc) <- rownames(transitions_soc) <- tolower(state_names)
```

```
      h  s1  s2  d
h   848 150   0  2
s1  500 389 105  6
s2    0   0 784 16
d     0   0   0 23
```

- The transitions from each state to the other 4 states can be modeled using a Dirichlet distribution

Combining all model parameters

```
params <- list(
  alpha_soc = transitions_soc,

  lrr_mean = log(.8),
  lrr_lower = log(.71),
  lrr_upper = log(.9),

  c_medical = c(H = 2000, S1 = 4000, S2 = 15000, D = 0),
  c_soc = 2000,
  c_new = 12000,

  u_mean = c(H = 1, S1 = .75, S2 = 0.5, D = 0),
  u_se = c(H = 0, S1 = 0.03, S2 = 0.05, D = 0.0)
)
```

Simulation

- The simulation proceeds by
  1. randomly sampling the parameters from the probability distributions specified
  2. running the Markov model for each draw of the parameters
- The result is a draw from the probability distribution of each of the model outputs of interest (i.e., state probabilities, QALYs, and costs).

Sampling parameters

- While Base R can be used to draw samples of parameters, the functions hesim::define_rng() and hesim::eval_rng() simplify this process.
- Any random number generation function can be used inside the define_rng() block

```
rng_def <- define_rng({

  lrr_se <- (lrr_upper - lrr_lower)/(2 * qnorm(.975))

  list( # Parameters to return
    p_soc = dirichlet_rng(alpha_soc),
    rr_new = lognormal_rng(lrr_mean, lrr_se),
    c_medical = gamma_rng(mean = c_medical, sd = c_medical),
    c_soc = c_soc,
    c_new = c_new,
    u = beta_rng(mean = u_mean, sd = u_se)
    )
}, n = 1000)

params_rng <- eval_rng(rng_def, params = params)

attr(params_rng, "n") <- rng_def$n
```

Sampling parameters

```
names(params_rng)
```

```
[1] "p_soc"     "rr_new"    "c_medical" "c_soc"     "c_new"     "u"
```

```
head(as.matrix(params_rng$p_soc))
```

```
          h_h      h_s1 h_s2         h_d      s1_h     s1_s1      s1_s2        s1_d s2_h s2_s1
[1,] 0.8686001 0.1293872    0 0.002012649 0.5003043 0.3847703 0.10868135 0.006244105    0     0
[2,] 0.8474698 0.1499868    0 0.002543368 0.5146218 0.3964675 0.08375492 0.005155768    0     0
[3,] 0.8559237 0.1428611    0 0.001215126 0.5117574 0.3775261 0.10868021 0.002036291    0     0
[4,] 0.8550586 0.1429657    0 0.001975648 0.5139857 0.3777508 0.10303967 0.005223752    0     0
[5,] 0.8678962 0.1304462    0 0.001657694 0.5164343 0.3815376 0.09725638 0.004771722    0     0
[6,] 0.8530231 0.1459388    0 0.001038023 0.4991514 0.3750200 0.11846736 0.007361245    0     0
         s2_s2       s2_d d_h d_s1 d_s2 d_d
[1,] 0.9786377 0.02136235   0    0    0   1
[2,] 0.9871702 0.01282980   0    0    0   1
[3,] 0.9786291 0.02137091   0    0    0   1
[4,] 0.9809938 0.01900624   0    0    0   1
[5,] 0.9793886 0.02061141   0    0    0   1
[6,] 0.9750211 0.02497889   0    0    0   1
```

Simulating the Markov model

- One way that a Markov simulation can be generalized is to store “input data” in an object, i.e. data frame.
- Input data might consist of
  - treatment strategies
  - patients and subgroups
    - For instance, if we were simulating different subgroups we might store the age and sex associated with the subgroup which could, in turn, be used as covariates in a statistical model.

```
data <- data.frame(
  strategy = c("New", "SOC")
)
```

```
  strategy
1      New
2      SOC
```

Simulating the Markov model – Create the function

- Set up a `sim_model()` function that runs the entire simulation.
  - Comprised of three smaller functions:
    - `sim_stateprobs()`
    - `compute_qalys()`
    - `compute_costs()`

Simulating the Markov model – Create the function

```
sim_stateprobs <- function(p0, rr, strategy, n_cycles){

  rr <- ifelse(strategy == "New", rr, 1)

  p <- tpmatrix(
    C,       p0$h_s1 * rr,  p0$h_s2 * rr,  p0$h_d * rr,
    p0$s1_h, C,             p0$s1_s2 * rr, p0$s1_d * rr,
    p0$s2_h, p0$s2_s1,      C,             p0$s2_d * rr,
    0,       0,             0,             1
  )

  x <- sim_markov_chain(x0 = c(1, 0, 0, 0),
                        p = matrix(as.matrix(p), ncol = 4, byrow = TRUE),
                        n_cycles = n_cycles)

  return(x)
}
```

`hesim::tpmatrix()` *makes it easy to define a transition probability matrix.*

C denotes that a given element is the complement of all other elements in that row, ensuring that the probabilities sum to 1.

`sim_markov_chain()` *is the function we created previously*

Simulating the Markov model – Create the function

```r
# QALYs
compute_qalys <- function(x, utility, dr = .03){

  n_cycles <- nrow(x) - 1
  pv(x %*% utility, dr, 0:n_cycles)
}
```

```r
# Costs
compute_costs <- function(x, costs_medical, costs_treat, dr = .03){

  n_cycles <- nrow(x) - 1
  costs_treat <- c(rep(costs_treat, 3), 0)
  costs <- cbind(
    pv(x %*% costs_medical, dr, 0:n_cycles),
    pv(x %*% costs_treat, dr, 0:n_cycles)
  )
  colnames(costs) <- c("dcost_med", "dcost_treat")
  return(costs)
}
```

```r
sim_model <- function(params_rng, data, n_cycles = 85, dr_qalys = .03, dr_costs = .03){
  # Initialize array of matrices
  n_samples <- attr(params_rng, "n")
  n_strategies <- nrow(data)
  out <- array(NA, dim = c(n_cycles + 1, 7, n_samples * n_strategies))
  dimnames(out) <- list(NULL, c("H", "S1", "S2", "D", "dqalys", "dcosts_med", "dcosts_treat"), NULL)

  # Run the simulation
  i <- 1
  for (s in 1:n_samples){ # Start PSA loop
    for (k in 1:n_strategies) { # Start treatment strategy loop
      x <- sim_stateprobs(p0 = params_rng$p_soc[s, ],
                          rr = params_rng$rr_new[s],
                          strategy = data$strategy[k],
                          n_cycles = n_cycles)
      dqalys <- compute_qalys(x, utility = unlist(params_rng$u[s]),dr = dr_qalys)
      dcosts <- compute_costs(x,
                              costs_medical = unlist(params_rng$c_medical[s]),
                              costs_treat = ifelse(data$strategy[k] == "SOC",
                                                   params_rng$c_soc,
                                                   params_rng$c_new), dr = dr_costs)
      out[, , i] <- cbind(x, dqalys, dcosts)
      i <- i + 1
    } # End treatment strategy loop
  } # End PSA loop

  # Store metadata and return
  attr(out, "n_samples") <- n_samples
  attr(out, "strategies") <- data$strategy
  return(out)
}
```

PSA. samples; tx strategy; no. cycles; discount rates

An array to store the output.
A series of matrices each with n_cycles rows and columns for each output.
There is one matrix for each parameter sample for the PSA and treatment strategy.

Simulates for each parameter sample and treatment strategy
- *state probabilities (with sim_stateprobs())*
- *QALYs (with compute_qalys())*
- *costs (with compute_costs())*

The number of parameter samples and the names of the treatment strategies are saved as attributes (i.e., metadata) to the array.

Simulating the Markov model

```
sim_out <- sim_model(params_rng, data = data)

head(sim_out[, , 1])
```

```
             H        S1          S2           D    dqalys dcosts_med dcosts_treat
[1,] 1.0000000 0.0000000 0.000000000 0.000000000 1.0000000   639.2051     12000.00
[2,] 0.8927749 0.1055827 0.000000000 0.001642364 0.9440327   968.8714     11631.35
[3,] 0.8498706 0.1371191 0.009363733 0.003646603 0.9027066  1072.4091     11269.90
[4,] 0.8273444 0.1453902 0.021361082 0.005904295 0.8667374  1105.0422     10916.86
[5,] 0.8113717 0.1463692 0.033882825 0.008376274 0.8332591  1114.3372     10572.54
[6,] 0.7976015 0.1450801 0.046273111 0.011045289 0.8013670  1114.9710     10236.97
```

Reorganize output

```
sim_out <- array_to_dt(sim_out)

head(sim_out)
```

Convert a 3D array (faster to store data) to a data.table so we can summarize outcomes for each parameter sample and treatment strategy very quickly.

```
   cycle strategy sample         H        S1          S2           D    dqalys dcosts_med dcosts_treat
1:     0      New      1 1.0000000 0.0000000 0.000000000 0.000000000 1.0000000   639.2051     12000.00
2:     1      New      1 0.8927749 0.1055827 0.000000000 0.001642364 0.9440327   968.8714     11631.35
3:     2      New      1 0.8498706 0.1371191 0.009363733 0.003646603 0.9027066  1072.4091     11269.90
4:     3      New      1 0.8273444 0.1453902 0.021361082 0.005904295 0.8667374  1105.0422     10916.86
5:     4      New      1 0.8113717 0.1463692 0.033882825 0.008376274 0.8332591  1114.3372     10572.54
6:     5      New      1 0.7976015 0.1450801 0.046273111 0.011045289 0.8013670  1114.9710     10236.97
```

Cost-effectiveness analysis

```
ce_out <- sim_out[cycle != 0,
                  .(dqalys = sum(dqalys),
                    dcosts = sum(dcosts_med) + sum(dcosts_treat)),
                  by = c("sample", "strategy")]
ce_out
```

```
       sample strategy   dqalys   dcosts
   1:       1      New 23.30567 372636.3
   2:       1      SOC 21.36614 103030.7
   3:       2      New 23.49382 380159.8
   4:       2      SOC 22.71959 104017.1
   5:       3      New 23.35243 463270.5
  ---
1996:     998      SOC 20.63295 166641.1
1997:     999      New 21.82111 356686.6
1998:     999      SOC 19.58578 104643.0
1999:    1000      New 23.03594 559476.0
2000:    1000      SOC 21.79923 320811.6
```

Cost-effectiveness analysis

```
ce_out_wider <- dcast(ce_out, sample ~ strategy,
                      value.var = c("dqalys", "dcosts"))

ce_out_wider
```

```
      sample dqalys_New dqalys_SOC dcosts_New dcosts_SOC
   1:      1   23.30567   21.36614   372636.3  103030.72
   2:      2   23.49382   22.71959   380159.8  104017.07
   3:      3   23.35243   21.81066   463270.5  207518.50
   4:      4   23.49622   21.45324   680491.2  474308.21
   5:      5   24.11580   22.73777   492869.3  239480.34
  ---
 996:    996   23.05054   20.86090   578855.4  336586.31
 997:    997   24.22340   22.48457   379255.5   93578.69
 998:    998   22.22956   20.63295   452278.4  166641.12
 999:    999   21.82111   19.58578   356686.6  104642.99
1000:   1000   23.03594   21.79923   559476.0  320811.62
```

Cost-effectiveness analysis

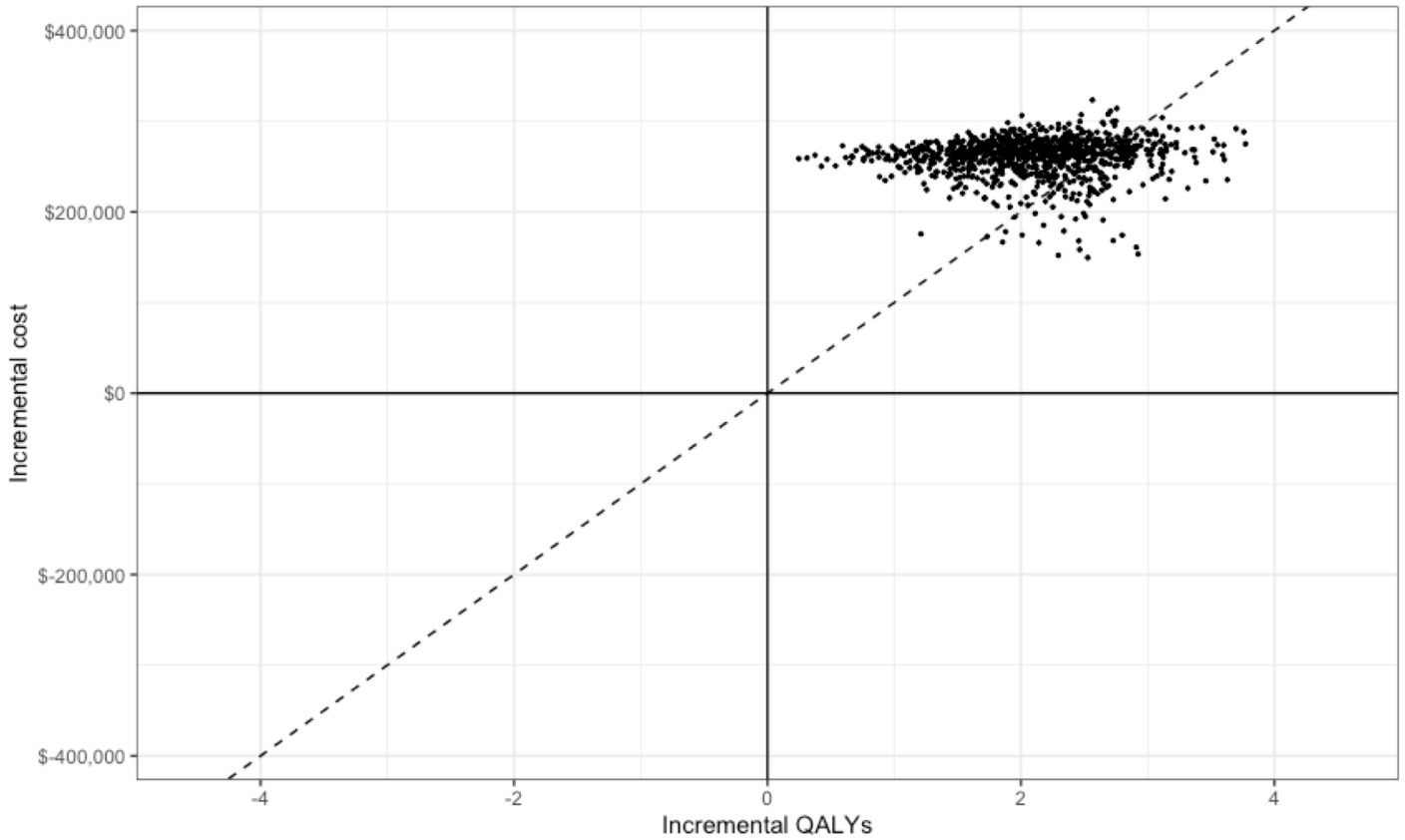
```
ce_out_wider[, idcosts := dcosts_New - dcosts_SOC]
ce_out_wider[, idqalys := dqalys_New - dqalys_SOC]
```

```
      sample dqalys_New dqalys_SOC dcosts_New dcosts_SOC  idcosts   idqalys
   1:      1   23.30567   21.36614   372636.3  103030.72 269605.6 1.9395308
   2:      2   23.49382   22.71959   380159.8  104017.07 276142.7 0.7742344
   3:      3   23.35243   21.81066   463270.5  207518.50 255752.0 1.5417635
   4:      4   23.49622   21.45324   680491.2  474308.21 206183.0 2.0429732
   5:      5   24.11580   22.73777   492869.3  239480.34 253388.9 1.3780308
  ---
 996:    996   23.05054   20.86090   578855.4  336586.31 242269.1 2.1896429
 997:    997   24.22340   22.48457   379255.5   93578.69 285676.8 1.7388263
 998:    998   22.22956   20.63295   452278.4  166641.12 285637.3 1.5966079
 999:    999   21.82111   19.58578   356686.6  104642.99 252043.6 2.2353267
1000:   1000   23.03594   21.79923   559476.0  320811.62 238664.4 1.2367111
```

```
ce_out_wider[, .(icer = mean(idcosts)/mean(idqalys))]
```

```
      icer
1: 127000.5
```

Cost-effectiveness plane



Steps

- Define uncertainty for all model input parameters
- Sampling parameter values `hesim::define_rng(), hesim::eval_rng()`
- Define treatment strategies and population (data)
- Simulating the Markov model

```
sim_model(params_rng, data, n_cycles, dr_qalys, dr_costs)
sim_stateprobs(po, rr, strategy, n_cycles)
compute_qalys(x, utility, dr)
compute_costs(x, cost_medical, costs_treat, dr)
```

- Reorganize output `rbind_array(), array_to_dt()`
- Cost-effectiveness analysis

Complete R script

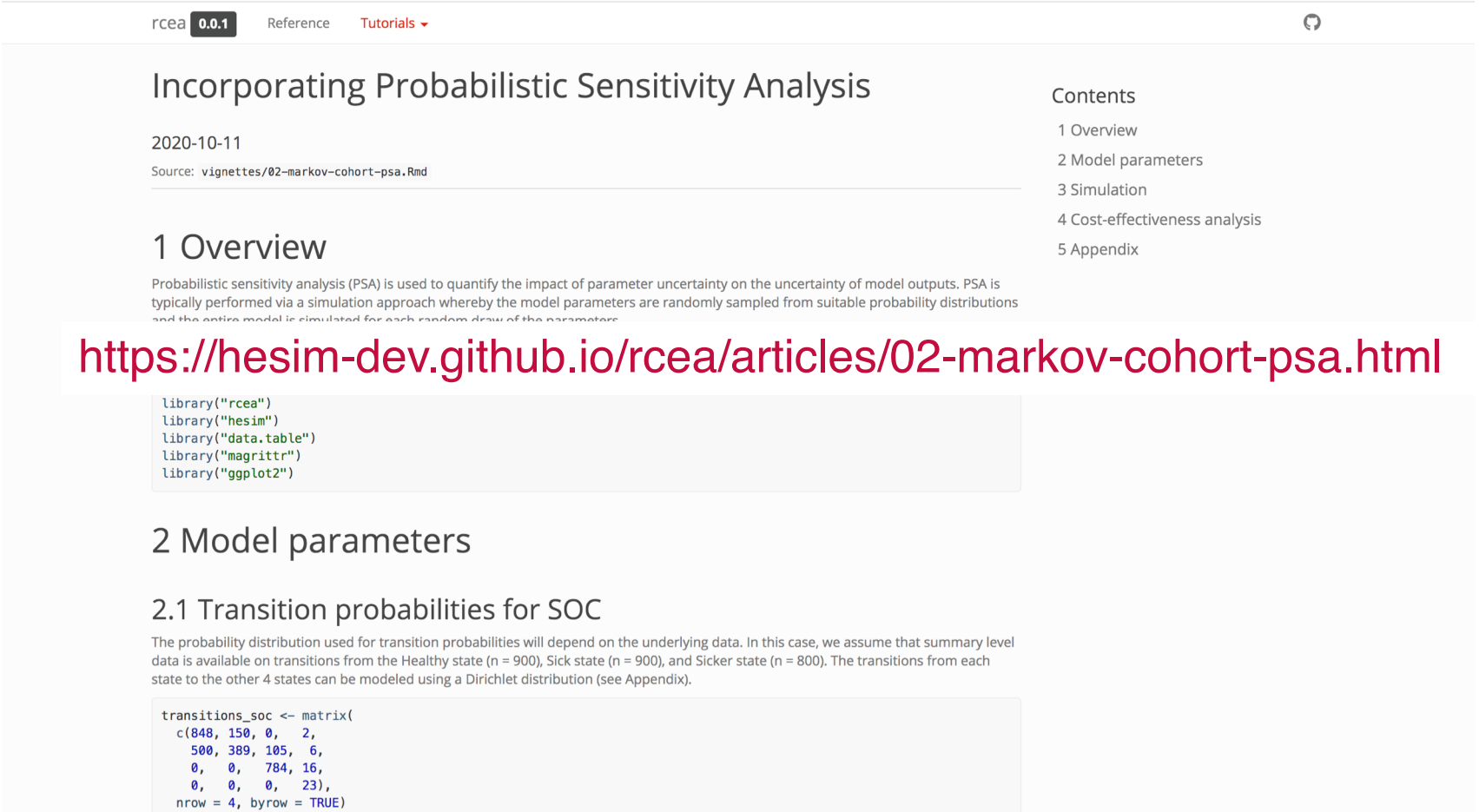
02-markov-cohort-psa.R

Source on Save | Run | Source

```r
## ---- Overview ------------------------------------------------------------
## @knitr R-packages
library("rcea")
library("hesim")
library("data.table")
library("magrittr")
library("ggplot2")

## ---- Model parameters ----------------------------------------------------
## @knitr tpmatrix
transitions_soc <- matrix(
  c(848, 150, 0,   2,
    500, 389, 105,  6,
    0,   0,   784, 16,
    0,   0,   0,   23),
  nrow = 4, byrow = TRUE)
state_names <- c("H", "S1", "S2", "D")
colnames(transitions_soc) <- rownames(transitions_soc) <- tolower(state_names)

## @knitr all-parameters
params <- list(
  alpha_soc = transitions_soc,
  lrr_mean = log(.8),
  lrr_lower = log(.71),
  lrr_upper = log(.9),
  c_medical = c(H = 2000, S1 = 4000, S2 = 15000, D = 0),
  c_soc = 2000,
  c_new = 12000,
  u_mean = c(H = 1, S1 = .75, S2 = 0.5, D = 0),
  u_se = c(H = 0, S1 = 0.03, S2 = 0.05, D = 0.0)
)

## ---- Simulation ----------------------------------------------------------
## @knitr sample-parameters
rng_def <- define_rng({
  lrr_se <- (lrr_upper - lrr_lower)/(2 * qnorm(.975)) # Local object
  # not returned
  list( # Parameters to return
    p_soc = dirichlet_rng(alpha_soc),
    rr_new = lognormal_rng(lrr_mean, lrr_se),
    c_medical = gamma_rng(mean = c_medical, sd = c_medical),
    c_soc = c_soc,
    c_new = c_new,
    u = beta_rng(mean = u_mean, sd = u_se)
  )
```

106:49 | sim_model(params_rng, data, n_cycles, dr_qalys, dr_costs) | R Script

Tutorial


rcea 0.0.1
Reference
Tutorials
Incorporating Probabilistic Sensitivity Analysis
2020-10-11
Source: vignettes/02-markov-cohort-psa.Rmd
Contents
1 Overview
2 Model parameters
3 Simulation
4 Cost-effectiveness analysis
5 Appendix
1 Overview
Probabilistic sensitivity analysis (PSA) is used to quantify the impact of parameter uncertainty on the uncertainty of model outputs. PSA is typically performed via a simulation approach whereby the model parameters are randomly sampled from suitable probability distributions
https://hesim-dev.github.io/rcea/articles/02-markov-cohort-psa.html
library("rcea")
library("hesim")
library("data.table")
library("magrittr")
library("ggplot2")
2 Model parameters
2.1 Transition probabilities for SOC
The probability distribution used for transition probabilities will depend on the underlying data. In this case, we assume that summary level data is available on transitions from the Healthy state (n = 900), Sick state (n = 900), and Sicker state (n = 800). The transitions from each state to the other 4 states can be modeled using a Dirichlet distribution (see Appendix).
transitions_soc <- matrix(
c(848, 150, 0, 2,
500, 389, 105, 6,
0, 0, 784, 16,
0, 0, 0, 23),
nrow = 4, byrow = TRUE)

Exercise 2: Incorporating probabilistic sensitivity analysis

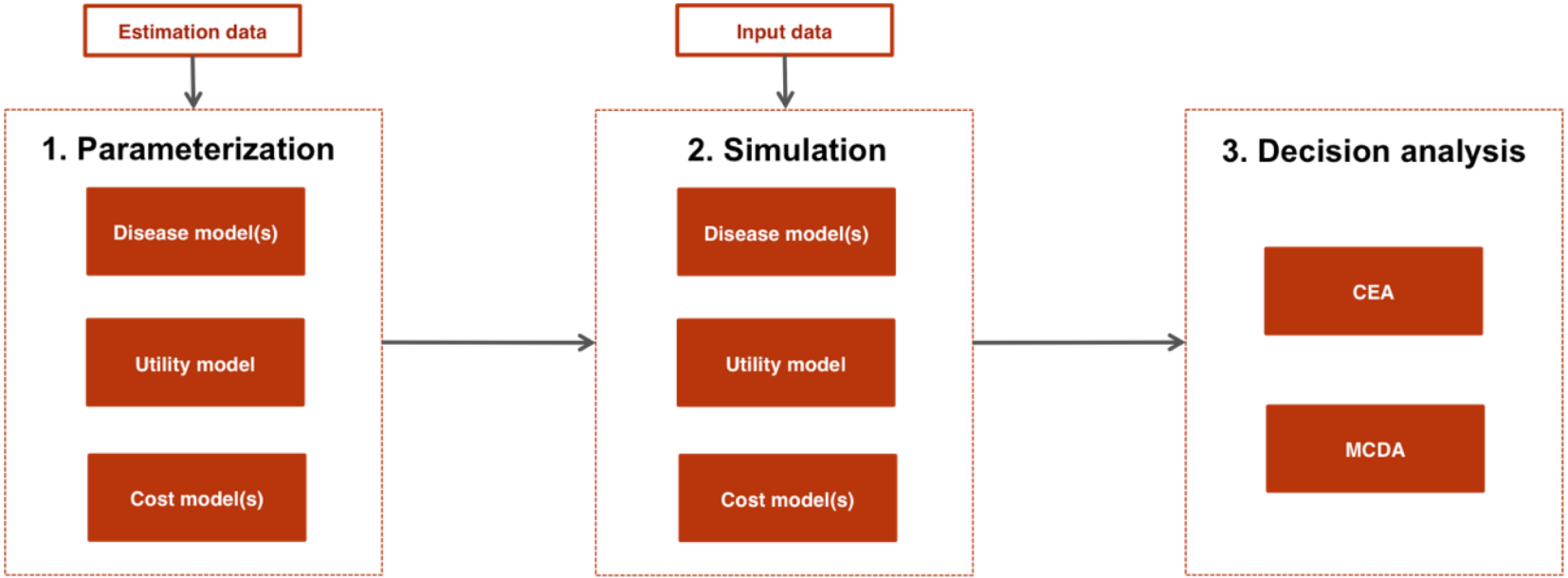
- *Modify R-script "02-markov-cohort-psa.R"*
  - *Reduce sample size of data for transition matrix by 50%*
  - *Increase confidence interval for relative risk*
- *Run modified script*

Simple Markov cohort model with hesim

What is hesim?

- A modular and computationally efficient R package for building simulation models for economic evaluation
- Supports both cohort and individual-level models, encompassing Markov (time-homogeneous and time-inhomogeneous) and semi-Markov processes
  - cohort discrete time state transition models (cDTSTM)
  - individual-level continuous time state transition models (iCTSTM)
  - n-state partitioned survival models (PSM)
- Parameterization by fitting a statistical model in R or by using estimates from external sources
- Nearly all simulation code written in C++ under the hood, but you don't need to know C++ to use it!

`hesim` modeling process



Economic models with `hesim`

1. Model set-up
2. Parameters
3. Simulation
    a. Construction of model
    b. Simulation of outcomes
4. Cost-effectiveness analysis

Model setup

- Define target population and intervention strategies

```
strategies <- data.frame(
  strategy_id = 1:2,
  strategy_name = c("SOC", "New")
)
patients <- data.frame(
  patient_id = 1,
  age = 25
)
hesim_dat <- hesim_data(
  strategies = strategies,
  patients = patients
)

print(hesim_dat)
```

```
$strategies
  strategy_id strategy_name
1           1           SOC
2           2           New

$patients
  patient_id age
1          1  25

attr(,"class")
[1] "hesim_data"
```

Model parameters

- Same list of parameters as used before

```
params <- list(
  alpha_soc = transitions_soc,
  lrr_mean = log(.8),
  lrr_lower = log(.71),
  lrr_upper = log(.9),
  c_medical = c(H = 2000, S1 = 4000, S2 = 15000),
  c_soc = 2000,
  c_new = 12000,
  u_mean = c(H = 1, S1 = .75, S2 = 0.5),
  u_se = c(H = 0, S1 = 0.03, S2 = 0.05)
)
```

Model parameters – Random number generation (for PSA)

```
rng_def <- define_rng({
  lrr_se <- (lrr_upper - lrr_lower)/(2 * qnorm(.975))

  list( # Parameters to return
    p_soc = dirichlet_rng(alpha_soc),
    rr_new = lognormal_rng(lrr_mean, lrr_se),
    c_medical = gamma_rng(mean = c_medical, sd = c_medical),
    c_soc = c_soc,
    c_new = c_new,
    u = beta_rng(mean = u_mean, sd = u_se)
  )
}, n = 1000)
```

Model parameters – Transformed parameters

- It is typically useful to ”transform” underlying parameters (`params`) into transformed parameters (`tparams`) more relevant for the simulation
  - e.g., predicting an element of a transition probability matrix as a function of the treatment strategy
- We previously did this in the base R PSA example…

```r
sim_model <- function(params_rng, data, n_cycles = 85, dr_qalys = .03, dr_costs = .03){

  # Initialize array of matrices
  n_samples <- attr(params_rng, "n")
  n_strategies <- nrow(data)
  out <- array(NA, dim = c(n_cycles + 1, 7, n_samples * n_strategies))
  dimnames(out) <- list(NULL, c("H", "S1", "S2", "D", "dqalys", "dcosts_med", "dcosts_treat"), NULL)

  # Run the simulation
  i <- 1
  for (s in 1:n_samples){ # Start PSA loop
    for (k in 1:n_strategies) { # Start treatment strategy loop
      x <- sim_stateprobs(p0 = params_rng$p_soc[s, ],
                          rr = params_rng$rr_new[s],
                          strategy = data$strategy[k],
                          n_cycles = n_cycles)
      dqalys <- compute_qalys(x, utility = unlist(params_rng$u[s]),dr = dr_qalys)
      dcosts <- compute_costs(x,
                              costs_medical = unlist(params_rng$c_medical[s]),
                              costs_treat = ifelse(data$strategy[k] == "SOC",
                                                   params_rng$c_soc,
                                                   params_rng$c_new), dr = dr_costs)
      out[, , i] <- cbind(x, dqalys, dcosts)
      i <- i + 1
    } # End treatment strategy loop
  } # End PSA loop

  # Store metadata and return
  attr(out, "n_samples") <- n_samples
  attr(out, "strategies") <- data$strategy
  return(out)
}
sim_out <- sim_model(params_rng, data = data)
sim_out <- array_to_dt(sim_out)
```

```r
sim_stateprobs <- function(p0, rr, strategy, n_cycles){

  rr <- ifelse(strategy == "New", rr, 1)

  p <- tpmatrix(
    C,       p0$h_s1 * rr,  p0$h_s2 * rr,  p0$h_d * rr,
    p0$s1_h, C,             p0$s1_s2 * rr, p0$s1_d * rr,
    p0$s2_h, p0$s2_s1,      C,             p0$s2_d * rr,
    0,       0,             0,             1
  )

  x <- sim_markov_chain(x0 = c(1, 0, 0, 0),
                        p = matrix(as.matrix(p), ncol = 4, byrow = TRUE),
                        n_cycles = n_cycles)

  return(x)
}
```

Model parameters – Transformed parameters

- A `define_tparams()` block in `hesim` does the same thing, but most of the implementation is done for you (efficiently)
- A `define_tparams()` block returns:
  - `tpmatrix`: The transition probability matrix
  - `utility`: Utility assigned to each health state
  - `costs`: Costs assigned to each health state for each cost category
- All parameters are "transformed" using:
  1. Columns of input data
  2. Parameters returned by `define_rng()`

Model parameters – Transformed parameters

- *Input data* (treatment strategies and patients) can be generated using `expand()`

```
input_data <- hesim::expand(hesim_dat, by = c("strategies", "patients"))

head(input_data)
```

```
   strategy_id patient_id strategy_name age
1:           1          1           SOC  25
2:           2          1           New  25
```

Model parameters – Transformed parameters

- You write mathematical expressions
- Automatically loops over PSA iterations and input data rows

```
tparams_def <- define_tparams({
  # The treatment effect (relative risk) varies by
  # strategies (SOC is the reference strategy)

  rr <- ifelse(strategy_name == "SOC", 1, rr_new)

  list(
    tpmatrix = tpmatrix(
      C,          p_soc$h_s1 * rr, p_soc$h_s2 * rr,  p_soc$h_d * rr,
      p_soc$s1_h, C,               p_soc$s1_s2 * rr, p_soc$s1_d * rr,
      p_soc$s2_h, p_soc$s2_s1,     C,                p_soc$s2_d * rr,
      0,          0,               0,                1
    ),
    utility = u,
    costs = list(
      treatment = ifelse(strategy_name == "SOC", c_soc, c_new),
      medical = c_medical
    )
  )
})
```

Input data

Parameter

Parameter (defined above)

Recall that C denotes the "complement", ensuring that the probabilities in a row sum to 1

Parameter

Input data

Parameter

Parameter

Simulation - Construct the model

- Combine the underlying parameters with the expressions for random number generation and parameter transformation

```
mod_def <- define_model(tparams_def = tparams_def,
                        rng_def = rng_def,
                        params = params)
```

- A economic model (of class `CohortDtstm`) can be created from a defined model (of class `model_def`) and data using the generic function create_CohortDtstm()

```
econmod <- create_CohortDtstm(mod_def, input_data)
```

- This object consists of a
  - transition model for simulating transition probabilities with `$sim_stateprobs()`
  - a utility model for simulating quality-adjusted life-years with `$sim_qalys()`
  - a set of cost models (for each cost category) for simulating costs with `$sim_costs()`

Simulation – Simulating outcomes

- Health state probabilities

```
econmod$sim_stateprobs(n_cycles = 85)
```

```
   sample strategy_id patient_id grp_id state_id t      prob
1:      1           1          1      1        1 0 1.0000000
2:      1           1          1      1        1 1 0.8466964
3:      1           1          1      1        1 2 0.7929241
4:      1           1          1      1        1 3 0.7652357
5:      1           1          1      1        1 4 0.7446251
6:      1           1          1      1        1 5 0.7261680
```

- QALYs

```
econmod$sim_qalys(
  dr = 0.03, lys = TRUE,
  integrate_method = "riemann_right"
)
```

```
   sample strategy_id patient_id grp_id state_id   dr     qalys       lys
1:      1           1          1      1        1 0.03 14.604941 14.604941
2:      1           1          1      1        2 0.03  2.718281  3.660628
3:      1           1          1      1        3 0.03  2.595318  7.331531
4:      1           2          1      1        1 0.03 17.633196 17.633196
5:      1           2          1      1        2 0.03  2.632315  3.544860
6:      1           2          1      1        3 0.03  2.083739  5.886368
```

- Costs

```
econmod$sim_costs(
  dr = 0.03,
  integrate_method = "riemann_right"
)
```

```
   sample strategy_id patient_id grp_id state_id   dr  category      costs
1:      1           1          1      1        1 0.03 treatment  29209.882
2:      1           1          1      1        2 0.03 treatment   7321.257
3:      1           1          1      1        3 0.03 treatment  14663.061
4:      1           2          1      1        1 0.03 treatment 211598.347
5:      1           2          1      1        2 0.03 treatment  42538.323
6:      1           2          1      1        3 0.03 treatment  70636.410
```

Cost-effectiveness analysis

- CEAs can be performed directly from the simulation output with `hesim`. Other R packages such as BCEA could also be considered
- First we need to aggregate (i.e., ”summarize”) costs and QALYs across health states

```
ce_sim <- econmod$summarize()
```

```
$costs
       category   dr sample strategy_id     costs grp_id
   1: treatment 0.03      1           1  51194.20      1
   2: treatment 0.03      1           2 324773.08      1
   3: treatment 0.03      2           1  48720.41      1
   4: treatment 0.03      2           2 316828.55      1
   5: treatment 0.03      3           1  51943.80      1
  ---
5996:     total 0.03    998           2 402500.90      1
5997:     total 0.03    999           1 197991.68      1
5998:     total 0.03    999           2 461755.98      1
5999:     total 0.03   1000           1 203541.38      1
6000:     total 0.03   1000           2 461558.77      1

$qalys
        dr sample strategy_id    qalys grp_id
   1: 0.03      1           1 19.91854      1
   2: 0.03      1           2 22.34925      1
   3: 0.03      2           1 19.87522      1
   4: 0.03      2           2 22.71959      1
   5: 0.03      3           1 21.39403      1
  ---
1996: 0.03    998           2 23.08819      1
1997: 0.03    999           1 21.25388      1
1998: 0.03    999           2 22.98019      1
1999: 0.03   1000           1 20.75811      1
2000: 0.03   1000           2 23.84501      1

attr(,"class")
[1] "ce"
```

Cost-effectiveness analysis

- Here, we will consider a pairwise comparison between the new treatment and SOC with the `cea_pw()` function

```
cea_pw_out <- cea_pw(ce_sim, comparator = 1,
                     dr_qalys = 0.03, dr_costs = 0.03)
```

- Although `cea_pw()` allows users to summarize output from a PSA we will just create an ICER table using means for now

```
icer_tbl(cea_pw_out, k = 50000, colnames = strategies$strategy_name)
```

```
                 SOC New
Incremental QALYs "-" "2.07 (0.98, 3.17)"
Incremental costs "-" "257,766 (206,974, 289,881)"
Incremental NMB   "-" "-154,131 (-213,515, -79,172)"
ICER              "-" "124,362"
Conclusion        "-" "Not cost-effective"
```

(at WTP threshold of $50,000)

Steps with `hesim`

1. Model set-up
   - Specify the treatment strategies and target population(s)
     - `hesim_data()`
2. Parameters
   - Estimate or define the parameters of the economic model
     - `define_rng()`, `define_tparams()`
3. Simulation
   a. Construction of model
      - Create an economic model—consisting of separate statistical models for disease progression, costs, and utilities—that simulate outcomes as a function of *input data* (derived from Step 1) and *parameters* (from Step 2)
      - `define_model()`, `create_CohortDtstm()`
   b. Simulation of outcomes
      - Simulate outcomes (disease progression, costs, and quality-adjusted life-years (QALYs)) using the model constructed in Step 3
      - `$sim_stateprobs()`, `$sim_qalys()`, `$sim_costs()`
4. Cost-effectiveness analysis

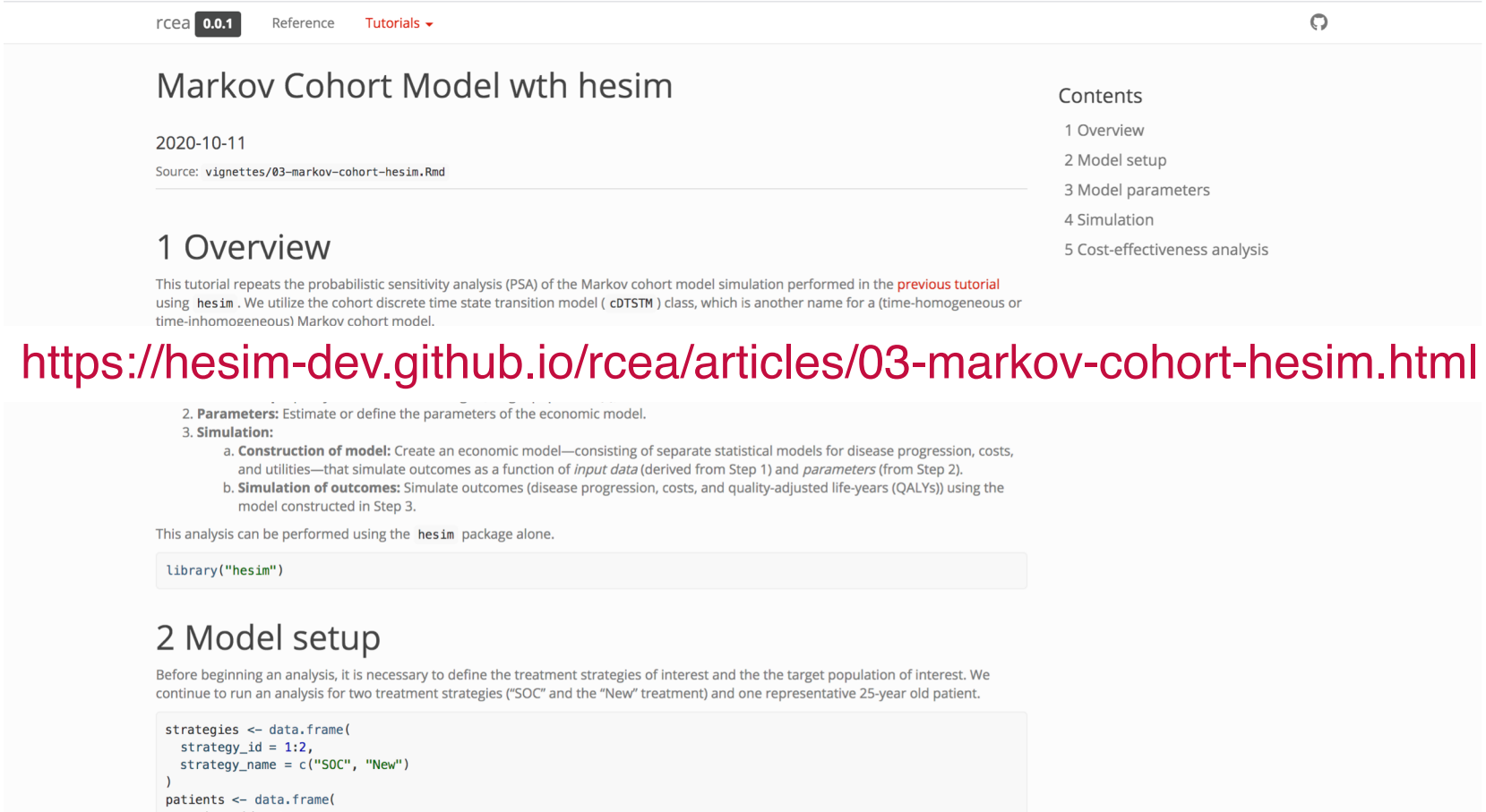
Complete R script

03-markov-cohort-hesim.R

```r
## ---- Overview ------------------------------------------------------------
## @knitr R-packages
library("hesim")

## ---- Model setup ---------------------------------------------------------
## @knitr hesim-data
strategies <- data.frame(
  strategy_id = 1:2,
  strategy_name = c("SOC", "New")
)
patients <- data.frame(
  patient_id = 1,
  age = 25
)
hesim_dat <- hesim_data(
  strategies = strategies,
  patients = patients
)
print(hesim_dat)

## ---- Model parameters ----------------------------------------------------
## @knitr transitions
transitions_soc <- matrix(
  c(848, 150, 0,   2,
    500, 389, 105,  6,
    0,   0,   784, 16,
    0,   0,   0,   23),
  nrow = 4, byrow = TRUE)
state_names <- c("H", "S1", "S2", "D")
colnames(transitions_soc) <- rownames(transitions_soc) <- tolower(state_names)

## @knitr all-parameters
params <- list(
  alpha_soc = transitions_soc,
  lrr_mean = log(.8),
  lrr_lower = log(.71),
  lrr_upper = log(.9),
  c_medical = c(H = 2000, S1 = 4000, S2 = 15000),
  c_soc = 2000,
  c_new = 12000,
  u_mean = c(H = 1, S1 = .75, S2 = 0.5),
  u_se = c(H = 0, S1 = 0.03, S2 = 0.05)
)

## @knitr define_rng
```

115:59 Cost-effectiveness analysis R Script

Tutorial


rcea 0.0.1
Reference
Tutorials
Markov Cohort Model wth hesim
2020-10-11
Source: vignettes/03-markov-cohort-hesim.Rmd
Contents
1 Overview
2 Model setup
3 Model parameters
4 Simulation
5 Cost-effectiveness analysis
1 Overview
This tutorial repeats the probabilistic sensitivity analysis (PSA) of the Markov cohort model simulation performed in the previous tutorial using hesim . We utilize the cohort discrete time state transition model (cDTSTM) class, which is another name for a (time-homogeneous or time-inhomogeneous) Markov cohort model.
https://hesim-dev.github.io/rcea/articles/03-markov-cohort-hesim.html
2. Parameters: Estimate or define the parameters of the economic model.
3. Simulation:
a. Construction of model: Create an economic model—consisting of separate statistical models for disease progression, costs, and utilities—that simulate outcomes as a function of input data (derived from Step 1) and parameters (from Step 2).
b. Simulation of outcomes: Simulate outcomes (disease progression, costs, and quality-adjusted life-years (QALYs)) using the model constructed in Step 3.
This analysis can be performed using the hesim package alone.
library("hesim")
2 Model setup
Before beginning an analysis, it is necessary to define the treatment strategies of interest and the the target population of interest. We continue to run an analysis for two treatment strategies ("SOC" and the "New" treatment) and one representative 25-year old patient.
strategies <- data.frame(
strategy_id = 1:2,
strategy_name = c("SOC", "New")
)
patients <- data.frame(

Exercise 3: Markov cohort model with hesim

- *Modify R-script "03-markov-cohort-hesim.R"*
  - *Increase confidence interval for relative risk*
  - *Modify the mean health state utility value*
  - *Remove impact of the intervention on transitions from "healthy" to "sick", "sicker" , and "death" (row 71)*
- *Run modified script*

```
tparams_def <- define_tparams({
  ## The treatment effect (relative risk) is transformed so that it varies by
  ## strategies (SOC is the reference strategy)
  rr <- ifelse(strategy_name == "SOC", 1, rr_new)

  list(
    tpmatrix = tpmatrix(
      C,          p_soc$h_s1 * rr, p_soc$h_s2 * rr,  p_soc$h_d * rr,
      p_soc$s1_h, C,               p_soc$s1_s2 * rr, p_soc$s1_d * rr,
      p_soc$s2_h, p_soc$s2_s1,     C,                p_soc$s2_d * rr,
      0,          0,               0,                1
    ),
    utility = u,
    costs = list(
      treatment = ifelse(strategy_name == "SOC", c_soc, c_new),
      medical = c_medical
    )
  )
})
```

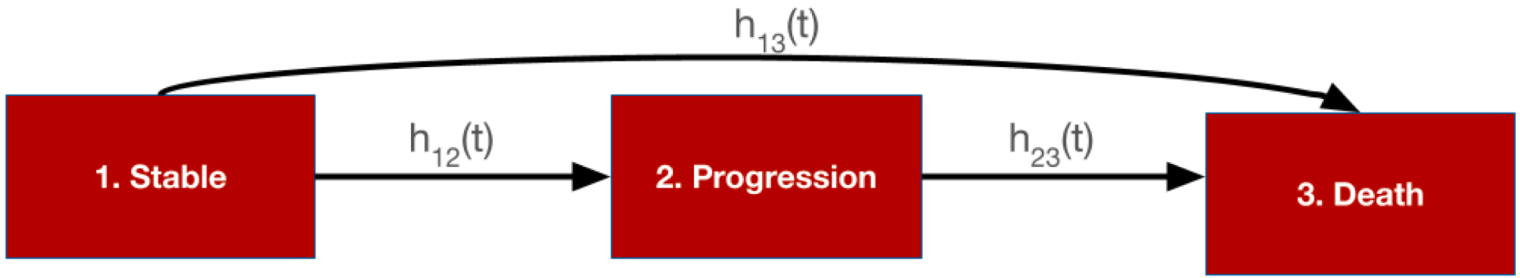
Semi-Markov multi-state model

Semi-Markov multi-state model

- Transition rates can depend on time in intermediate health states (unlike in a Markov model)
- Can only be simulated in a general manner using individual patient simulation (IPS)
- IPS is performed most efficiently using a **continuous time state transition model (CTSTM)**
- Ideally parameterizing by fitting a multi-state model

Semi-Markov multi-state model

Clock reset



Economic models with `hesim`

1. Model set-up
2. Parameters
3. Simulation
    a. Construction of model
    b. Simulation of outcomes
4. Cost-effectiveness analysis

Model setup

- The transitions of a multi-state model in `hesim` must be characterized by a matrix where each element denotes a transition from a row to a column

```
tmat <- rbind(
  c(NA, 1, 2),
  c(NA, NA, 3),
  c(NA, NA, NA)
)
colnames(tmat) <- rownames(tmat) <- c("Stable", "Progression", "Dead")

print(tmat)
```

```
            Stable Progression     Dead
Stable          NA           1        2
Progression     NA          NA        3
Dead            NA          NA       NA
```

Model setup

```
n_patients <- 1000

patients <- data.table(
  patient_id = 1:n_patients,
  age = rnorm(n_patients, mean = 45, sd = 7),
  female = rbinom(n_patients, size = 1, prob = .51)
)

strategies <- data.frame(
  strategy_id = 1:2,
  strategy_name = c("SOC", "New")
)
n_strategies <- nrow(strategies)


states <- data.table(
  state_id = c(1, 2),
  state_name = c("Stable", "Progression")
)
n_states <- nrow(states)


hesim_dat <- hesim_data(
  strategies = strategies,
  patients = patients,
  states = states
)
```

As in the cohort model, we must specify the target population and treatment strategies of interest

In an IPS we simulate many patients and then average outcomes across the simulated patients. 1,000 simulated patients should produce reasonably stable results

We also explicitly define the (non-death) health states, which we will use to model utility and costs

We always combine this information into one object

Model setup

```
print(hesim_dat)
```

```
$strategies
  strategy_id strategy_name
1           1           SOC
2           2           New

$patients
      patient_id      age female
   1:          1 42.71774      0
   2:          2 48.86723      0
   3:          3 40.27539      1
   4:          4 46.50052      1
   5:          5 47.17538      1
  ---
 996:        996 43.22279      0
 997:        997 54.22166      0
 998:        998 37.27172      0
 999:        999 45.20616      0
1000:       1000 39.15981      1

$states
   state_id  state_name
1:        1      Stable
2:        2 Progression

attr(,"class")
[1] "hesim_data"
```

Parameter estimation

- In the cohort examples, we used parameter estimates from the literature
  - We will continue to do this for utility and costs
- However, in an ideal scenario, we would estimate parameters ourselves using patient-level data
  - We will fit a multi-state model in this manner by estimating transition specific hazards using the R package `flexsurv`

Parameter estimation – Multi-state model

- Multi-state models can be fit by:
  - Estimating a joint survival model with interaction terms for different transition
  - Fitting separate survival models for each transition
    (Method used here)

Parameter estimation – Multi-state model

- Dataset

```
   from to female      age patient_id final time_start time_stop status transition_id strategy_id
1:    1  2      1 54.37365          1     0    0.00000  11.66523      1             1           2
2:    1  3      1 54.37365          1     0    0.00000  11.66523      0             2           2
3:    2  3      1 54.37365          1     1   11.66523  15.00000      0             3           2
4:    1  2      1 58.93437          2     0    0.00000  15.00000      0             1           1
5:    1  3      1 58.93437          2     1    0.00000  15.00000      0             2           1
   strategy_name      time
1:           New 11.665231
2:           New 11.665231
3:           New  3.334769
4:           SOC 15.000000
5:           SOC 15.000000
```

- Estimate parameters

```
wei_fits <- vector(length = 3, mode = "list")

for (i in 1:3){ # 3 possible transitions
  wei_fits[[i]] <- flexsurvreg(
    Surv(time, status) ~ strategy_name + female,
    data = data,
    subset = (transition_id == i) ,
    dist = "weibull")
}

wei_fits <- flexsurvreg_list(wei_fits)
```

Parameter estimation – Multi-state model

Stable -> Progression

```
[[1]]
Call:
flexsurvreg(formula = Surv(time, status) ~ strategy_name + female,
    data = data, subset = (transition_id == i), dist = "weibull")

Estimates:
                  data mean  est      L95%     U95%     se       exp(est)  L95%     U95%
shape                    NA   1.9689   1.8843   2.0572   0.0441       NA        NA       NA
scale                    NA  11.3409  10.8364  11.8688   0.2633       NA        NA       NA
strategy_nameNew     0.4930  -0.4609  -0.5147  -0.4072   0.0274   0.6307    0.5977   0.6655
female               0.4990   0.3493   0.2958   0.4028   0.0273   1.4181    1.3442   1.4961

N = 2000,  Events: 1401,  Censored: 599
Total time at risk: 16884.86
Log-likelihood = -4458.597, df = 4
AIC = 8925.194
```

New treatment increases time to progression

Shape parameter: whether the hazard is increasing (>1), decreasing (<1), or constant (=1)
Scale: whether the hazard is lower/higher at given time point

Parameter estimation – Multi-state model

Stable -> Death

```
[[2]]
Call:
flexsurvreg(formula = Surv(time, status) ~ strategy_name + female,
    data = data, subset = (transition_id == i), dist = "weibull")

Estimates:
                  data mean  est      L95%     U95%     se       exp(est)  L95%     U95%
shape                    NA   3.0233   2.7877   3.2788   0.1252       NA        NA       NA
scale                    NA  18.3770  17.1128  19.7346   0.6683       NA        NA       NA
strategy_nameNew     0.4930  -0.5162  -0.5905  -0.4419   0.0379   0.5968    0.5541   0.6428
female               0.4990   0.4104   0.3376   0.4833   0.0372   1.5074    1.4015   1.6214

N = 2000,  Events: 343,  Censored: 1657
Total time at risk: 16884.86
Log-likelihood = -1407.966, df = 4
AIC = 2823.932
```

Shape parameter: whether the hazard is increasing, decreasing, or constant
Scale: whether the hazard is lower/higher at given time point

Parameter estimation – Multi-state model

Progression -> Death

```
[[3]]
Call:
flexsurvreg(formula = Surv(time, status) ~ strategy_name + female,
    data = data, subset = (transition_id == i), dist = "weibull")

Estimates:
                  data mean  est      L95%     U95%     se       exp(est)  L95%     U95%
shape                  NA     3.6185   3.3423   3.9174   0.1466       NA        NA       NA
scale                  NA    16.1803  14.8094  17.6781   0.7309       NA        NA       NA
strategy_nameNew   0.5360    -0.5450  -0.6376  -0.4523   0.0473   0.5799    0.5285   0.6361
female             0.4675     0.3213   0.2547   0.3879   0.0340   1.3789    1.2901   1.4739

N = 1401,  Events: 377,  Censored: 1024
Total time at risk: 9736.58
Log-likelihood = -1185.006, df = 4
AIC = 2378.012
```

Shape parameter: whether the hazard is increasing, decreasing, or constant
Scale: whether the hazard is lower/higher at given time point

Parameters – Utility

```
utility_tbl <- stateval_tbl(
  data.table(state_id = states$state_id,
             mean = c(.8, .6),
             se = c(0.02, .05)
  ),
  dist = "beta",
  hesim_data = hesim_dat)
```

```
   state_id mean   se
1:        1  0.8 0.02
2:        2  0.6 0.05
```

Parameters – Medical cost

```
medcost_tbl <- stateval_tbl(
  data.table(state_id = states$state_id,
             mean = c(2000, 9500),
             se = c(2000, 9500)
  ),
  dist = "gamma",
  hesim_data = hesim_dat)
```

```
   state_id mean   se
1:        1 2000 2000
2:        2 9500 9500
```

Parameters – Drug cost

```
n_times <- 2

drugcost_tbl <- stateval_tbl(
  data.table(
    strategy_id = rep(strategies$strategy_id, each = n_states * n_times),
    state_id = rep(rep(states$state_id, each = n_strategies), n_times),
    time_start = rep(c(0, 3/12), n_states * n_strategies),
    est = c(rep(2000, 4), # Costs are always the same with SOC
            12000, 12000, 12000, 10000 # Costs with New drop after 3 months in progression state
    )
  ),
  dist = "fixed",
  hesim_data = hesim_dat)
```

When using an IPS, "state values" (like transition rates) can depend on time in an intermediate health state

We illustrate by assuming that costs for the new treatment are $12,000 for the first 3 months in the progression state and then $10,000 thereafter

(Would not be possible in a cohort model without creating tunnel states)

	strategy_id	state_id	time_id	time_start	time_stop	est
1:	1	1	1	0.00	0.25	2000
2:	1	1	2	0.25	Inf	2000
3:	1	2	1	0.00	0.25	2000
4:	1	2	2	0.25	Inf	2000
5:	2	1	1	0.00	0.25	12000
6:	2	1	2	0.25	Inf	12000
7:	2	2	1	0.00	0.25	12000
8:	2	2	2	0.25	Inf	10000

Simulation – Construct the model

Disease model

- The transition model is constructed as a function of the fitted multi-state model and *input data* (treatment strategy and patients)

```
transmod_data <- expand(hesim_dat,
                        by = c("strategies", "patients"))
```

```
   strategy_id patient_id strategy_name      age female
1:           1          1           SOC 42.71774      0
2:           1          2           SOC 48.86723      0
3:           1          3           SOC 40.27539      1
4:           1          4           SOC 46.50052      1
5:           1          5           SOC 47.17538      1
6:           1          6           SOC 53.21776      0
```

```
transmod <- create_IndivCtstmTrans(wei_fits, transmod_data,
                                   trans_mat = tmat, n = 500,
                                   clock = "reset",
                                   start_age = patients$age)
```

Simulation – Construct the model

Utility and cost models

```
# Utility
utilitymod <- create_StateVals(utility_tbl, n = 500)


# Costs
drugcostmod <- create_StateVals(drugcost_tbl, n = 500,
                                time_reset = TRUE)
medcostmod <- create_StateVals(medcost_tbl, n = 500)



costmods <- list(Drug = drugcostmod,
                 Medical = medcostmod)
```

So that costs depend on time in intermediate state

Simulation – Construct the model

Combining the disease progression, cost, and utility models

```
econmod <- IndivCtstm$new(trans_model = transmod,
                          utility_model = utilitymod,
                          cost_models = costmods)
```

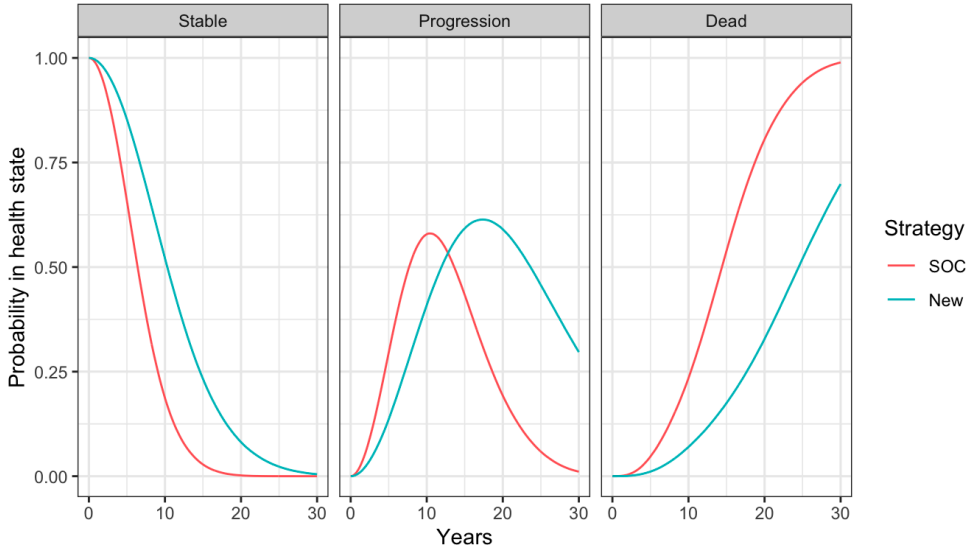
Simulation - Simulating outcomes

Disease progression

```
econmod$sim_disease(max_age = 100)

head(econmod$disprog_)
```

	sample	strategy_id	patient_id	grp_id	from	to	final	time_start	time_stop
1:	1	1	1	1	1	2	0	0.000000	10.403221
2:	1	1	1	1	2	3	1	10.403221	20.171049
3:	1	1	2	1	1	2	0	0.000000	4.759907
4:	1	1	2	1	2	3	1	4.759907	19.568465
5:	1	1	3	1	1	2	0	0.000000	5.384254
6:	1	1	3	1	2	3	1	5.384254	19.427541

```
econmod$sim_stateprobs(t = seq(0, 30 , 1/12))
```



Simulation - Simulating outcomes

QALYs and costs

```
econmod$sim_qalys(dr = c(0,.03))
```

```
   sample strategy_id grp_id state_id dr    qalys       lys
1:      1           1      1        1  0 5.518113  6.778010
2:      1           1      1        2  0 5.596111  8.399302
3:      1           2      1        1  0 9.171519 11.265563
4:      1           2      1        2  0 9.377958 14.075543
5:      2           1      1        1  0 5.573405  6.806206
6:      2           1      1        2  0 4.870313  8.003933
```

```
econmod$sim_costs(dr = 0.03)
```

```
   sample strategy_id grp_id state_id   dr category     costs
1:      1           1      1        1 0.03     Drug  11968.06
2:      1           1      1        2 0.03     Drug  11710.68
3:      1           2      1        1 0.03     Drug 110002.04
4:      1           2      1        2 0.03     Drug  78500.50
5:      2           1      1        1 0.03     Drug  11998.25
6:      2           1      1        2 0.03     Drug  11226.46
```

Cost-effectiveness analysis

```r
ce_sim <- econmod$summarize()

cea_pw_out <- cea_pw(ce_sim, comparator = 1,
                     dr_qalys = .03, dr_costs = .03)

icer_tbl(cea_pw_out, colnames = strategies$strategy_name)
```

```
                 SOC New
Incremental QALYs "-" "3.79 (3.26, 4.38)"
Incremental costs "-" "191,540 (164,466, 251,826)"
Incremental NMB   "-" "-2,280 (-60,588, 34,068)"
ICER              "-" "50,602"
Conclusion        "-" "Not cost-effective"
```

(default WTP in icer_tbl() is $50,000)

Steps with `hesim`

1. Model set-up
   - Specify the treatment strategies, target population(s), and model structure
     - `hesim_data()`
2. Parameters
   - Estimate or define the parameters of the economic model
     - `flexsurvreg_list()`, `stateval_tbl()`
3. Simulation
   a. Construction of model
      - Create an economic model—consisting of separate statistical models for disease progression, costs, and utilities—that simulate outcomes as a function of *input data* (derived from Step 1) and *parameters* (from Step 2)
      - `create_IndivCtstmTrans()`, `create_StateVals()`, `IndivCtstm$new()`

   b. Simulation of outcomes
      - Simulate outcomes (disease progression, costs, and quality-adjusted life-years (QALYs)) using the model constructed in Step 3
      - `$sim_disease()`, `$sim_stateprobs()`, `$sim_qalys()`, `$sim_costs()`
4. Cost-effectiveness analysis

Complete R script

04-mstate.R

Source on Save | Run | Source

```r
## ---- Overview ------------------------------------------------------------
## @knitr R-setup
library("rcea")
library("hesim")
library("data.table")
library("ggplot2")
library("flexsurv")

set.seed(101) # Make random number generation reproducible

## ---- Model setup ---------------------------------------------------------
## @knitr tmat
tmat <- rbind(
  c(NA, 1, 2),
  c(NA, NA, 3),
  c(NA, NA, NA)
)
colnames(tmat) <- rownames(tmat) <- c("Stable", "Progression", "Dead")
print(tmat)

## @knitr hesim_data
n_patients <- 1000
patients <- data.table(
  patient_id = 1:n_patients,
  age = rnorm(n_patients, mean = 45, sd = 7),
  female = rbinom(n_patients, size = 1, prob = .51)
)

states <- data.table(
  state_id = c(1, 2),
  state_name = c("Stable", "Progression") # Non-death health states
)
n_states <- nrow(states)

strategies <- data.frame(
  strategy_id = 1:2,
  strategy_name = c("SOC", "New")
)
n_strategies <- nrow(strategies)

hesim_dat <- hesim_data(
  strategies = strategies,
  patients = patients,
  states = states
)
```

156:1 | Exercises | R Script

Tutorial

rcea 0.0.1 Reference Tutorials

Semi-Markov Multi-state Model

2020-10-10

Source: `vignettes/04-mstate.Rmd`

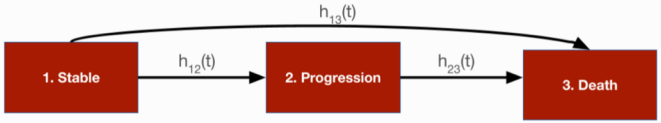
Contents

1 Overview

In this tutorial we use a continuous time state transition model (CTSTM) and relax many of the assumptions made in cohort discrete time state transition models (DTSTMs). First, since the model is in continuous time we do not require model cycles. Second, we estimate the

https://hesim-dev.github.io/rcea/articles/04-mstate.html

To illustrate, we simplify the sick-sicker model so that it only contains three health states and modify the states—*Stable*, *Progression*, and *Dead*—to mimic an oncology application where patients transition from stable disease to progression to death. There are three transitions: (1) *Stable* to *Progression*, (2) *Stable* to *Dead*, and (3) *Progression* to *Dead*.



The following packages and settings will be used for the analysis. Note that while individual-level simulations can be computationally intensive, they run very quickly in `hesim` because they are implemented fully in `C++` under the hood. You can learn more by looking at the `hesim` multi-state modeling vignette.

```
library("rcea")
library("hesim")
library("data.table")
library("ggplot2")
library("flexsurv")

set.seed(101) # Make random number generation reproducible
```

Exercise 4: Semi-Markov multi-state model

- *Review and run the R-script “04-mstate.R”*

Cost-effectiveness analysis

Cost-effectiveness analysis

- The optimal treatment strategy is the one that maximizes the expected net monetary benefit (NMB)

$$NMB(j,\theta) = e_j(\theta) \cdot k - c_j(\theta)$$

Effectiveness (e.g., QALYs) *Willingness to pay* *Costs*

j indexes a treatment strategy
θ indexes a parameter set

$$j^* = argmax_j E_\theta[\mathrm{NMB}(\mathrm{j},\theta)]$$

j is treatment with highest NMB averaged across all sets of θ*

- Can also assess optimal strategy using incremental cost-effectiveness ratio (ICER). Treatment 1 is preferred to treatment 0 if[1]

$$k > \frac{E_\theta[c_1(\theta) - c_0(\theta)]}{E_\theta[e_1(\theta) - e_0(\theta)]} = ICER$$

[1]*Only true if both incremental costs (numerator) and incremental effectiveness (denominator) are both positive. Treatment 1 dominates treatment 0 if it is more effective and less costly. Treatment 1 is dominated by* treatment 0 *if it is less effective and more costly. Treatment 1 is preferred to treatment 0 if it is less costly and less effective when k < ICER.*

Economic models with `hesim`

1. Model set-up
2. Parameters
3. Simulation
    a. Construction of model
    b. Simulation of outcomes
4. Cost-effectiveness analysis

Cost-effectiveness analysis

- CEA functions from `hesim` to summarize decision uncertainty and `ggplot2` for visualization

```
library("hesim")
library("ggplot2")
theme_set(theme_minimal()) # Set ggplot2 theme
```

- CEA can be performed using the `cea()` and `cea_pw()` functions
  - `cea()` summarizes results by taking into account each treatment strategy in the analysis
  - `cea_pw()` summarizes "pairwise" results in which each treatment is compared to a comparator

- We will use these functions to analyze the distribution of costs and QALYs produced from the simulation of the semi-Markov CTSTM (a `hesim::ce object`)

Cost-effectiveness analysis

```
ce_sim <- readRDS("ce_sim.rds")
```

```
$costs
      category   dr sample strategy_id     costs grp_id
   1:     Drug 0.03      1           1  23678.74      1
   2:     Drug 0.03      1           2 188502.54      1
   3:     Drug 0.03      2           1  23224.71      1
   4:     Drug 0.03      2           2 184885.23      1
   5:     Drug 0.03      3           1  23011.36      1
  ---
2996:    total 0.03    498           2 250188.10      1
2997:    total 0.03    499           1  63343.39      1
2998:    total 0.03    499           2 239276.32      1
2999:    total 0.03    500           1  50651.70      1
3000:    total 0.03    500           2 233785.48      1

$qalys
        dr sample strategy_id     qalys grp_id
   1: 0.00      1           1 11.114224      1
   2: 0.00      1           2 18.549477      1
   3: 0.00      2           1 10.443717      1
   4: 0.00      2           2 17.221927      1
   5: 0.00      3           1 10.404786      1
  ---
1996: 0.03    498           2 11.460201      1
1997: 0.03    499           1  8.406517      1
1998: 0.03    499           2 11.858450      1
1999: 0.03    500           1  8.063408      1
2000: 0.03    500           2 12.152292      1
```

Cost-effectiveness analysis

```
wtp <- seq(0, 250000, 500) # Willingness to pay per QALY

cea_pw_out <- cea_pw(ce_sim, comparator = 1, # Comparator is SOC (ID = 1)
                     dr_qalys = 0.03, dr_costs = 0.03,
                     wtp)

cea_out <- cea(ce_sim,
               dr_qalys = 0.03, dr_costs = 0.03,
               k = wtp)
```

Cost-effectiveness plane

```
cea_pw_out$delta
```

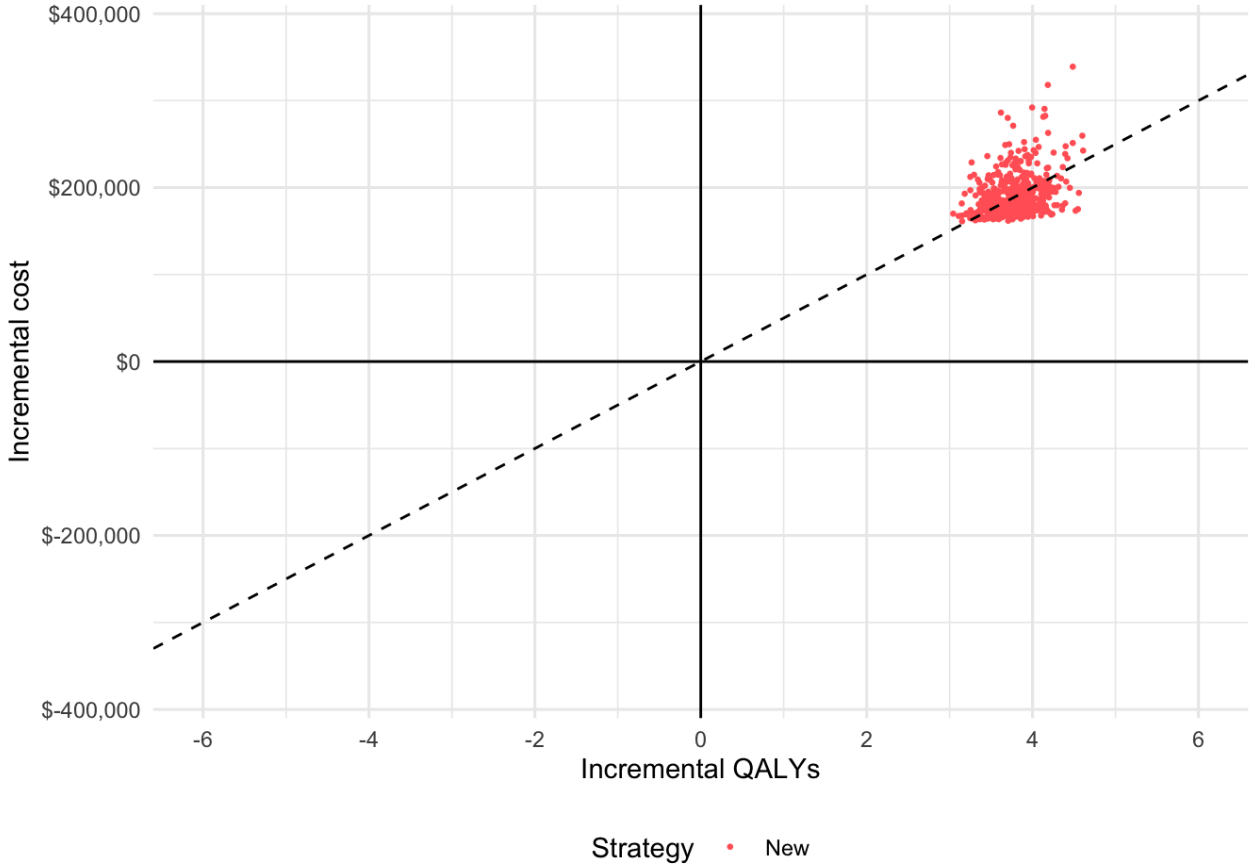
```
      sample strategy_id grp_id       ie       ic
  1:       1           2      1 3.900703 173752.1
  2:       2           2      1 3.700907 280151.1
  3:       3           2      1 4.140923 207748.0
  4:       4           2      1 3.765042 202362.6
  5:       5           2      1 4.003303 188159.4
 ---
496:     496           2      1 3.635187 178517.2
497:     497           2      1 3.903630 169764.4
498:     498           2      1 3.539451 180691.1
499:     499           2      1 3.451933 175932.9
500:     500           2      1 4.088884 183133.8
```

Cost-effectiveness plane

```
strategy_factor <- function (x) {
  factor(x, levels = 1:2, labels = c("SOC", "New"))
}
format_dollar <- function(x) {
  paste0("$", formatC(x, format = "d", big.mark =
","))
}
```

```
ylim <- max(cea_pw_out$delta[, ic]) * 1.1
xlim <- ceiling(max(cea_pw_out$delta[, ie]) * 1.1)

ggplot(cea_pw_out$delta,
       aes(x = ie, y = ic, col = strategy_factor(strategy_id))) +
  geom_jitter(size = .5)  +
  xlab("Incremental QALYs") +
  ylab("Incremental cost") +
  scale_y_continuous(limits = c(-ylim, ylim),
                     labels = format_dollar) +
  scale_x_continuous(limits = c(-xlim, xlim), breaks = seq(-6, 6, 2)) +
  theme(legend.position = "bottom") +
  scale_colour_discrete(name = "Strategy") +
  geom_abline(slope = 50000, linetype = "dashed") +
  geom_hline(yintercept = 0) +
  geom_vline(xintercept = 0)
```

Cost-effectiveness plane



Cost-effectiveness acceptability curve – Simultaneous comparison

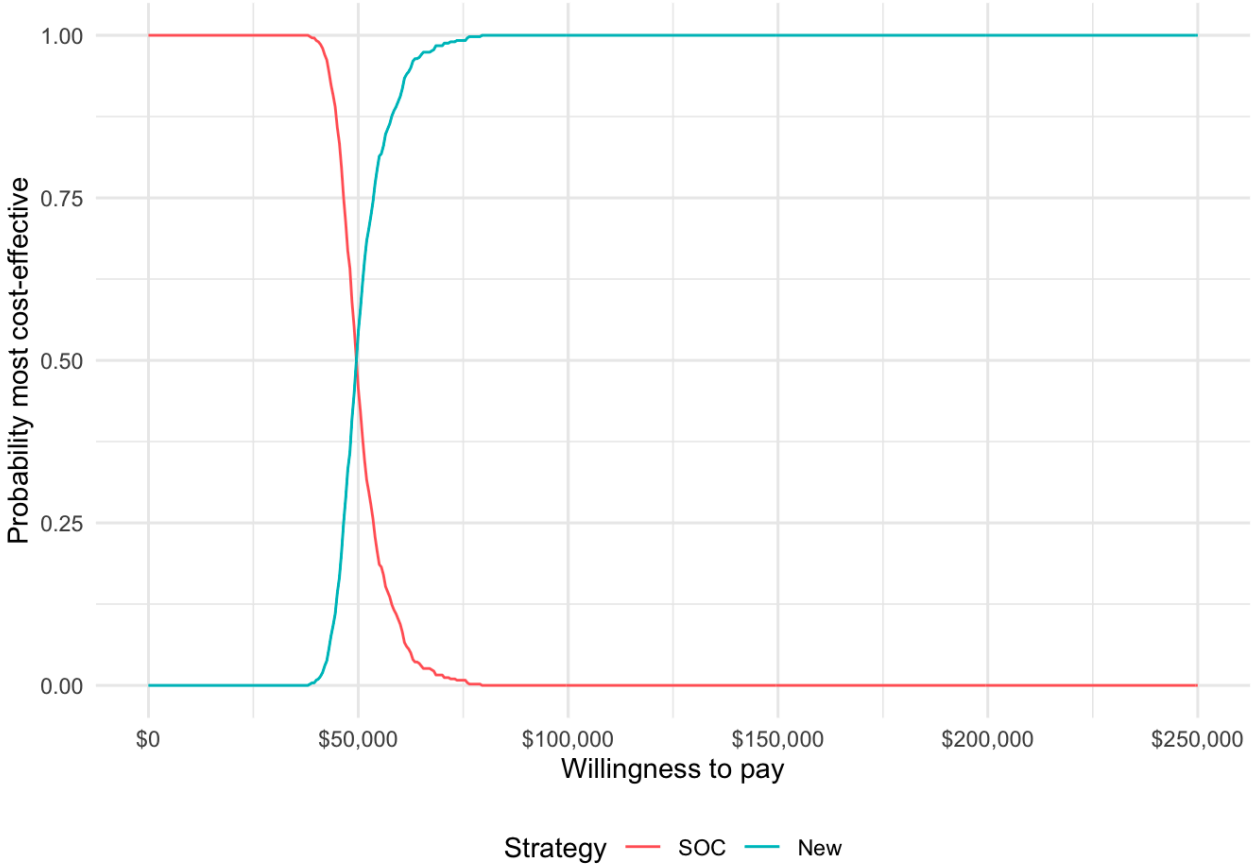
```
cea_out$mce
```

```
          k strategy_id grp_id best prob
   1:     0           1      1    1    1
   2:     0           2      1    0    0
   3:   500           1      1    1    1
   4:   500           2      1    0    0
   5:  1000           1      1    1    1
  ---
 998: 249000           2      1    1    1
 999: 249500           1      1    0    0
1000: 249500           2      1    1    1
1001: 250000           1      1    0    0
1002: 250000           2      1    1    1
```

Cost-effectiveness acceptability curve – Simultaneous comparison

```
ggplot(cea_out$mce,
       aes(x = k, y = prob, col = strategy_factor(strategy_id))) +
  geom_line() +
  xlab("Willingness to pay") +
  ylab("Probability most cost-effective") +
  scale_x_continuous(breaks = seq(0, max(wtp), length.out = 6),
                     label = format_dollar) +
  theme(legend.position = "bottom") +
  scale_colour_discrete(name = "Strategy")
```

Cost-effectiveness acceptability curve – Simultaneous comparison



Value of perfect information

- The expected value of perfect information (EVPI) combines the probability of being most effective with the *magnitude* of the expected NMB
- Intuitively, EVPI is the amount that a decision maker would be willing to pay to collect additional data and completely eliminate uncertainty
- Mathematically, the EVPI is defined as the difference between the maximum expected NMB given perfect information and the maximum expected NMB given current information

$$EVPI = E_\theta[\max_j NMB(j,\theta)] - \max_j E_\theta[NMB(j,\theta)]$$

NMB for optimal treatment at each random draw of the parameters

NMB of treatment that is optimal when averaged across all parameter draws

Value of perfect information

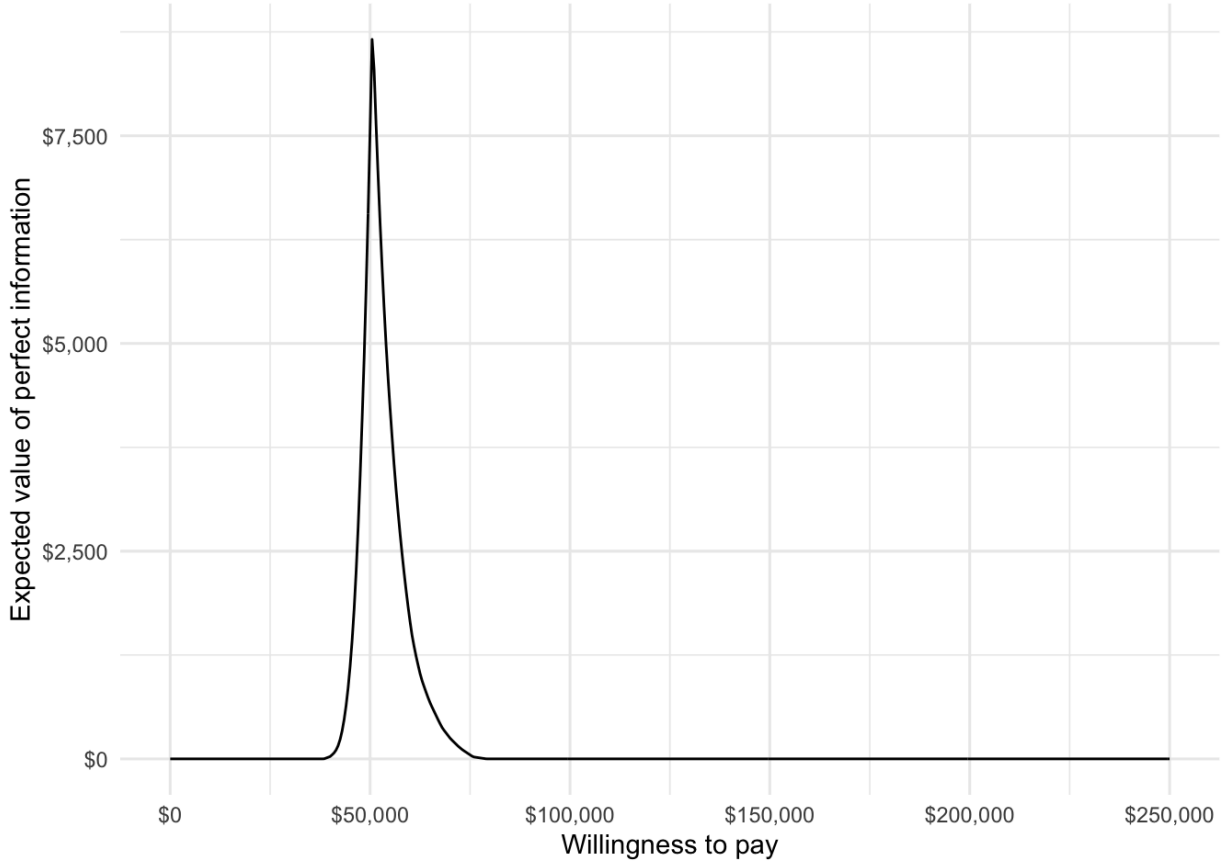
```
cea_out$evpi
```

```
     grp_id      k best     enmbci     enmbpi          evpi
  1:      1      0    1 -115243.22 -115243.22  7.275958e-11
  2:      1    500    1 -110620.84 -110620.84 -4.365575e-11
  3:      1   1000    1 -105998.46 -105998.46  4.365575e-11
  4:      1   1500    1 -101376.08 -101376.08  0.000000e+00
  5:      1   2000    1  -96753.71  -96753.71  0.000000e+00
 ---
497:      1 248000    2 3297781.34 3297781.34  9.313226e-10
498:      1 248500    2 3305188.05 3305188.05 -4.656613e-10
499:      1 249000    2 3312594.76 3312594.76  0.000000e+00
500:      1 249500    2 3320001.46 3320001.46 -4.656613e-10
501:      1 250000    2 3327408.17 3327408.17 -9.313226e-10
```

Value of perfect information

```
ggplot(cea_out$evpi, aes(x = k, y = evpi)) +
  geom_line()  +
  xlab("Willingness to pay") +
  ylab("Expected value of perfect information") +
  scale_x_continuous(breaks = seq(0, max(wtp), length.out = 6),
                     label = format_dollar) +
  scale_y_continuous(label = format_dollar) +
  theme(legend.position = "bottom")
```

Value of perfect information



Complete R script

06-cea.R

Source on Save | Run | Source

```r
## ---- Overview ------------------------------------------------------------
## @knitr R-setup
library("hesim")
library("ggplot2")
theme_set(theme_minimal()) # Set ggplot2 theme

## ---- Application ---------------------------------------------------------
## @knitr conduct-cea
ce_sim <- readRDS("ce_sim.rds") # Load cost-effectiveness object
head(ce_sim)

wtp <- seq(0, 250000, 500) # Willingness to pay per QALY
cea_pw_out <- cea_pw(ce_sim, comparator = 1, # Comparator is SOC (ID = 1)
                     dr_qalys = 0.03, dr_costs = 0.03,
                     wtp)
cea_out <- cea(ce_sim,
               dr_qalys = 0.03, dr_costs = 0.03,
               k = wtp)

## ---- Cost-effectiveness plane --------------------------------------------
## @knitr helper-functions
strategy_factor <- function (x) {
  factor(x, levels = 1:2, labels = c("SOC", "New"))
}

format_dollar <- function(x) {
  paste0("$", formatC(x, format = "d", big.mark = ","))
}

## @knitr ceplane-plot
ylim <- max(cea_pw_out$delta[, ic]) * 1.1
xlim <- ceiling(max(cea_pw_out$delta[, ie]) * 1.1)
ggplot(cea_pw_out$delta,
       aes(x = ie, y = ic, col = strategy_factor(strategy_id))) +
  geom_jitter(size = .5)  +
  xlab("Incremental QALYs") +
```

13:1 | Application | R Script

Tutorial

rcea 0.0.1 Reference Tutorials

Cost-effectiveness Analysis

2020-10-10

Source: `vignettes/06-cea.Rmd`

Contents

https://hesim-dev.github.io/rcea/articles/06-cea.html

1 Overview

The prior tutorials have focused on constructing economic models to simulate disease progression, costs, and quality-adjusted life-years (QALYs). While incremental cost-effectiveness ratios (ICERs) have been computed and probabilistic sensitivity analysis (PSA) has been employed, we have not yet formalized cost-effectiveness analysis (CEA) or represented decision uncertainty

subgroups.

```
library("hesim")
library("ggplot2")
theme_set(theme_minimal()) # Set ggplot2 theme
```

2 Theory

CEA is based on estimating the net monetary benefit (NMB). For a given parameter set θ, the NMB with treatment j is computed as the difference between the monetized health gains from an intervention less costs, or,

$$NMB(j, \theta) = e_j(\theta) \cdot k - c_j(\theta),$$

where e_j and c_j are measures of health outcomes (e.g. QALYs) and costs using treatment j respectively, and k is a decision makers willingness to pay (WTP) per unit of health outcomes. The optimal treatment is the one that maximizes the expected NMB,

$$j^* = \text{argmax}_j E_\theta [NMB(j, \theta)].$$

For a pairwise comparison, treatment 1 is preferred to treatment 0 if the expected incremental net monetary benefit (INMB) is positive; that is, if $E_\theta [INMB] > 0$ where the INMB is given by

$$INMB(\theta) = NMB(j = 1, \theta) - NMB(j = 0, \theta).$$

Exercise 5: Cost-effectiveness analysis

- *Review and run the R-script “06-cea.R”*

Summary

So why R for CE modeling?

- One platform to do everything
  - parameter estimation
  - simulation
- More complex analysis and individual patient simulation
- Your problems are rarely unique
  - But even if they are, R facilitates development of custom models
- Easier to share and review
- Reproducible

Cost-effectiveness analysis with R

- BCEA
- heemod
- hesim
- …

Why `hesim`?

- Focus on setting up and interpreting a model rather than implementation
  - Burdensome programming tasks have already been implemented and optimized
- Flexible enough to cover many problems
  - May need to write custom code for very complex or unique problems (beyond scope of course)
- Very fast!
  - IPS in `hesim` with 100 PSA iterations ran in .44 seconds; equivalent simulation in `mstate` package took 34 minutes (see here)
  - Cohort model in `hesim` with 1,000 PSA iterations ran in 1 second (IPS in 9 seconds), while equivalent cohort model in `heemod` took 85 seconds (see here)

User interfaces with R Shiny

LESSON 1

Welcome to Shiny

Shiny is an R package that makes it easy to build interactive web applications (apps) straight from R. This lesson will get you started building Shiny apps right away.

```
install.packages("shiny")
```

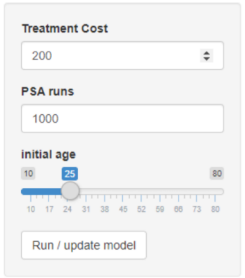
Examples

https://shiny.rstudio.com/

Making Markov Models Shiny: A Tutorial

Robert Smith and Paul Schneider, ScHARR, University of Sheffield

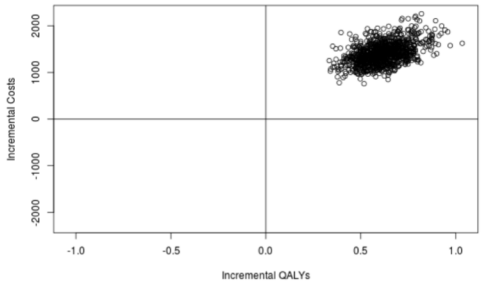
Sick Sicker Model in Shiny



Results Table

Option	QALYs	Costs	Inc.QALYs	Inc.Costs	ICER
Treatment	18.59	100441.67	0.62	1406.24	2324.54
No Treatment	17.97	99035.43	NA	NA	NA

Cost-effectiveness Plane



https://r-hta.org/tutorial/markov_models_shiny/

Restore defaults

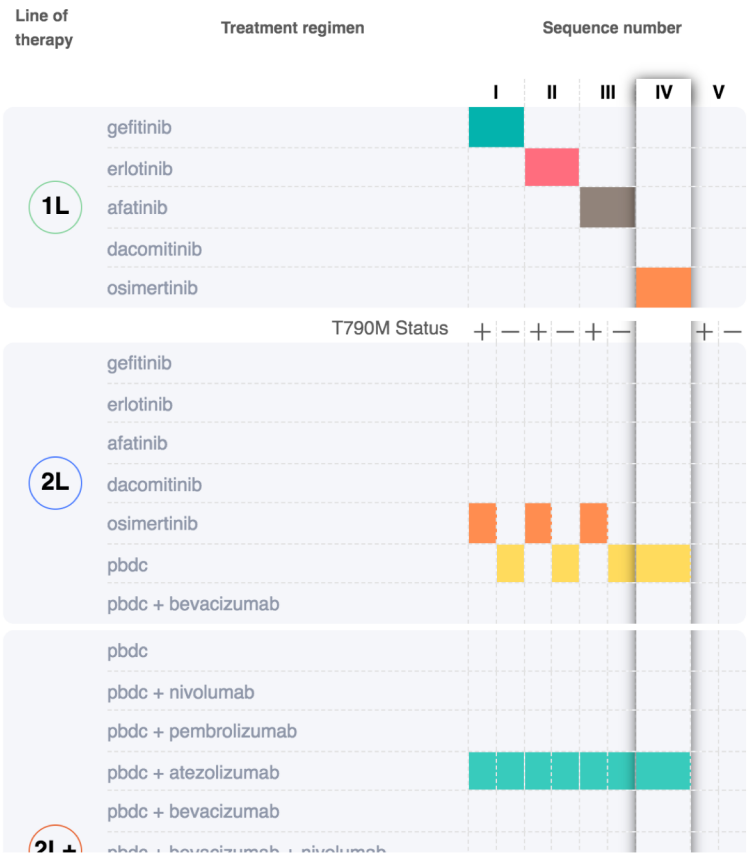
Value of a QALY $ 150,000

Save

Run simulation

Treatment sequences

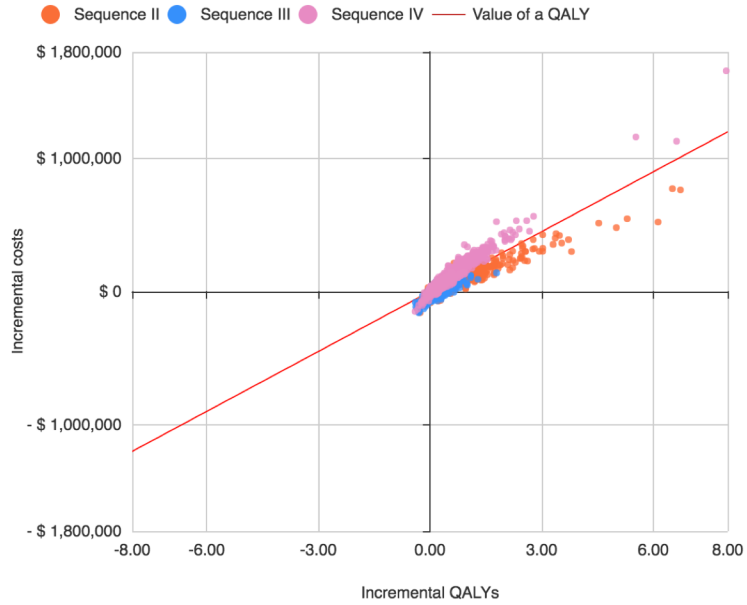
Show survival curves



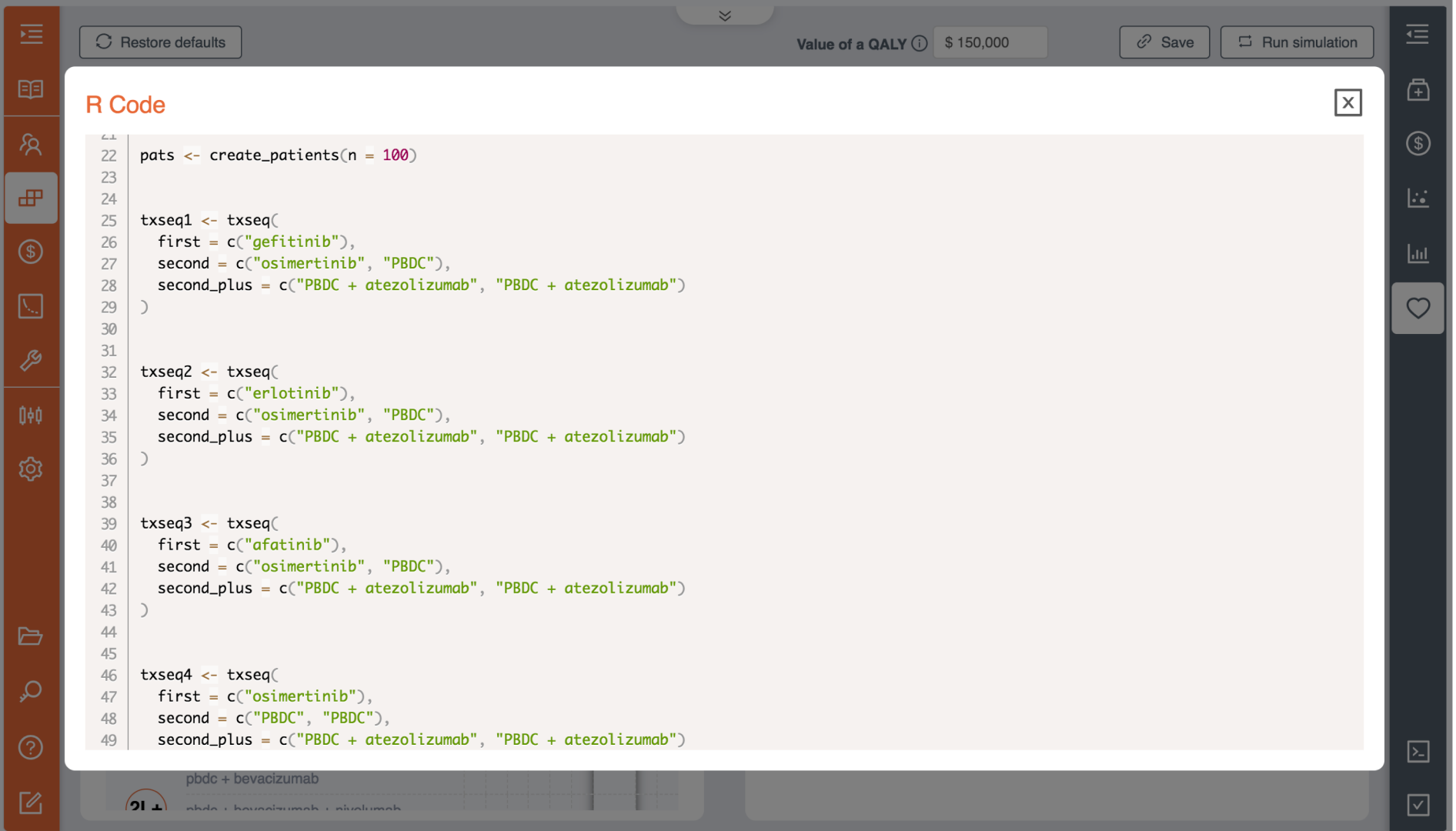
Cost-effectiveness plane

Sequence-comparator I II III IV

Expected outcomes | ICER | CE PLANE | CEAC



Expected value of perfect information


Restore defaults
Value of a QALY
$ 150,000
Save
Run simulation
R Code
pats <- create_patients(n = 100)
txseq1 <- txseq(
first = c("gefitinib"),
second = c("osimertinib", "PBDC"),
second_plus = c("PBDC + atezolizumab", "PBDC + atezolizumab")
)
txseq2 <- txseq(
first = c("erlotinib"),
second = c("osimertinib", "PBDC"),
second_plus = c("PBDC + atezolizumab", "PBDC + atezolizumab")
)
txseq3 <- txseq(
first = c("afatinib"),
second = c("osimertinib", "PBDC"),
second_plus = c("PBDC + atezolizumab", "PBDC + atezolizumab")
)
txseq4 <- txseq(
first = c("osimertinib"),
second = c("PBDC", "PBDC"),
second_plus = c("PBDC + atezolizumab", "PBDC + atezolizumab")
pbdc + bevacizumab

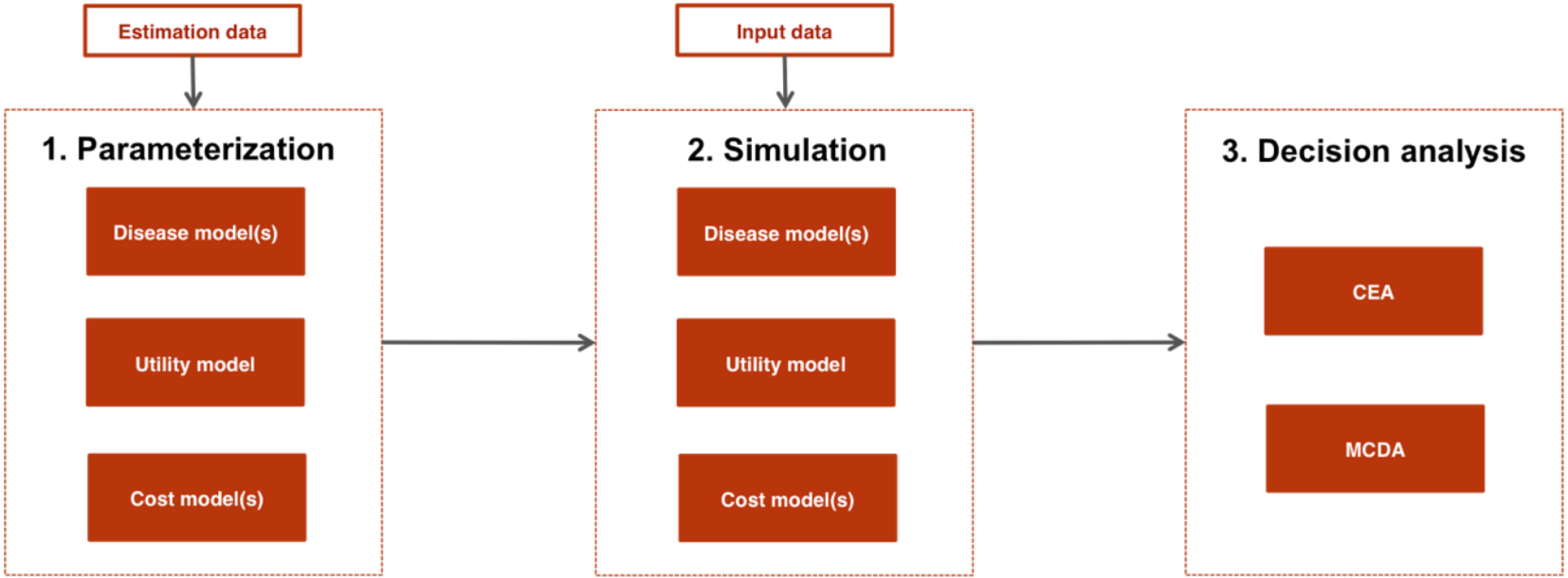
Thank you

incerti.devin@gene.com

jeroen.jansen@ucsf.edu

Appendix – Building a model with `hesim`

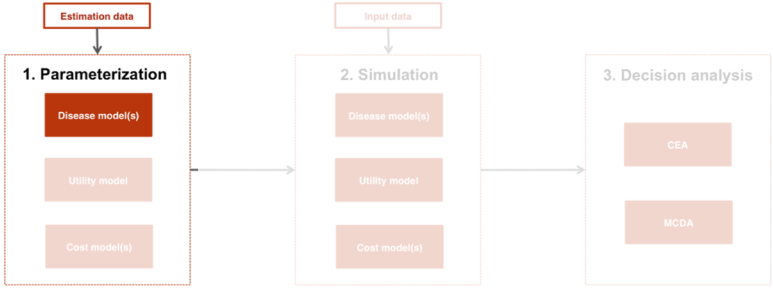
hesim


Estimation data
Input data
1. Parameterization
Disease model(s)
Utility model
Cost model(s)
2. Simulation
Disease model(s)
Utility model
Cost model(s)
3. Decision analysis
CEA
MCDA

`hesim` overview

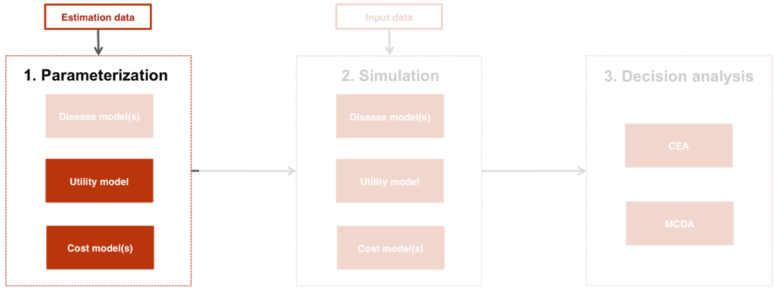
Economic model	*Object*	*Function*
Cohort discrete time state transition model (cDTSTM)	`CohortDtstm`	`create_CohortDtstm()` `CohortDtstm$new()`
Individual-level continuous time state transition model (iCTSTM)	`IndivCtstm`	`IndivCtstm$new()`
N-state partitioned survival model (PSM)	`Psm`	`Psm$new()`

hesim - Parameterization



Economic model		Disease progression		
		Statistical model	*Parameter Object*	*Model fit Function*
cDTSTM	`CohortDtstm`	Custom	`tparams_transprobs`	`define_model()`
		Multinomial logistic regression	`params_mlogit_list`	`multinom_list()`
iCTSTM	`IndivCtstm`	Multi-state model (joint likelihood)	`params_surv`	`flexsurv::flexsurvreg()`
		Multi-state model (transition-specific)	`params_surv_list`	`flexsurvreg_list()`
PSM	`Psm`	Independent survival models	`params_surv_list`	`flexsurvreg_list()`

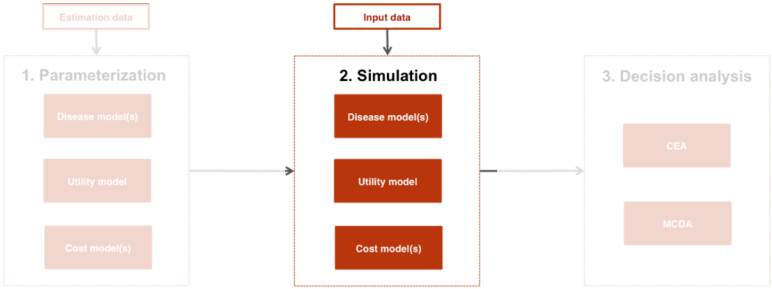
`hesim` - Parameterization



Cost and utility		
Statistical model	*Parameter Object*	*Model fit Function*
Predicted means	`tparams_mean`	`stateval_tbl()`
	`tparams_mean`	`define_model()`
Linear model	`params_lm`	`Stats::lm()`

hesim - Simulation

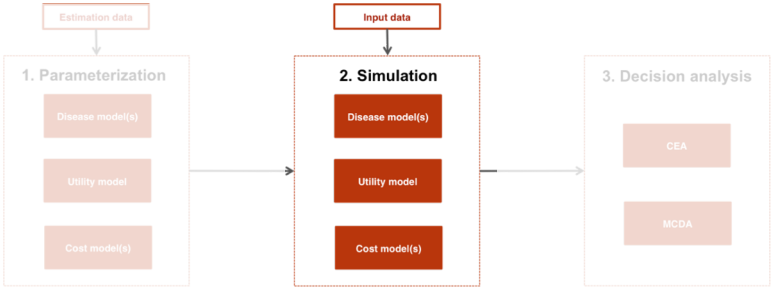
Constructing an economic model



Economic model		Disease model	Utility model	Cost model(s)
cDTSTM	`CohortDtstm`	`CohortDtstmTrans` `create_CohortDtstmTrans()`	`StateVals` `create_StateVals()`	`StateVals` `create_StateVals()`
iCTSTM	`IndivCtstm`	`IndivCtstmTrans` `create_IndivCtstmTrans()`	`StateVals` `create_StateVals()`	`StateVals` `create_StateVals()`
PSM	`Psm`	`PsmCurves` `create_PsmCurves()`	`StateVals` `create_StateVals()`	`StateVals` `create_StateVals()`

`hesim` - Simulation

Simulating outcomes



Economic model		Disease progression	QALYs	Costs
cDTSTM	`CohortDtstm`	`$sim_stateprobs()`	`$sim_qalys()`	`$sim_costs()`
iCTSTM	`IndivCtstm`	`$sim_disease()` `$sim_stateprobs()`	`$sim_qalys()`	`$sim_costs()`
PSM	`Psm`	`$sim_survival()` `$sim_stateprobs()`	`$sim_qalys()`	`$sim_costs()`