

APPLIED COST-EFFECTIVENESS MODELING WITH R

2022

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
- Partitioned-survival model
- Cost-effectiveness analysis
- Summary

Structure

- Presentation
- Exercises using R

Online tutorial

The screenshot shows the GitHub repository page for `rcea`. The top navigation bar includes tabs for `rcea` (version 0.1.2), Reference, Tutorials (which is currently selected and has a dropdown arrow), Slides, and a GitHub icon. The main content area features a sidebar with the repository name `rcea` and a brief description: "This is the repository for the `rcea` package. A range of models are covered including simple Markov cohort models, semi-Markov individual patient data models, and semi-Markov individual patient survival models. Sensitivity analysis can be used with both base R and the R package `hesim`". Below this, a list of tutorial topics is shown: Simple Markov Cohort Model, Incorporating Probabilistic Sensitivity Analysis, Markov Cohort Model with `hesim`, Semi-Markov Multi-state Model, Partitioned Survival Model, and Cost-effectiveness Analysis. The `Markov Cohort Model with hesim` topic is expanded, showing its detailed description: "A model-based cost-effectiveness analysis (CEA) with R. A range of models are covered including simple Markov cohort models, semi-Markov individual patient data models, and semi-Markov individual patient survival models. Sensitivity analysis can be used with both base R and the R package `hesim`".

The course materials are available at <https://hesim-dev.github.io/rcea>.

Installation and setup

All required R packages and course materials can be installed with the following steps.

1. Open an R session. We recommend using [RStudio](#).
2. Install the `rcea` package from GitHub, which will also install all other required packages.

```
# install.packages("devtools") # You must install the "devtools" R package first.
devtools::install_github("hesim-dev/rcea")
```

3. Create a new project in your desired directory.

```
# Create a project named "rcea-exercises" within a directory named "Projects"
usethis::create_project("~/Projects/rcea-exercises")
```

4. Add the course materials (R scripts for the tutorials) to your new project.

```
rcea::use_rcea("~/Projects/rcea-exercises")
```

Links

Browse source code at
<https://github.com/hesim-dev/rcea/>

Report a bug at
<https://github.com/hesim-dev/rcea/issues>

License

GPL-3

Developers

Devin Incerti
Author, maintainer

Jeroen P. Jansen
Author

R scripts for exercises

The screenshot shows a Mac OS X Finder window titled "Projects". The sidebar on the left lists "Favorites" including Google, Box, Macintosh HD, Applications, Desktop, Downloads, and Jero... (with a triangle icon). The main pane displays a file list with the following structure:

Name	Date Modified	Size	Kind
▼ rcea-exercises	Today at 4:30 PM	--	Folder
01-markov-cohort.R	Today at 4:30 PM	4 KB	R Source File
02-markov-cohort-psa.R	Today at 4:30 PM	6 KB	R Source File
03-markov-cohort-hesim.R	Today at 4:30 PM	4 KB	R Source File
04-mstate.R	Today at 4:30 PM	5 KB	R Source File
05-psm.R	Today at 4:30 PM	4 KB	R Source File
06-cea.R	Today at 4:30 PM	5 KB	R Source File
► R	Today at 4:29 PM	--	Folder
rcea-exercises.Rproj	Today at 4:29 PM	276 bytes	R Project

RStudio Cloud

Your Workspace / RCEA

RAM    Jeroen Jansen

File Edit Code View Plots Session Build Debug Profile Tools Help

   Go to file/function 

R 4.1.0

01-markov-cohort.R

02-markov-cohort-psa.R

03-markov-cohort-hesim.R

04-mstate.R

05-psm.R

06-cea.R

Run Source

```
1 ## ---- Overview
2 ## @knitr R-packages
3 library("rcea")
4 library("knitr")
5 library("kableExtra")
6 library("magrittr")
7 library("tibble")
8
9 ## ---- Model parameters -----
10 ## @knitr transition-probabilities
11 p_hd <- .002 # constant probability of dying when Healthy (all-cause mortality)
12 p_hs1 <- .15 # probability of becoming Sick when Healthy
13 p_s1h <- .05 # probability of becoming Healthy when Sick
14 p_s1s2 <- .105 # probability of becoming Sicker when Sick
15 p_s1d <- .006 # constant probability of dying when Sick
16 p_s2d <- .02 # constant probability of dying when Sicker
17
18 ## @knitr transition-probability-complements
19 p_hh <- 1 - p_hs1 - p_hd
20 p_s1s1 <- 1 - p_s1h - p_s1s2 - p_s1d
21 p_s2s2 <- 1 - p_s2d
```

1:1  Overview

Environment History Connections Tutorial

    293 MiB 

R Global Environment

Environment is empty

Files Plots Packages Help Viewer

 Home > Projects > rcea-exercises 

Name	Size
 ..	
 .gitignore	12 B
 R	
 rcea-exercises.Rproj	276 B
 01-markov-cohort.R	4.1 KB
 02-markov-cohort-psa.R	5.5 KB
 03-markov-cohort-hesim.R	3.4 KB
 04-mstate.R	4.5 KB
 05-psm.R	4 KB
 06-cea.R	4.7 KB

Console Terminal Jobs

R 4.1.0 · ~/Projects/rcea-exercises/

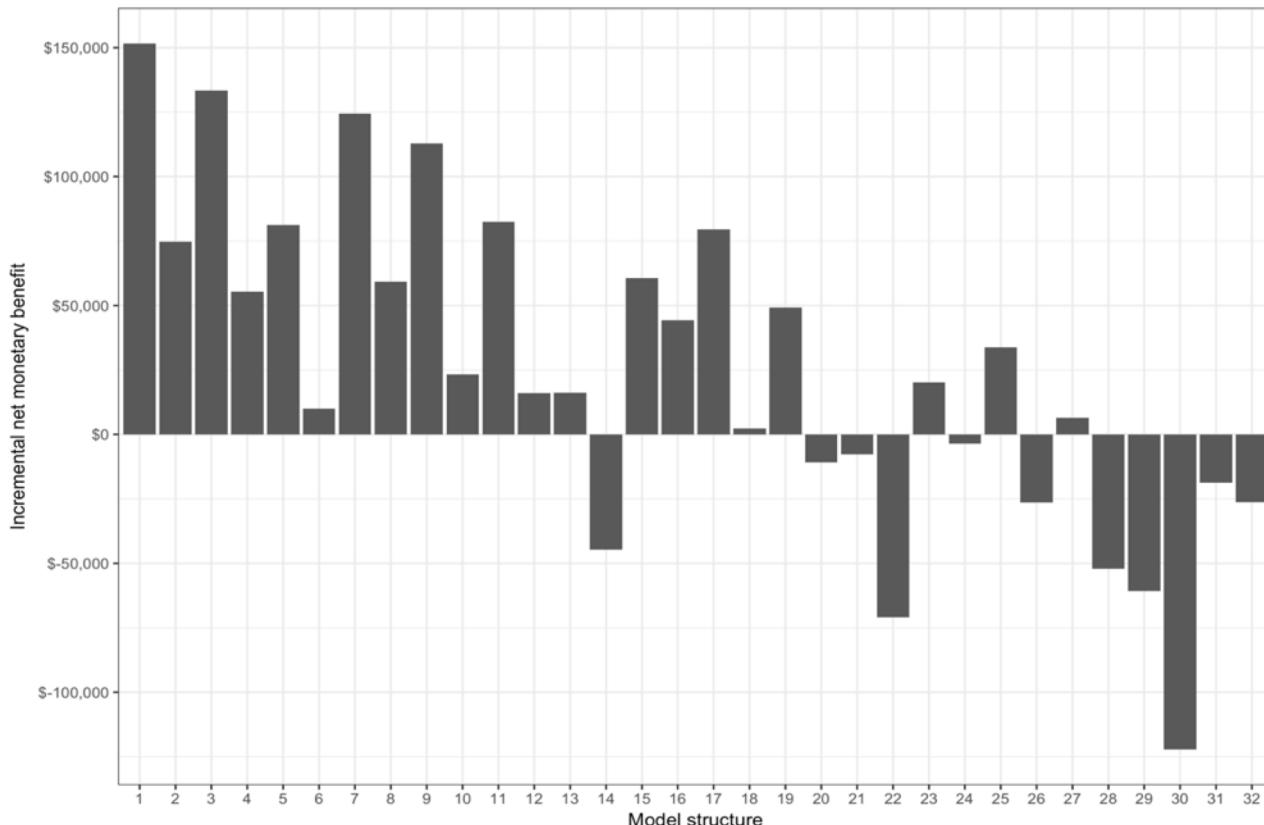
>

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

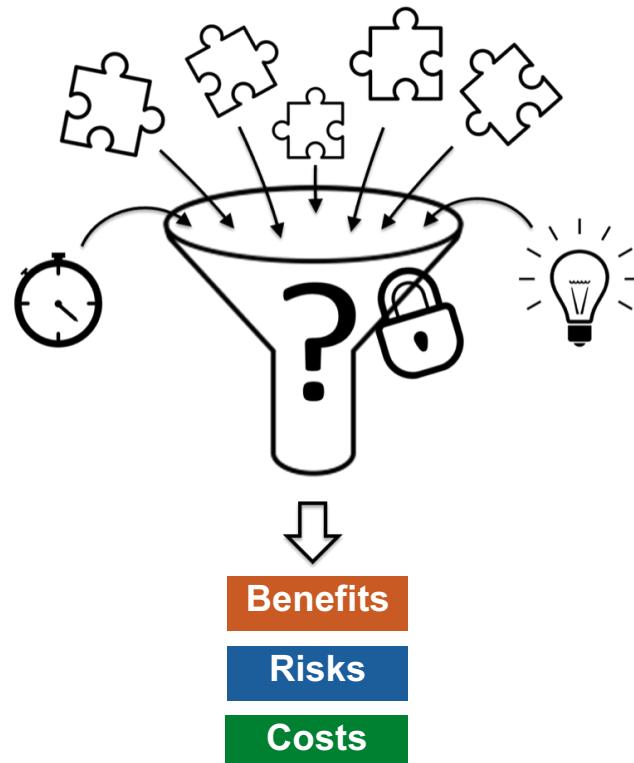


Plugging in model input parameter estimates



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

The screenshot shows the Microsoft Word ribbon with the 'Home' tab selected. The ribbon includes sections for Paragraph, Font, Styles, Alignment, Numbering, Conditional Formatting, Format as Table, Cell Styles, Insert, Delete, and Format. On the far right, there are buttons for AutoSum, Fill, Clear, Sort & Filter, and a filter icon.

128

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
PSA INPUTS																					
Efficacy and transitions between health states (other than death)		point estimate	SD(point 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	3.953	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	4.118	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	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	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	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	11.640	log-normal				5.2710	5.2710					2.60	11.64					
PROBABILITY sustained response Standard non-biologic treatment	0.829	0.016	0.797	0.858	beta	489.00	191.18		0.8459	0.8286					0.80	0.86					
OR sustained response Adalimumab 160/80/40	1.311	0.669	2.590	2.590	log-normal				1.9203	1.3110					0.67	2.59					
OR sustained response Golimumab 200/100/50(100)	1.623	0.667	2.502	2.502	log-normal				1.9808	1.6230					1.07	2.50					
OR sustained response Infliximab 5mg/kg	2.116	1.021	4.540	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	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	2.452	log-normal				1.1049	0.9180					0.36	2.45					
OR sustained remission Infliximab 5mg/kg	1.300	0.435	4.053	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	#NUM!	#NUM!	log-normal				#NUM!	0.0000					0.00	0.00					
OR maintenance discontinuation Golimumab 200/100/50(100)	0.000	0.000	#NUM!	#NUM!	log-normal				#NUM!	0.0000					0.00	0.00					
OR maintenance discontinuation Infliximab 5mg/kg	0.000	0.000	#NUM!	#NUM!	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	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	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	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	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	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	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	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	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	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	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	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	1.000	log-normal				1.0000	1.0000					1.00	1.00					
PROBABILITY Adverse event 5 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 5 Adalimumab 160/80/40 vs Standard non-biologic treatment	1.000	1.000	1.000	1.000	log-normal				1.0000	1.0000					1.00	1.00					
OR Adverse event 5 Golimumab 200/100/50(100) vs Standard non-biologic treatment	1.000	1.000	1.000	1.000	log-normal				1.0000	1.0000					1.00	1.00					
OR Adverse event 5 Infliximab 5mg/kg vs Standard non-biologic treatment	1.000	1.000	1.000	1.000	log-normal				1.0000	1.0000					1.00	1.00					

Cut Copy Paste Format Arial 9 A A Wrap Text Number Conditional Formatting Merge & Center \$ % .00 Insert Delete Format Clear Sort & Filter

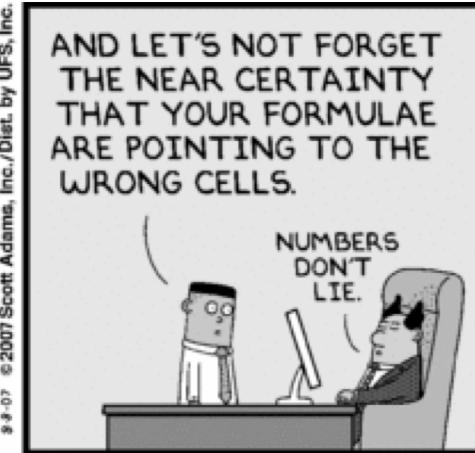
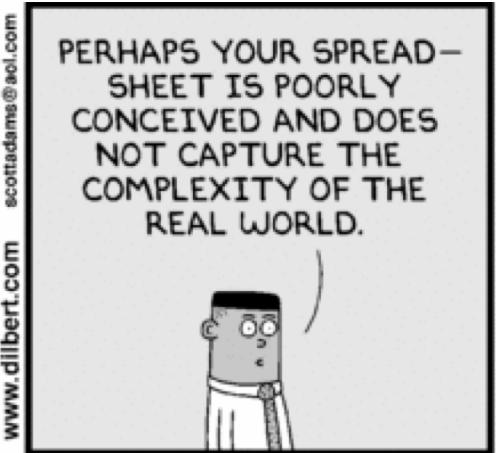
Q12 fx {=MMULT(N11:T11,\$DN\$7:\$DT\$13)*(1-H12)}

	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	X	Z	A	AB	AC	AD	AO	All	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20	A21	A22	A23	A24	A25	A26	A27	A28	A29	A30	A31	A32	A33	A34	A35	A36	A37	A38	A39	A40	A41	A42	A43	A44	A45	A46	A47	A48	A49	A50	A51	A52	A53	A54	A55	A56	A57	A58	A59	A60	A61	A62	A63	A64	A65	A66	A67	A68	A69	A70	A71	A72	A73	A74	A75	A76	A77	A78	A79	A80	A81	A82	A83	A84	A85	A86	A87	A88	A89	A90	A91	A92	A93	A94	A95	A96	A97	A98	A99	A100	A101	A102	A103	A104	A105	A106	A107	A108	A109	A110	A111	A112	A113	A114	A115	A116	A117	A118	A119	A120	A121	A122	A123	A124	A125	A126	A127	A128	A129	A130	A131	A132	A133	A134	A135	A136	A137	A138	A139	A140	A141	A142	A143	A144	A145	A146	A147	A148	A149	A150	A151	A152	A153	A154	A155	A156	A157	A158	A159	A160	A161	A162	A163	A164	A165	A166	A167	A168	A169	A170	A171	A172	A173	A174	A175	A176	A177	A178	A179	A180	A181	A182	A183	A184	A185	A186	A187	A188	A189	A190	A191	A192	A193	A194	A195	A196	A197	A198	A199	A200	A201	A202	A203	A204	A205	A206	A207	A208	A209	A210	A211	A212	A213	A214	A215	A216	A217	A218	A219	A220	A221	A222	A223	A224	A225	A226	A227	A228	A229	A230	A231	A232	A233	A234	A235	A236	A237	A238	A239	A240	A241	A242	A243	A244	A245	A246	A247	A248	A249	A250	A251	A252	A253	A254	A255	A256	A257	A258	A259	A260	A261	A262	A263	A264	A265	A266	A267	A268	A269	A270	A271	A272	A273	A274	A275	A276	A277	A278	A279	A280	A281	A282	A283	A284	A285	A286	A287	A288	A289	A290	A291	A292	A293	A294	A295	A296	A297	A298	A299	A299	A300	A301	A302	A303	A304	A305	A306	A307	A308	A309	A310	A311	A312	A313	A314	A315	A316	A317	A318	A319	A320	A321	A322	A323	A324	A325	A326	A327	A328	A329	A330	A331	A332	A333	A334	A335	A336	A337	A338	A339	A340	A341	A342	A343	A344	A345	A346	A347	A348	A349	A350	A351	A352	A353	A354	A355	A356	A357	A358	A359	A360	A361	A362	A363	A364	A365	A366	A367	A368	A369	A370	A371	A372	A373	A374	A375	A376	A377	A378	A379	A380	A381	A382	A383	A384	A385	A386	A387	A388	A389	A390	A391	A392	A393	A394	A395	A396	A397	A398	A399	A399	A400	A401	A402	A403	A404	A405	A406	A407	A408	A409	A410	A411	A412	A413	A414	A415	A416	A417	A418	A419	A420	A421	A422	A423	A424	A425	A426	A427	A428	A429	A430	A431	A432	A433	A434	A435	A436	A437	A438	A439	A440	A441	A442	A443	A444	A445	A446	A447	A448	A449	A450	A451	A452	A453	A454	A455	A456	A457	A458	A459	A460	A461	A462	A463	A464	A465	A466	A467	A468	A469	A470	A471	A472	A473	A474	A475	A476	A477	A478	A479	A480	A481	A482	A483	A484	A485	A486	A487	A488	A489	A490	A491	A492	A493	A494	A495	A496	A497	A498	A499	A499	A500	A501	A502	A503	A504	A505	A506	A507	A508	A509	A510	A511	A512	A513	A514	A515	A516	A517	A518	A519	A520	A521	A522	A523	A524	A525	A526	A527	A528	A529	A530	A531	A532	A533	A534	A535	A536	A537	A538	A539	A540	A541	A542	A543	A544	A545	A546	A547	A548	A549	A550	A551	A552	A553	A554	A555	A556	A557	A558	A559	A560	A561	A562	A563	A564	A565	A566	A567	A568	A569	A570	A571	A572	A573	A574	A575	A576	A577	A578	A579	A580	A581	A582	A583	A584	A585	A586	A587	A588	A589	A590	A591	A592	A593	A594	A595	A596	A597	A598	A599	A599	A600	A601	A602	A603	A604	A605	A606	A607	A608	A609	A610	A611	A612	A613	A614	A615	A616	A617	A618	A619	A620	A621	A622	A623	A624	A625	A626	A627	A628	A629	A630	A631	A632	A633	A634	A635	A636	A637	A638	A639	A640	A641	A642	A643	A644	A645	A646	A647	A648	A649	A650	A651	A652	A653	A654	A655	A656	A657	A658	A659	A660	A661	A662	A663	A664	A665	A666	A667	A668	A669	A670	A671	A672	A673	A674	A675	A676	A677	A678	A679	A680	A681	A682	A683	A684	A685	A686	A687	A688	A689	A690	A691	A692	A693	A694	A695	A696	A697	A698	A699	A699	A700	A701	A702	A703	A704	A705	A706	A707	A708	A709	A710	A711	A712	A713	A714	A715	A716	A717	A718	A719	A720	A721	A722	A723	A724	A725	A726	A727	A728	A729	A730	A731	A732	A733	A734	A735	A736	A737	A738	A739	A740	A741	A742	A743	A744	A745	A746	A747	A748	A749	A750	A751	A752	A753	A754	A755	A756	A757	A758	A759	A760	A761	A762	A763	A764	A765	A766	A767	A768	A769	A770	A771	A772	A773	A774	A775	A776	A777	A778	A779	A780	A781	A782	A783	A784	A785	A786	A787	A788	A789	A789	A790	A791	A792	A793	A794	A795	A796	A797	A798	A799	A799	A800	A801	A802	A803	A804	A805	A806	A807	A808	A809	A810	A811	A812	A813	A814	A815	A816	A817	A818	A819	A820	A821	A822	A823	A824	A825	A826	A827	A828	A829	A830	A831	A832	A833	A834	A835	A836	A837	A838	A839	A840	A841	A842	A843	A844	A845	A846	A847	A848	A849	A850	A851	A852	A853	A854	A855	A856	A857	A858	A859	A860	A861	A862	A863	A864	A865	A866	A867	A868	A869	A870	A871	A872	A873	A874	A875	A876	A877	A878	A879	A880	A881	A882	A883	A884	A885	A886	A887	A888	A889	A889	A890	A891	A892	A893	A894	A895	A896	A897	A898	A899	A899	A900	A901	A902	A903	A904	A905	A906	A907	A908	A909	A910	A911	A912	A913	A914	A915	A916	A917	A918	A919	A920	A921	A922	A923	A924	A925	A926	A927	A928	A929	A930	A931	A932	A933	A934	A935	A936	A937	A938	A939	A940	A941	A942	A943	A944	A945	A946	A947	A948	A949	A950	A951	A952	A953	A954	A955	A956	A957	A958	A959	A960	A961	A962	A963	A964	A965	A966	A967	A968	A969	A970	A971	A972	A973	A974	A975	A976	A977	A978	A979	A980	A981	A982	A983	A984	A985	A986	A987	A988	A989	A989	A990	A991	A992	A993	A994	A995	A996	A997	A998	A999	A999	A1000	A1001	A1002	A1003	A1004	A1005	A1006	A1007	A1008	A1009	A1010	A1011	A1012	A1013	A1014	A1015	A1016	A1017	A1018	A1019	A1020	A1021	A1022	A1023	A1024	A1025	A1026	A1027	A1028	A1029	A1030	A1031	A1032	A1033	A1034	A1035	A1036	A1037	A1038	A1039	A1040	A1041	A1042	A1043	A1044	A1045	A1046	A1047	A1048	A1049	A1050	A1051	A1052	A1053	A1054	A1055	A1056	A1057	A1058	A1059	A1060	A1061	A1062	A1063	A1064	A1065	A1066	A1067	A1068	A1069	A1069	A1070	A1071	A1072	A1073	A1074	A1075	A1076	A1077	A1078	A1079	A1079	A1080	A1081	A1082	A1083	A1084	A1085	A1086	A1087	A1088	A1089	A1089	A1090	A1091	A1092	A1093	A1094	A1095	A1096	A1097	A1098	A1099	A1099	A1100	A1101	A1102	A1103	A1104	A1105	A1106	A1107	A1108	A1109	A1110	A1111	A1112	A1113	A1114	A1115	A1116	A1117	A1118	A1119	A1119	A1120	A1121	A1122	A1123	A1124	A1125	A1126	A1127	A1128	A1129	A1129	A1130	A1131	A1132	A1133	A1134	A1135	A1136	A1137	A1138	A1139	A1139	A1140	A1141	A1142	A1143	A1144	A1145	A1146	A1147	A1148	A1149	A1149	A1150	A1151	A1152	A1153	A1154	A1155	A1156	A1157	A1158	A1159	A1159	A1160	A1161	A1162	A1163	A1164	A1165	A1166	A1167	A1168	A1169	A1169	A1170	A1171	A1172	A1173	A1174	A1175	A1176	A1177	A1178	A1179	A1179	A1180	A1181	A1182	A1183	A1184	A1185	A1186	A1187	A1188	A1189	A1189	A1190	A1191	A1192	A1193	A1194	A1195	A1196	A1197	A1198	A1199	A1199	A1200	A1201	A1202	A1203	A1204	A1205	A1206	A1207	A1208	A1209	A1210	A1211	A1212	A1213	A1214	A1215	A1216	A1217	A1218	A1219	A1219	A1220	A1221	A1222	A1223	A1224	A1225	A1226	A1227	A1228	A1229	A1229	A1230	A1231	A1232	A1233	A1234	A1235	A1236	A1237	A1238	A1239	A1239	A1240	A1241	A1242	A1243	A1244	A1245	A1246	A1247	A1248	A1249	A1249	A1250	A1251	A1252	A1253	A1254	A1255	A1256	A1257	A1258	A1259	A1259	A1260	A1261	A1262	A1263	A1264	A1265	A1266	A1267	A1268	A1269	A1269	A1270	A1271	A1272	A1273	A1274	A1275	A1276	A1277	A1278	A1279	A1279	A1280	A1281	A1282	A1283	A1284	A1285	A1286	A1287	A1288	A1289	A1289	A1290	A1291	A1292	A1293	A1294	A1295	A1296	A1297	A1298	A1299	A1299	A1300	A1301	A1302	A1303	A1304	A1305	A1306	A1307	A1308	A1309	A1310	A1311	A1312	A1313	A1314	A1315	A1316	A1317	A1318	A1319	A1319	A1320	A1321	A1322	A1323	A1324	A1325	A1326	A1327	A1328	A1329	A1329	A1330	A1331	A1332	A1333	A1334	A1335	A1336	A1337	A1338	A1339	A1339	A1340	A1341	A1342	A1343	A1344	A1345	A1346	A1347	A1348	A1349	A1349	A1350	A1351	A1352	A1353	A1354	A1355	A1356	A1357	A1358	A1359	A1359	A1360	A1361	A1362	A1363	A1364	A1365	A1366	A1367	A1368	A1369	A1369	A1370	A1371	A1372	A1373	A1374	A1375	A1376	A1377	A1378	A1379	A1379	A1380	A1381	A1382	A1383	A1384	A1385	A1386	A1387	A1388	A1389	A1389	A1390	A1391	A1392	A1393	A1394	A1395	A1396	A1397	A1398	A1399	A1399	A1400	A1401	A1402	A1403	A1404	A1405	A1406	A1407	A1408	A1409	A1410	A1411	A1412	A1413	A1414	A1415	A1416	A141

Where is Waldo



Time to change?



Alternative



BCEA

heemod

hesim

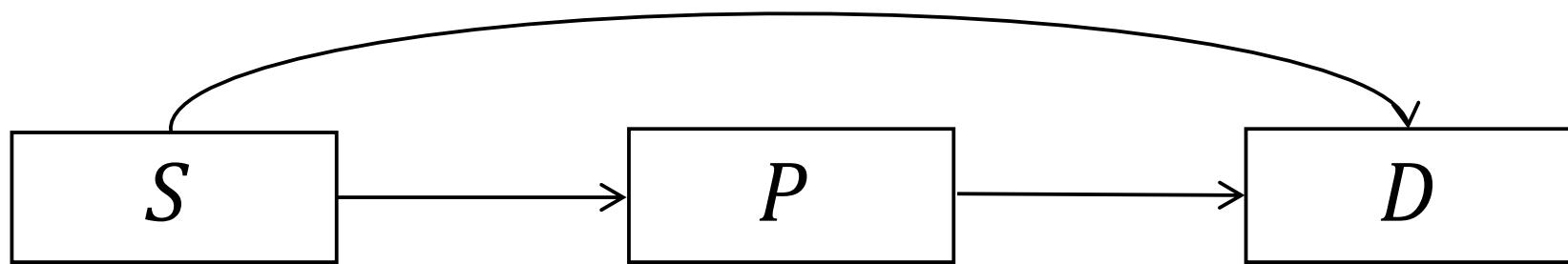
...

What is R

- Statistical programming 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

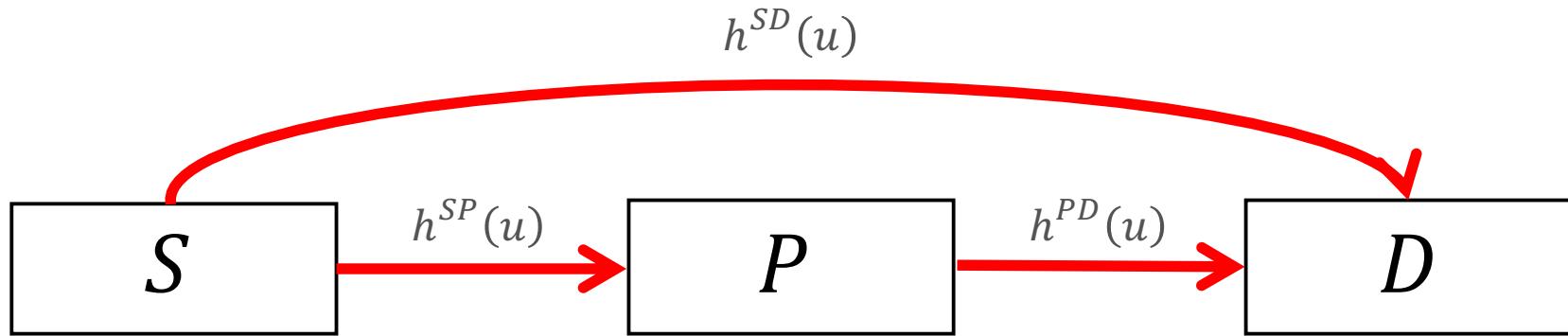
Basic model taxonomy

Health states describing course of disease over time



Markov model

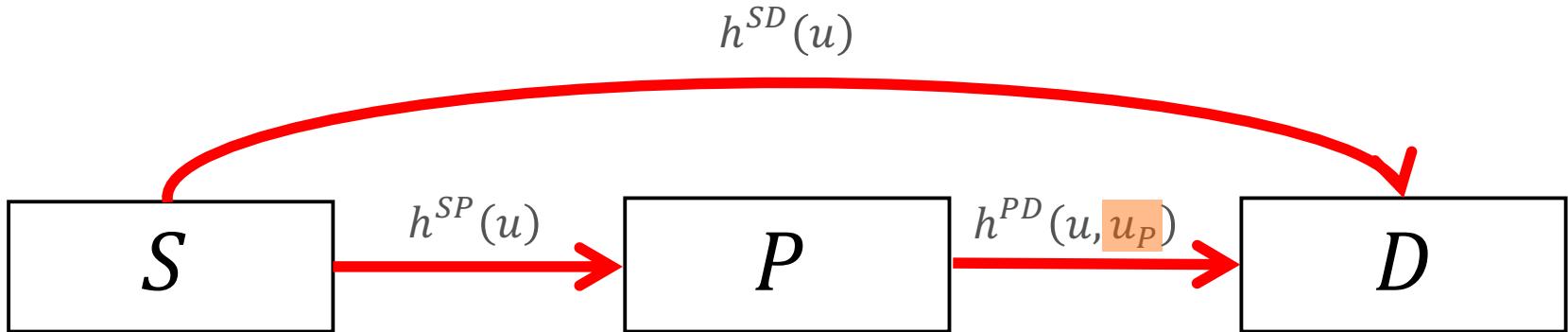
Clock forward



transition rates depend only on time in model

Semi-Markov model

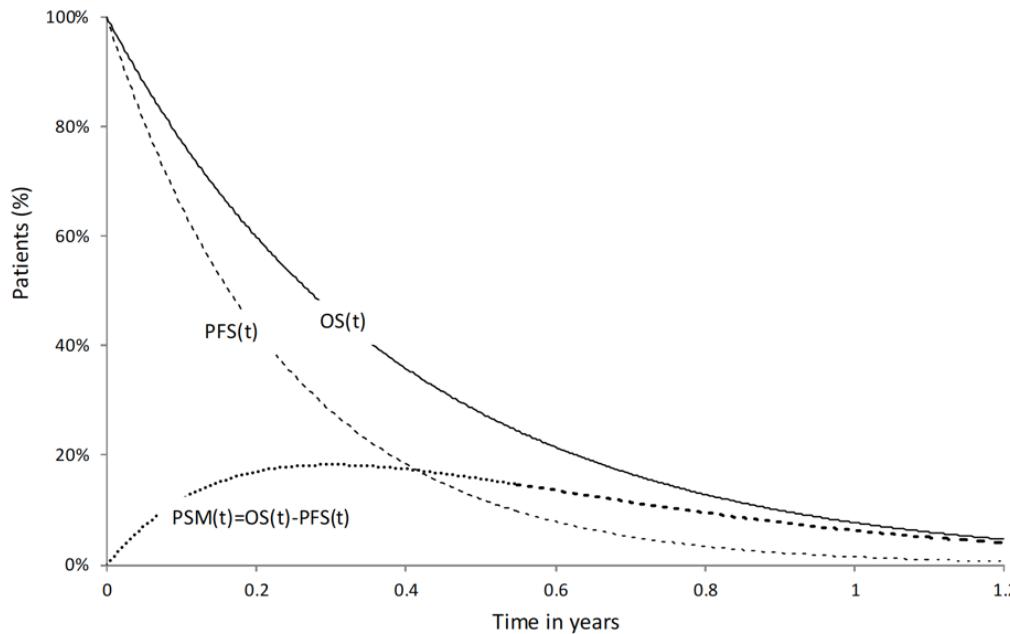
Clock reset



transition rates depend on time in model

some transitions depend on time in an intermediate health state

Partitioned survival model



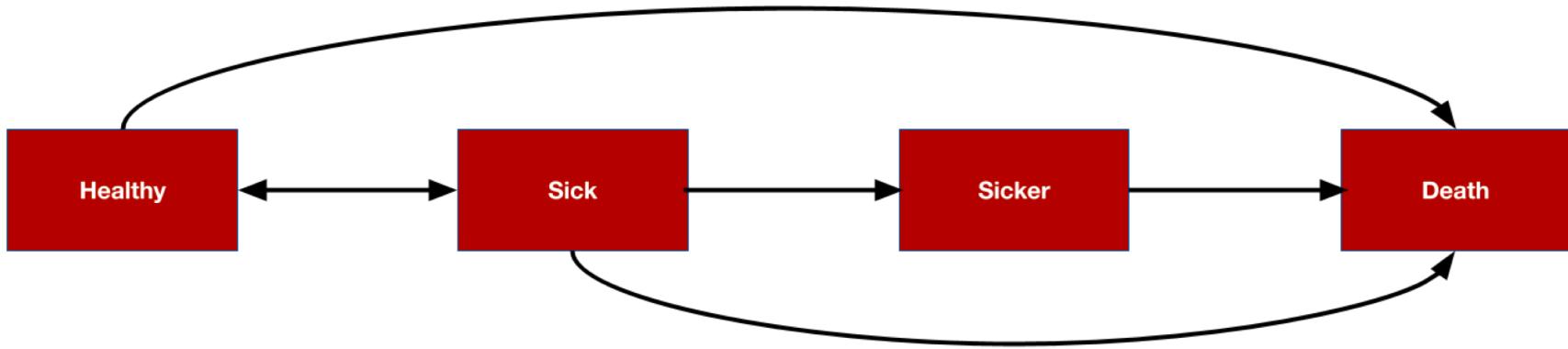
Summary of model types

			discrete time	continuous time
State transition models	Markov ("Clock forward")	Cohort	cohort discrete time state transition models (cDTSTM) <ul style="list-style-type: none"> • time-homogeneous Markov models • time-inhomogeneous Markov models 	cohort continuous time state transition models
		Individual-level	individual-level discrete time state transition models (iDTSTM)	individual-level continuous time state transition models (iCTSTM)
	Semi-Markov ("Clock reset")*	Individual-level	iDTSTM	iCTSTM
Partitioned survival model		Cohort		✓

*tunnel states can be used in a cohort model to approximate a semi-Markov process

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	Cycle	Healthy	Sick	Sicker	Death	discounted QALYs	discounted medical costs	discounted treatment costs	discounted total costs
Healthy	0.848	0.15	0	0.002	0	1.00000	0.00000	0.00000	0.00000	1.00000	2000.000	2000	
Sick	0.5	0.389	0.105	0.006	1	0.84800	0.15000	0.00000	0.00200	0.93252	2229.126	1937.864	
Sicker	0	0	0.98	0.02	2	0.79410	0.18555	0.01575	0.00460	0.88712	2419.321	1876.527	
Death	0	0	0	1	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	
	Healthy	Sick	Sicker	Death	80	0.11578	0.02960	0.29714	0.55748	0.02693	451.752	83.173	
Utility	1	0.75	0.5	0	81	0.11298	0.02888	0.29430	0.56384	0.02571	433.941	79.591	
Direct medical	2000	4000	15000	0	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	
Drug costs	2000				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

```
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
```

```
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

```
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])  
  
  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)
```

	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	0.0000000	0.0000000	0.000000000
1	0.8480000	0.1500000	0.0000000	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	0.0000000	0.0000000	0.000000000
1	0.8784000	0.1200000	0.0000000	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

```
compute_qalys <- function(x, utility, dr = .03){  
  n_cycles <- nrow(x) - 1  
  pv(x %*% utility, dr, 0:n_cycles)  
}
```

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

```
head(dqalys_soc)
```

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

```
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_tb1 <- 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_tb1(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 x
Source on Save | Run | Source | ...
1 ## ---- Overview -----
2 ## @knitr R-packages
3 library("rcea")
4 library("knitr")
5 library("kableExtra")
6 library("magrittr")
7 library("tibble")
8
9 ## ---- Model parameters -----
10 ## @knitr transition-probabilities
11 p_hd <- .0002 # constant probability of dying when Healthy (all-cause mortality)
12 p_hs1 <- .15 # probability of becoming Sick when Healthy
13 p_s1h <- .05 # probability of becoming Healthy when Sick
14 p_s1s2 <- .0105 # probability of becoming Sicker when Sick
15 p_s1d <- .006 # constant probability of dying when Sick
16 p_s2d <- .02 # constant probability of dying when Sicker
17
18 ## @knitr transition-probability-complements
19 p_hh <- 1 - p_hs1 - p_hd
20 p_s1s1 <- 1 - p_s1h - p_s1s2 - p_s1d
21 p_s2s2 <- 1 - p_s2d
22
23 ## @knitr tpmatrix
24 p_soc <- matrix(
25   c(p_hh, p_hs1, 0, p_hd,
26     p_s1h, p_s1s1, p_s1s2, p_s1d,
27     0, 0, p_s2s2, p_s2d,
28     0, 0, 0, 1),
29   byrow = TRUE,
30   nrow = 4, ncol = 4
31 )
32 state_names <- c("H", "S1", "S2", "D")
33 colnames(p_soc) <- rownames(p_soc) <- state_names
34 print(p_soc)
35
36 ## @knitr apply_rr
37 apply_rr <- function(p, rr = .8){
38   p["H", "S1"] <- p["H", "S1"] * rr
39   p["H", "S2"] <- p["H", "S2"] * rr
40   p["H", "D"] <- p["H", "D"] * rr
41   p["H", "H"] <- 1 - sum(p["H", -1])
42
43   p["S1", "S2"] <- p["S1", "S2"] * rr
44   p["S1", "D"] <- p["S1", "D"] * rr
45   p["S1", "S1"] <- 1 - sum(p["S1", -2])
```

Tutorial

rcea **0.1.2**

Reference

Tutorials ▾

Slides



Simple Markov Cohort Model

2021-07-26

Source: vignettes/01-markov-cohort.Rmd

Contents

Overview

Theory

Model parameters

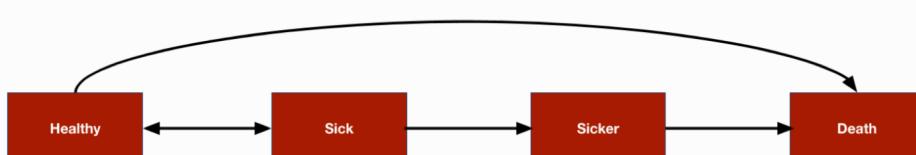
Simulation

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

REFERRED TO AS A MARKOV COHORT MODEL. IN THIS TUTORIAL WE DEMONSTRATE IMPLEMENTATION WITH R OF THE SIMPLEST OF COST-EFFECTIVENESS MODELS, A LITTLE HOMOGENEOUS MODEL WITH TRANSITION PROBABILITIES THAT ARE CONSTANT OVER TIME. THE ENTIRE ANALYSIS CAN BE RUN USING **BASE R** (I.E., WITHOUT INSTALLING ANY PACKAGES). HOWEVER, WE WILL USE THE FOLLOWING PACKAGES TO CREATE A NICE LOOKING COST-EFFECTIVENESS TABLE.

```
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 of 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.



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)
```

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

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

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

```
# 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)
}
```

```

sim_model <- function(params_rng, data, n_cycles = 85, dr_qalys = .03, dr_costs = .03){
  PSA samples; tx strategy; no. cycles; discount rates
  # 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)
}

```

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 output

```
ce_sim <- sim_out[cycle != 0,
                  .(dqalys = sum(dqalys),
                    dcots = sum(dcots_med) + sum(dcots_treat)),
                  by = c("sample", "strategy")]
ce_sim
```

```
      sample strategy    dqalys    dcots
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
```

Save for later

```
saveRDS(ce_sim, file = "markov-cohort-ce_sim.rds")
```

Cost-effectiveness output

```
ce_sim_wider <- dcast(ce_sim, sample ~ strategy,  
                      value.var = c("dqalys", "dcosts"))  
  
ce_sim_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 output

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

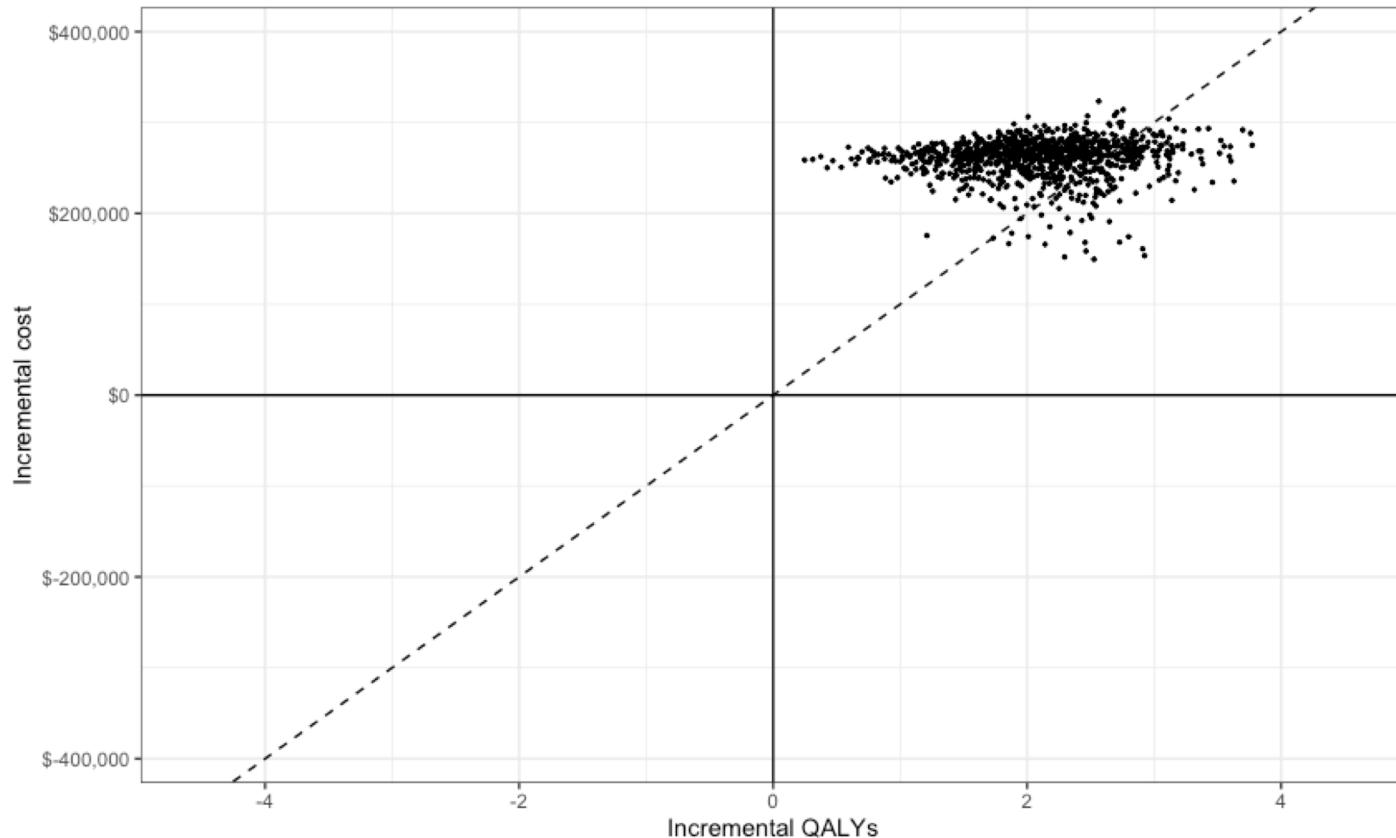
	sample	dqalys_New	dqalys_SOC	dcosts_New	dcosts_SOC	idcosts	idqalys
1:	1	23.30567	21.36614	372636.3	103030.72	269605.6	1.9395308
2:	2	23.49382	22.71959	380159.8	104017.07	276142.7	0.7742344
3:	3	23.35243	21.81066	463270.5	207518.50	255752.0	1.5417635
4:	4	23.49622	21.45324	680491.2	474308.21	206183.0	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_sim_wider[, .(icer = mean(idcosts)/mean(idqalys))]
```

	icer
1:	125028.7

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       
1 ## ---- Overview -----  
2 ## @knitr R-packages  
3 library("rcaea")  
4 library("hesim")  
5 library("data.table")  
6 library("magrittr")  
7 library("ggplot2")  
8  
9 ## ---- Model parameters -----  
10 ## @knitr tpmatrix  
11 transitions_soc <- matrix(  
12   c(848, 150, 0, 2,  
13     500, 389, 105, 6,  
14     0, 0, 784, 16,  
15     0, 0, 0, 23),  
16   nrow = 4, byrow = TRUE)  
17 state_names <- c("H", "S1", "S2", "D")  
18 colnames(transitions_soc) <- rownames(transitions_soc) <- tolower(state_names)  
19  
20 ## @knitr all-parameters  
21 params <- list(  
22   alpha_soc = transitions_soc,  
23   lrr_mean = log(.8),  
24   lrr_lower = log(.71),  
25   lrr_upper = log(.9),  
26   c_medical = c(H = 2000, S1 = 4000, S2 = 15000, D = 0),  
27   c_soc = 2000,  
28   c_new = 12000,  
29   u_mean = c(H = 1, S1 = .75, S2 = 0.5, D = 0),  
30   u_se = c(H = 0, S1 = 0.03, S2 = 0.05, D = 0.0)  
31 )  
32  
33 ## ---- Simulation -----  
34 ## @knitr sample-parameters  
35 rng_def <- define_rng({  
36   lrr_se <- (lrr_upper - lrr_lower)/(2 * qnorm(.975)) # Local object  
37   # not returned  
38   list( # Parameters to return  
39     p_soc = dirichlet_rng(alpha_soc),  
40     rr_new = lognormal_rng(lrr_mean, lrr_se),  
41     c_medical = gamma_rng(mean = c_medical, sd = c_medical),  
42     c_soc = c_soc,  
43     c_new = c_new,  
44     u = beta_rng(mean = u_mean, sd = u_se)  
45   )  
46 } sim_model(params_rng, data, n_cycles, dr_qalys, dr_costs) #
```

Tutorial

rcea **0.1.2**

Reference

Tutorials ▾

Slides



2021-07-26

Source: vignettes/02-markov-cohort-psa.Rmd

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")
```



Model parameters

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,
```

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*

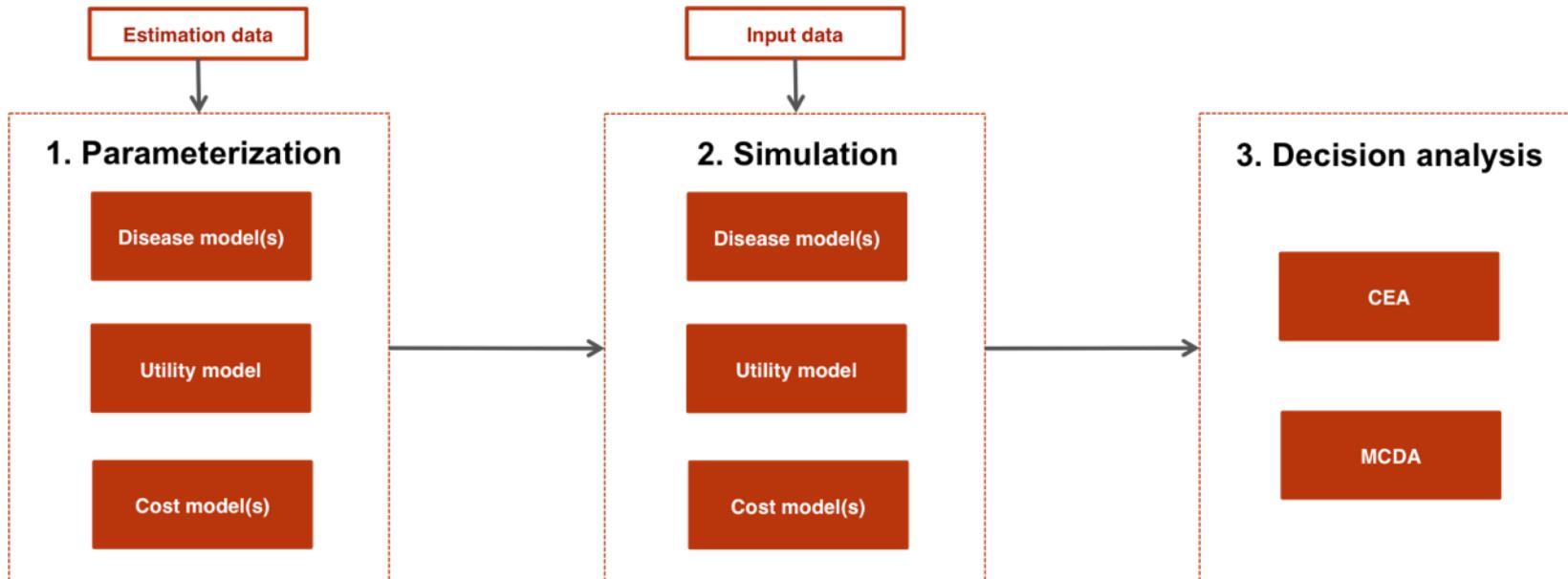
Simple Markov cohort model with hesim



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"
```

```
Tabs <- get_labels(hesim_dat)
```

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

- Typically underlying parameters (`params`) are transformed (`tparams`) into more relevant parameters 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...

```

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)

```

```
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()`¹

```
input_data <- 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
```

¹Could also be expanded by time intervals in a time-inhomogeneous model

Model parameters – Transformed parameters

- You write mathematical expressions
- Vectorized 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  
    )  
  )  
})
```

Annotations:

- rr <- ifelse(strategy_name == "SOC", 1, rr_new)
 - rr: Input data
 - 1: Parameter
- tpmatrix(
 - C: Parameter
 - p_soc\$h_s1 * rr: Parameter (defined above)
 - p_soc\$h_s2 * rr: Parameter (defined above)
 - p_soc\$h_d * rr: Parameter (defined above)
 - p_soc\$s1_h: Parameter
 - C: Parameter (defined above)
 - p_soc\$s1_s2 * rr: Parameter (defined above)
 - p_soc\$s1_d * rr: Parameter (defined above)
 - p_soc\$s2_h: Parameter
 - p_soc\$s2_s1: Parameter
 - C: Parameter (defined above)
 - p_soc\$s2_d * rr: Parameter (defined above)
 - 0: Input data
 - 0: Input data
 - 0: Input data
 - 1: Parameter
- utility = u: Parameter
- costs = list(
 - treatment = ifelse(strategy_name == "SOC", c_soc, c_new): Input data
 - medical = c_medical: Input data
-)

tpmatrix() is a powerful function for creating/storing transition probabilities that vary across PSA samples, strategies, subgroups, and time intervals

C denotes the “complement”, ensuring that the probabilities in a row sum to 1

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 0	1.0000000
2:	1		1	1		1 1	0.8466964
3:	1		1	1		1 2	0.7929241
4:	1		1	1		1 3	0.7652357
5:	1		1	1		1 4	0.7446251
6:	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 0.03	14.604941	14.604941
2:	1		1	1		2 0.03	2.718281	3.660628
3:	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 0.03	treatment	29209.882
2:	1		1	1		2 0.03	treatment	7321.257
3:	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**.
- 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"
```

Save for later

```
saveRDS(ce_sim, file = "markov-cohort-hesim-ce_sim.rds")
saveRDS(hesim_dat, file = "markov-cohort-hesim_data.rds")
```

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,  
                      k = seq(0, 25000, 500))
```

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

```
format(icer(cea_pw_out, k = 50000, labels = 1abs))
```

	Outcome	New
1: Incremental QALYs		2.07 (1.05, 3.13)
2: Incremental costs	257,297	(203,094, 284,985)
3: Incremental NMB	-153,833	(-210,941, -77,824)
4: ICER		124,342

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

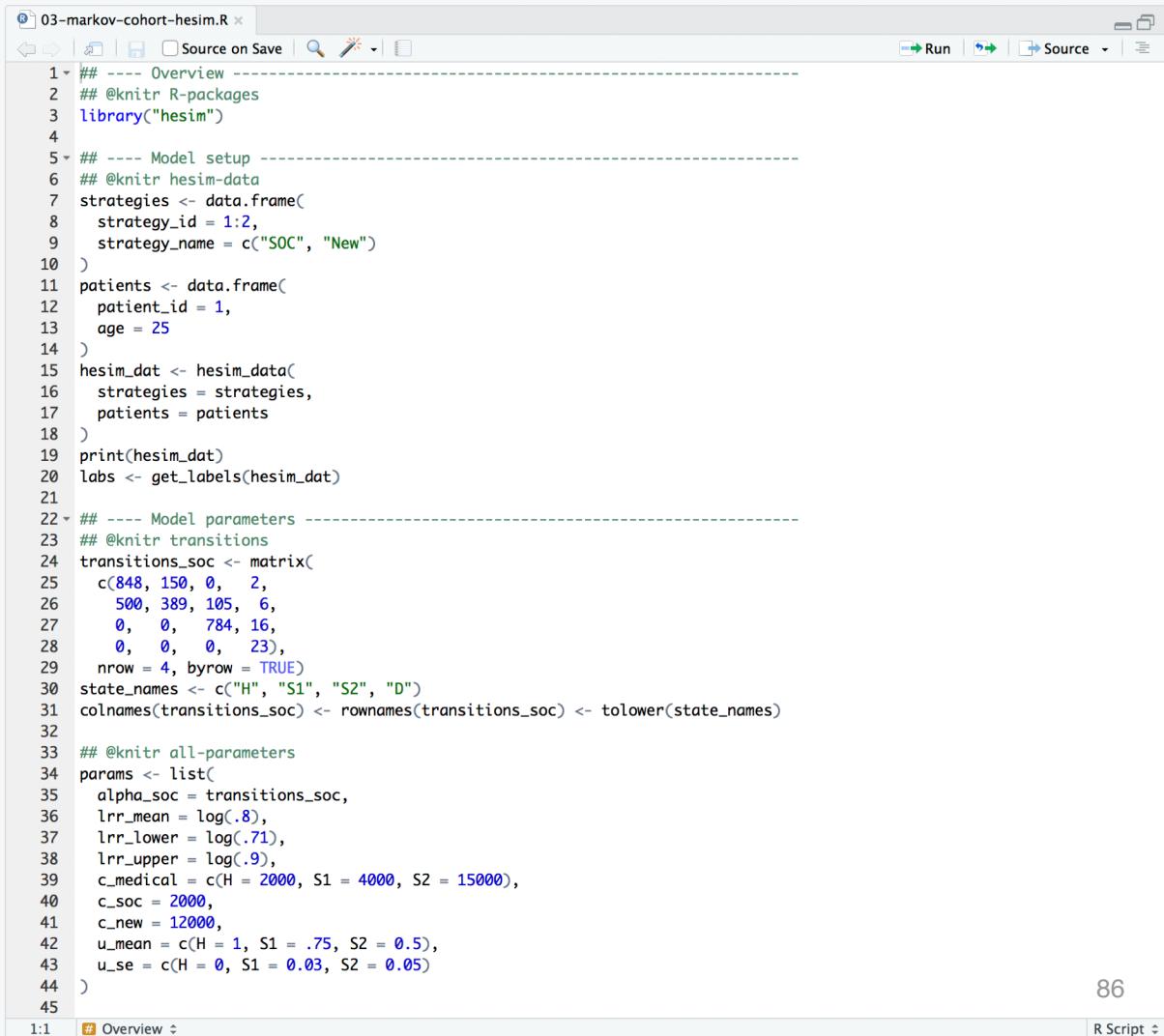
- Create an economic model—consisting of separate statistical models for disease progression, costs, and utilities—that simulate outcomes as a function of *input data* (derived from Step 1) and *parameters* (from Step 2)
- `define_model()`, `create_CohortDtsim()`

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



```
R 03-markov-cohort-hesim.R x
Source on Save | Run | Source | Overview | Help
1 ## ---- Overview -----
2 ## @knitr R-packages
3 library("hesim")
4
5 ## ---- Model setup -----
6 ## @knitr hesim-data
7 strategies <- data.frame(
8   strategy_id = 1:2,
9   strategy_name = c("SOC", "New")
10 )
11 patients <- data.frame(
12   patient_id = 1,
13   age = 25
14 )
15 hesim_dat <- hesim_data(
16   strategies = strategies,
17   patients = patients
18 )
19 print(hesim_dat)
20 labs <- get_labels(hesim_dat)
21
22 ## ---- Model parameters -----
23 ## @knitr transitions
24 transitions_soc <- matrix(
25   c(848, 150, 0, 2,
26     500, 389, 105, 6,
27     0, 0, 784, 16,
28     0, 0, 0, 23),
29   nrow = 4, byrow = TRUE)
30 state_names <- c("H", "S1", "S2", "D")
31 colnames(transitions_soc) <- rownames(transitions_soc) <- tolower(state_names)
32
33 ## @knitr all-parameters
34 params <- list(
35   alpha_soc = transitions_soc,
36   lrr_mean = log(.8),
37   lrr_lower = log(.71),
38   lrr_upper = log(.9),
39   c_medical = c(H = 2000, S1 = 4000, S2 = 15000),
40   c_soc = 2000,
41   c_new = 12000,
42   u_mean = c(H = 1, S1 = .75, S2 = 0.5),
43   u_se = c(H = 0, S1 = 0.03, S2 = 0.05)
44 )
45
# Overview
```

Tutorial

rcea **0.1.2**

Reference

Tutorials ▾

Slides



Markov Cohort Model with hesim

2021-07-26

<https://hesim-dev.github.io/rcea/articles/03-markov-cohort-hesim.html>

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.

More information about `hesim` can be found by visiting the [package website](#). We recommend reading the "Articles"—starting with the "[Introduction to hesim](#)"—to learn more. Economic models can, in general, be simulated with the following steps:

1. **Model setup:** Specify the treatment strategies, target population(s), and model structure.
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")
```

Model setup

Before beginning an analysis, it is necessary to define the treatment strategies of interest and the the target population of interest. We

Contents

Overview

Simulation

Cost-effectiveness analysis

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 72)

■ 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  
        )  
    )  
})
```

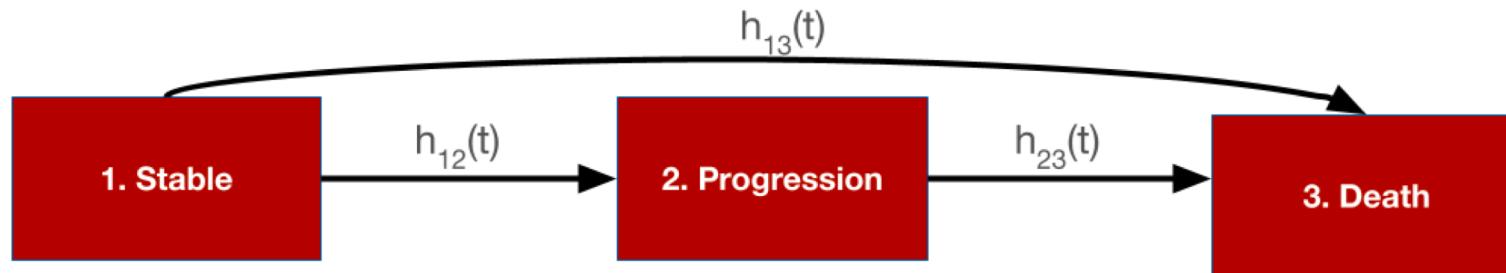
Semi-Markov multi-state model

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"
```

```
tabs <- get_labels(hesim_dat)
```

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 strategy_name female      age patient_id time_start
1: Stable Progression      New 0 59.85813 1 0.000000
2: Stable Death            New 0 59.85813 1 0.000000
3: Progression Death      New 0 59.85813 1 2.420226
4: Stable Progression      New 0 62.57282 2 0.000000
5: Stable Death            New 0 62.57282 2 0.000000

      time_stop status transition_id strategy_id      time
1: 2.420226    1           1          3 2.420226
2: 2.420226    0           2          3 2.420226
3: 14.620258   1           3          3 12.200032
4: 7.497464    0           1          3 7.497464
5: 7.497464    0           2          3 7.497464
```

■ 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  2.0152  1.9226  2.1124  0.0484      NA       NA       NA
scale        NA  7.1668  6.7796  7.5762  0.2031      NA       NA       NA
strategy_nameNew 0.5281  0.2745  0.2131  0.3360  0.0314  1.3159  1.2375  1.3994
female       0.4947 -0.1902 -0.2518 -0.1286  0.0314  0.8268  0.7774  0.8793
```

N = 1975, Events: 1006, Censored: 969

1) Total time at risk: 9192.058

Log-likelihood = -2906.248, df = 4

AIC = 5820.497

*New treatment increases time
to progression (AFT model)*

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  2.482459  2.303776  2.675000  0.094614      NA        NA        NA
scale          NA 10.406898  9.612760 11.266641  0.421473      NA        NA        NA
strategy_nameNew 0.528101  0.241482  0.157695  0.325268  0.042749  1.273134  1.170810  1.384402
female         0.494684 -0.083337 -0.167300  0.000626  0.042839  0.920041  0.845945  1.000626
```

N = 1975, Events: 358, Censored: 1617

Total time at risk: 9192.058

Log-likelihood = -1333.968, df = 4

AIC = 2675.936

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  3.48340  3.25461  3.72826  0.12074      NA      NA      NA  
scale        NA  8.96768  8.55835  9.39658  0.21376      NA      NA      NA  
strategy_nameNew 0.50398  0.00922 -0.04311  0.06155  0.02670  1.00926  0.95780  1.06348  
female       0.52386 -0.11650 -0.16914 -0.06385  0.02686  0.89003  0.84439  0.93815
```

N = 1006, Events: 468, Censored: 538

Total time at risk: 5479.46

Log-likelihood = -1237.573, df = 4

AIC = 2483.147

Shape parameter: whether the hazard is increasing, decreasing, or constant

Scale: whether the hazard is lower/higher at given time point

Parameters – Utility

```
utility_tb1 <- stateval_tb1(
  data.table(state_id = states$state_id,
            mean = c(.8, .6),
            se = c(0.02, .05)
  ),
  dist = "beta")
```

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

Parameters – Medical cost

```
medcost_tb1 <- stateval_tb1(  
  data.table(state_id = states$state_id,  
            mean = c(2000, 9500),  
            se = c(2000, 9500)  
,  
  dist = "gamma")
```

```
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")
```

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	0.00	0.25	2000
2:		1	1	0.25	Inf	2000
3:		1	2	0.00	0.25	2000
4:		1	2	0.25	Inf	2000
5:		2	1	0.00	0.25	12000
6:		2	1	0.25	Inf	12000
7:		2	2	0.00	0.25	12000
8:		2	2	0.25	Inf	10000
						104

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,
                                 hesim_data = hesim_dat)

# Costs
drugcostmod <- create_StateVals(drugcost_tbl, n = 500,
                                   time_reset = TRUE, ←
                                   hesim_data = hesim_dat) So that costs depend on  
time in intermediate state

medcostmod <- create_StateVals(medcost_tbl, n = 500,
                                 hesim_data = hesim_dat)

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

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

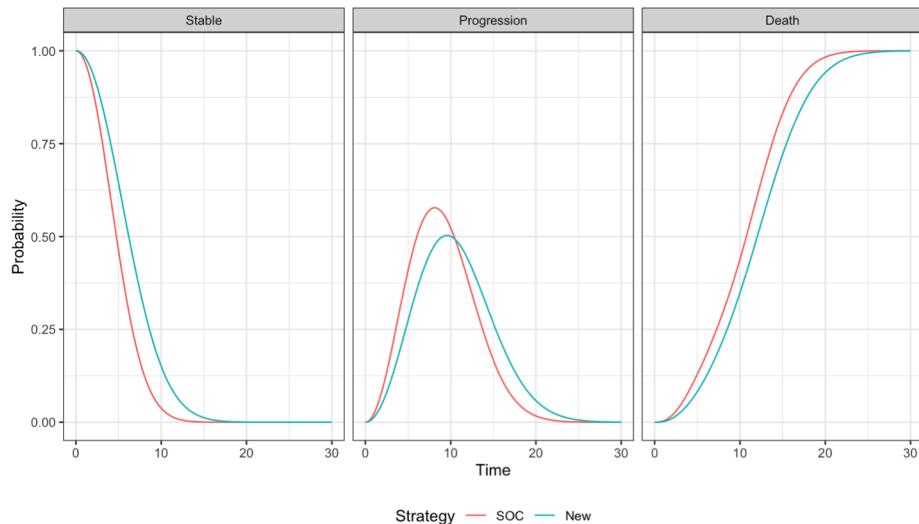
Disease progression

```
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	2	0	0.000000	1.420130
2:	1		1	1	1	2	3	1	1.420130
3:	1		1	2	1	1	2	0	0.000000
4:	1		1	2	1	2	3	1	5.711780
5:	1		1	3	1	1	2	0	0.000000
6:	1		1	3	1	2	3	1	5.249609

```
econmod$sim_stateprobs(t = seq(0, 30 , 1/12))  
autoplots(econmod$stateprobs_, labels = 1abs)
```



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 3.795199 5.009791
2:     1             1       1           2 0 3.433499 5.368864
3:     1             2       1           1 0 4.898627 6.466354
4:     1             2       1           2 0 3.491527 5.459601
5:     2             1       1           1 0 3.882274 4.793997
6:     2             1       1           2 0 3.220405 5.746082
```

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

```
sample strategy_id grp_id state_id dr category      costs
1:     1             1       1           1 0.03    Drug  9124.834
2:     1             1       1           2 0.03    Drug  8261.846
3:     1             2       1           1 0.03    Drug  69025.893
4:     1             2       1           2 0.03    Drug  40511.631
5:     2             1       1           1 0.03    Drug  8778.825
6:     2             1       1           2 0.03    Drug  8863.685
```

Cost-effectiveness analysis

```
ce_sim <- econmod$summarize()

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

format(icer(cea_pw_out, labels = 1abs))
```

	Outcome	New
1:	Incremental QALYs	0.86 (0.56, 1.17)
2:	Incremental costs	92,981 (81,657, 101,697)
3:	Incremental NMB	-50,160 (-61,751, -38,254)
4:	ICER	108,571

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_tb1()`

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

The screenshot shows an RStudio interface with a script file titled "04-mstate.R". The code is a complete R script for state transition modeling, using packages like rcea, hesim, data.table, ggplot2, and flexsurv. It includes sections for library imports, setting random seeds, defining transition matrices, generating patient data, and creating state and strategy data frames. The code is annotated with comments and uses various R functions like rbind, rnorm, and rbinom.

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

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

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

## @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
)
```

Tutorial

rcea **0.1.2**

Reference

Tutorials ▾

Slides



Semi-Markov Multi-state Model

2021-07-26

Contents

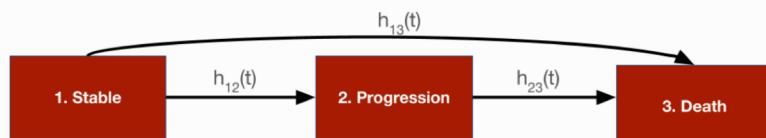
Overview

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

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 parameters of the health state transitions using a multi-state model so that the simulation model is completely integrated with an underlying statistical model. Third, we use individual patient simulation (IPS) to simulate a semi-Markov model, meaning that (unlike in a Markov model) transitions cannot depend on prior history.

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](#).

RADICAL ESTIMATION

Simulation

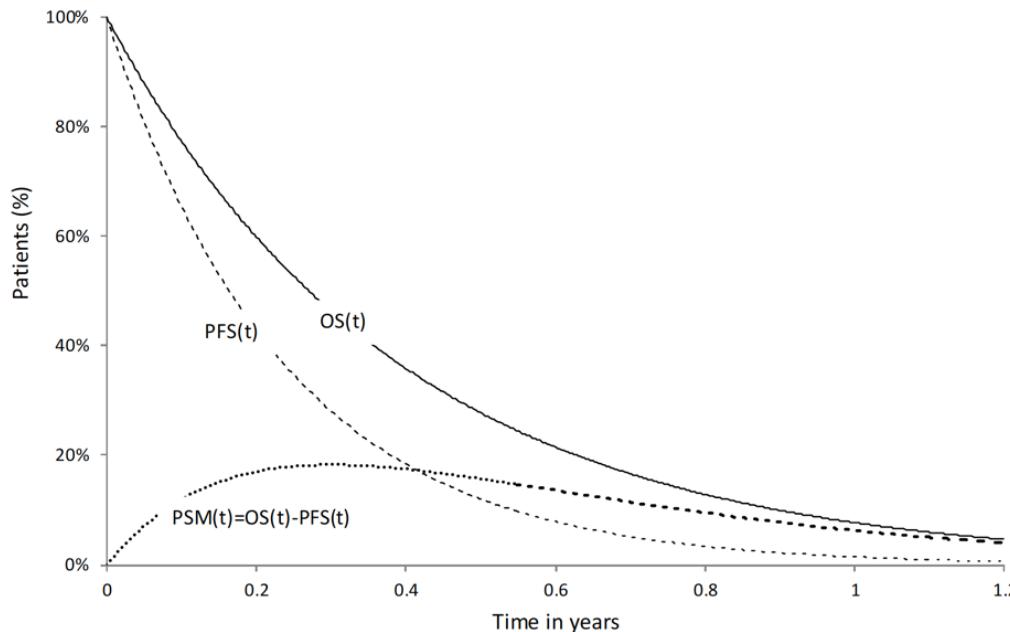
Cost-effectiveness analysis

Exercise 4: Semi-Markov multi-state model

- *Modify R-script “04-mstate.R”*
 - *Simulate homogeneous patient cohort*
 - *Use a generalized gamma distribution for transitions (hint: type ?flexsurvreg into R and look at the options for the dist argument)*
- *Run modified script*

Partitioned survival model

Partitioned survival model



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

- We set up the model in the same way as in the multi-state model, except that since PSMs are cohort models, we no longer need to simulate a large number of patients to compute expected values
- Instead, we will simulate separate cohorts of representative patients of varying ages and sexes

```
patients <- data.table(  
  patient_id = 1:4,  
  patient_wt = rep(1/4, 4), # Each patient has same weight  
  age = c(45, 45, 65, 65),  
  female = c(0, 1, 0, 1)  
)  
  
states <- data.table(  
  state_id = c(1, 2),  
  state_name = c("Stable", "Progression") # Non-death states  
)  
  
strategies <- data.frame(  
  strategy_id = 1:2,  
  strategy_name = c("SOC", "New")  
)  
  
hesim_dat <- hesim_data(  
  patients = patients,  
  strategies = strategies,  
  states = states  
)
```

Model setup

```
print(hesim_dat)
$strategies
  strategy_id strategy_name
  1           1             SOC
  2           2             New

$patients
  patient_id patient_wt age female
  1:          1      0.25  45    0
  2:          2      0.25  45    1
  3:          3      0.25  65    0
  4:          4      0.25  65    1

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

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

```
labs <- get_labels(hesim_dat)
```

Parameter estimation – Survival models

- PSMs are parameterized by estimating separate survival models for different endpoints (e.g., PFS and OS)

- Dataset

	patient_id	female	age	strategy_name	pfs_time	pfs_status	os_status	os_time
1:	1	0	59.85813		New	2.420226	1	14.620258
2:	2	0	62.57282		New	7.497464	0	7.497464
3:	3	1	61.44379	SOC	2.365867		1	2.365867
4:	4	0	62.90770		New	9.342265	1	0 11.383095
5:	6	1	53.66665		New	9.922407	0	9.922407

- Estimate parameters

```
fit_pfs_wei <- flexsurvreg(  
  Surv(pfs_time, pfs_status) ~ strategy_name + female,  
  data = surv_est_data,  
  dist = "weibull")  
  
fit_os_wei <- flexsurvreg(  
  Surv(os_time, os_status) ~ strategy_name + female,  
  data = surv_est_data,  
  dist = "weibull")  
  
psfit_wei <- flexsurvreg_list(fit_pfs_wei, fit_os_wei)
```

To maintain consistency with the multi-state model, we will fit Weibull survival models for both PFS and OS and include an indicator for female as a covariate.

The PFS and OS fits are stored in a flexsurvreg_list, which is just a list of models fit using flexsurvreg()

Parameters – Utility

```
utility_tb1 <- stateval_tb1(
  data.table(state_id = states$state_id,
            mean = c(.8, .6),
            se = c(0.02, .05)
  ),
  dist = "beta"
)
```

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

Parameters – Cost

```
medcost_tb1 <- stateval_tb1(  
  data.table(state_id = states$state_id,  
            mean = c(2000, 9500),  
            se = c(2000, 9500)  
,  
  dist = "gamma"  
)
```

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

```
drugcost_tb1 <- stateval_tb1(  
  data.table(strategy_id = strategies$strategy_id,  
            est = c(2000, 12000)  
,  
  dist = "fixed"  
)
```

	strategy_id	est
1:	1	2000
2:	2	12000

Simulation – Construct the model

Survival models

- Survival predictions are made as a function of the fitted Weibull models and input data. The latter consists of each treatment strategy and patient combination

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

```
strategy_id patient_id strategy_name patient_wt age female
1:          1           1           SOC    0.25   45     0
2:          1           2           SOC    0.25   45     1
3:          1           3           SOC    0.25   65     0
4:          1           4           SOC    0.25   65     1
5:          2           1           New    0.25   45     0
6:          2           2           New    0.25   45     1
7:          2           3           New    0.25   65     0
8:          2           4           New    0.25   65     1
```

```
survmods <- create_PsmCurves(psfit_wei,
                               input_data = survmods_data,
                               n = 100,
                               uncertainty = "bootstrap",
                               est_data = surv_est_data)
```

We must specify arguments related to the PSA. We sample the parameters via bootstrapping, whereby the survival models are refit repeatedly to resamples of the estimation dataset. It preserves the correlation between PFS and OS and reduces the chances that the curves cross

Simulation – Construct the model

Utility and cost models

```
# Utility
utilitymod <- create_StateVals(utility_tbl, n = 100,
                                 hesim_data = hesim_dat)

# Costs
drugcostmod <- create_StateVals(drugcost_tbl, n = 100,
                                   hesim_data = hesim_dat)

medcostmod <- create_StateVals(medcost_tbl, n = 100,
                                 hesim_data = hesim_dat)

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

Simulation – Construct the model

Combining the disease progression, cost, and utility models

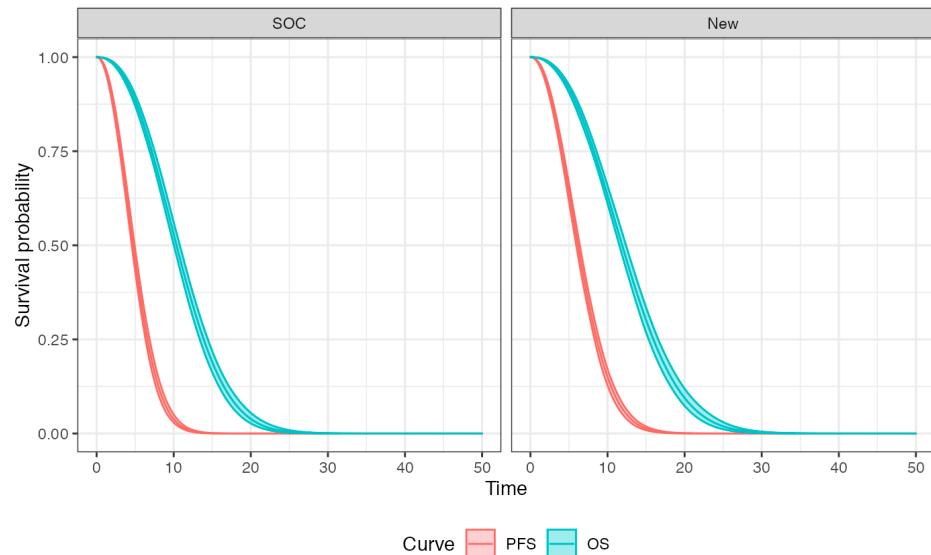
```
econmod <- Psm$new(survival_models = survmods,  
                      utility_model = utilitymod,  
                      cost_models = costmods)
```

Simulation - Simulating outcomes

Survival curves

```
times <- seq(0, 50, by = .1)
econmod$sim_survival(t = times)

# Plot
labs <- c(
  labs,
  list(curve = c("PFS" = 1, "OS" = 2)))
)
autoplot(econmod$survival_, ci = TRUE,
         labels = labs)
```



Simulation - Simulating outcomes

Health state probabilities

```
econmod$sim_stateprobs()
```

	sample	strategy_id	patient_id	grp_id	patient_wt	state_id	t	prob
1:	1	1	1	1	0.25	1	0.0	1.0000000
2:	1	1	1	1	0.25	1	0.1	0.9998138
3:	1	1	1	1	0.25	1	0.2	0.9992079
4:	1	1	1	1	0.25	1	0.3	0.9981533
5:	1	1	1	1	0.25	1	0.4	0.9966344

1202396:	100	2	4	1	0.25	3	49.6	1.0000000
1202397:	100	2	4	1	0.25	3	49.7	1.0000000
1202398:	100	2	4	1	0.25	3	49.8	1.0000000
1202399:	100	2	4	1	0.25	3	49.9	1.0000000
1202400:	100	2	4	1	0.25	3	50.0	1.0000000

Simulation - Simulating outcomes

QALYs and costs

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

sample	strategy_id	patient_id	grp_id	patient_wt	state_id	dr	qalys	lys
1:	1	1	1	1	0.25	1	0.03	3.954569 4.896886
2:	1	1	1	1	0.25	2	0.03	2.491677 4.092347
3:	1	1	2	1	0.25	1	0.03	3.406112 4.217740
4:	1	1	2	1	0.25	2	0.03	2.538972 4.170025
5:	1	1	3	1	0.25	1	0.03	3.954569 4.896886
6:	1	1	3	1	0.25	2	0.03	2.491677 4.092347

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

sample	strategy_id	patient_id	grp_id	patient_wt	state_id	dr	category	costs
1:	1	1	1	1	0.25	1	0.03	Drug 9793.773
2:	1	1	1	1	0.25	2	0.03	Drug 8184.694
3:	1	1	2	1	0.25	1	0.03	Drug 8435.481
4:	1	1	2	1	0.25	2	0.03	Drug 8340.050
5:	1	1	3	1	0.25	1	0.03	Drug 9793.773
6:	1	1	3	1	0.25	2	0.03	Drug 8184.694

Cost-effectiveness analysis

```
ce_sim <- econmod$summarize()

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

format(icer(cea_pw_out, labels = 1abs))
```

	Outcome	New
1:	Incremental QALYs	0.88 (0.61, 1.17)
2:	Incremental costs	102,746 (92,119, 113,340)
3:	Incremental NMB	-58,941 (-70,112, -48,391)
4:	ICER	117,277

Steps with `hesim`

1. Model set-up

- Specify the treatment strategies, target population(s), and states
 - `hesim_data()`

2. Parameters

- Estimate or define the parameters of the economic model
 - `flexsurvreg_list()`, `stateval_tb1()`

3. Simulation

a. Construction of model

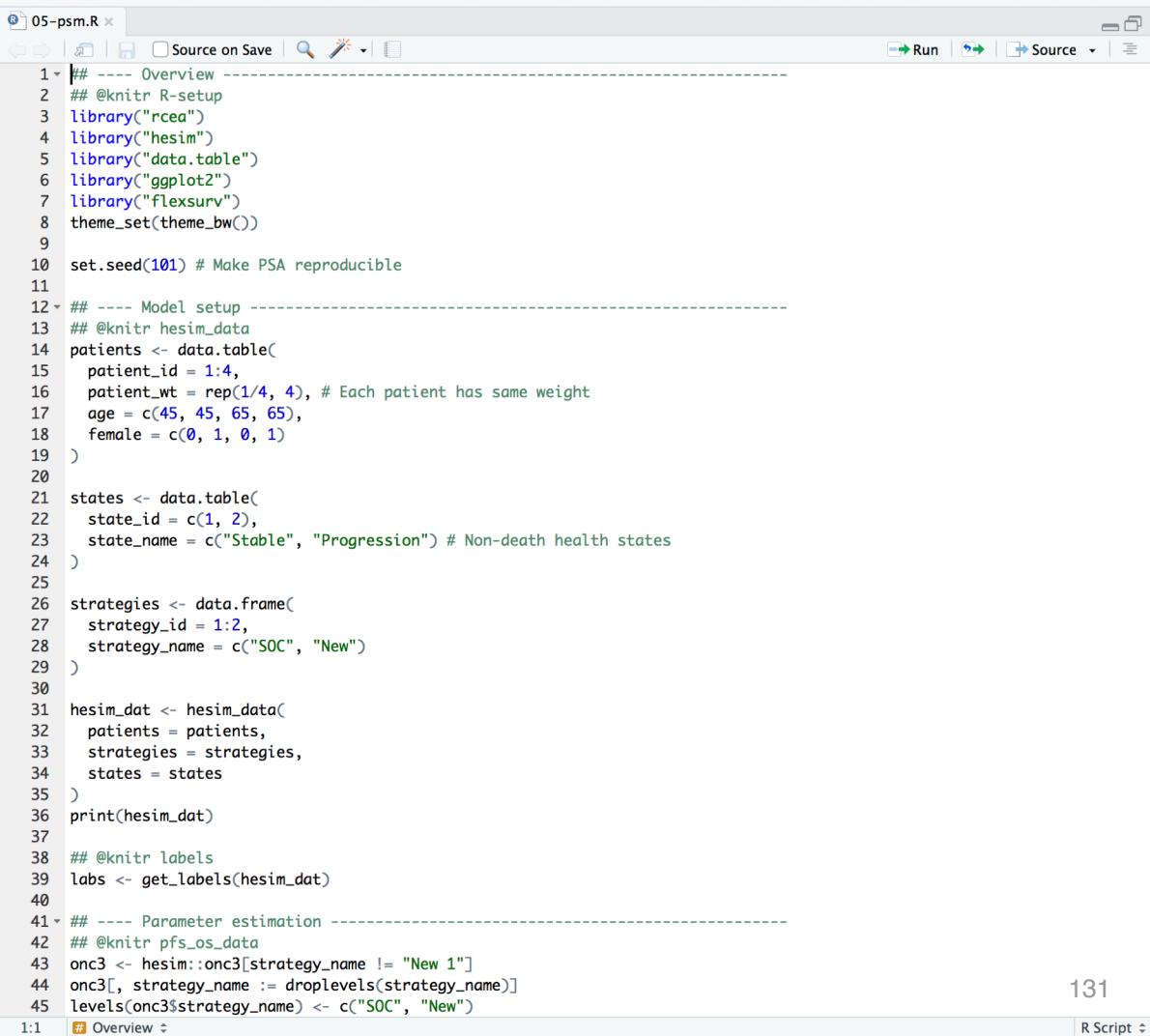
- Create an economic model—consisting of separate statistical models for disease progression (survival models), costs, and utilities—that simulate outcomes as a function of *input data* (derived from Step 1) and *parameters* (from Step 2)
- `create_PsmCurves()`, `create_Statevals()`, `Psm$new()`

b. Simulation of outcomes

- Simulate outcomes using the model constructed in Step 3
- `$sim_survival()`, `$sim_stateprobs()`, `$sim_qalys()`, `$sim_costs()`

4. Cost-effectiveness analysis

Complete R script



```
## ---- Overview ----
## @knitr R-setup
library("rcdd")
library("hesim")
library("data.table")
library("ggplot2")
library("flexsurv")
theme_set(theme_bw())
set.seed(101) # Make PSA reproducible
## ---- Model setup ----
## @knitr hesim_data
patients <- data.table(
  patient_id = 1:4,
  patient_wt = rep(1/4, 4), # Each patient has same weight
  age = c(45, 45, 65, 65),
  female = c(0, 1, 0, 1)
)
states <- data.table(
  state_id = c(1, 2),
  state_name = c("Stable", "Progression") # Non-death health states
)
strategies <- data.frame(
  strategy_id = 1:2,
  strategy_name = c("SOC", "New")
)
hesim_dat <- hesim_data(
  patients = patients,
  strategies = strategies,
  states = states
)
print(hesim_dat)
## @knitr labels
labs <- get_labels(hesim_dat)
## ---- Parameter estimation ----
## @knitr pfs_os_data
onc3 <- hesim::onc3[strategy_name != "New 1"]
onc3[, strategy_name := droplevels(strategy_name)]
levels(onc3$strategy_name) <- c("SOC", "New")
```

Tutorial

rcea **0.1.2**

Reference

Tutorials ▾

Slides



Partitioned Survival Model

2021-07-26

<https://hesim-dev.github.io/rcea/articles/05-psm.html>

Overview

While multi-state models can be used to estimate the parameters of a state transition model (STM) in a very flexible manner, data availability can make it difficult (or infeasible) to fit such a model. This is often the case when an evidence synthesis model based on summary level data is used to parameterize the STM. For example, in oncology, published articles of clinical trials often provide survival curves of progression-free survival (PFS) and overall survival (OS), but do not release information on time to event (and censoring) for each transition. In this setting partitioned survival analysis may consequently be a simpler approach.

We will use the same packages as in the ["Semi-Markov Multi-state Model"](#) tutorial.

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

set.seed(101) # Make PSA reproducible
```

Contents

Overview

Theory

Simulation

Cost-effectiveness analysis

Theory

An 3-state partitioned survival model (PSM) simulates the probability that a patient is in each of 3 distinct health states at a given point of time when treated by a particular therapy. State membership is estimated from 2 survival curves (e.g., PFS and OS) using an "area under the

Exercise 5: Partitioned survival model

- *Review and run the R-script “05-psm.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

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

*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 if¹

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

¹Only 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 $k < ICER$.

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

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 with `hesim`

- `hesim` can be used to perform CEA and summarize decision uncertainty
 - Other R packages such as BCEA, dampack, SAVI, and EVSI could also be considered, especially for more advanced value of information analysis
- Implementation via 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
- Both are “generic” functions that work with (i) data frame like objects or (ii) “ce” objects

Cost-effectiveness analysis with `hesim`

```
markov_hesim_ce <- readRDS("markov-cohort-hesim-ce_sim.rds")
```

```
markov_hesim_ce
```

```
$costs
  category dr sample strategy_id      costs grp_id
1: treatment 0.03     1             1 52301.47    1
2: treatment 0.03     1             2 323892.01    1
3: treatment 0.03     2             1 49767.92    1
4: treatment 0.03     2             2 322401.36    1
5: treatment 0.03     3             1 52661.20    1
---
```

```
5996:   total 0.03    998            2 434677.65    1
5997:   total 0.03    999            1 165176.48    1
5998:   total 0.03    999            2 446988.07    1
5999:   total 0.03   1000            1 79522.29    1
6000:   total 0.03   1000            2 360039.77    1
```

```
$qalys
  dr sample strategy_id      qalys grp_id
1: 0.03     1             1 20.90995    1
2: 0.03     1             2 22.24340    1
3: 0.03     2             1 20.77562    1
4: 0.03     2             2 23.51086    1
5: 0.03     3             1 20.89219    1
---
```

```
1996: 0.03    998            2 24.15825    1
1997: 0.03    999            1 20.88165    1
1998: 0.03    999            2 23.63537    1
1999: 0.03   1000            1 20.77720    1
2000: 0.03   1000            2 23.02164    1
```

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

```
cea_pw_out <- cea_pw(markov_hesim_ce, comparator = 1,
                        dr_qalys = .03, dr_costs = .03,
                        k = wtp
)
```

```
cea_out <- cea(markov_hesim_ce,
                  dr_qalys = .03, dr_costs = .03,
                  k = wtp
)
```

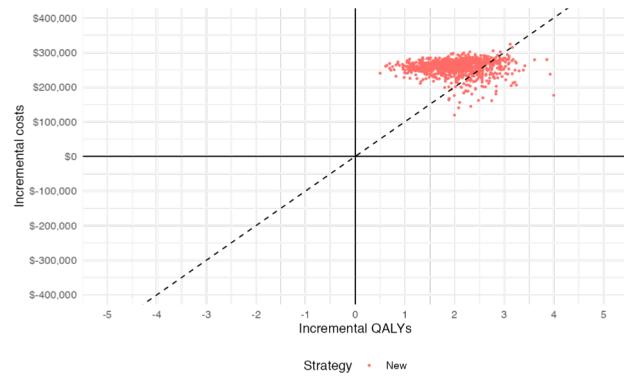
Incremental cost-effectiveness ratio with **hesim**

```
icer(cea_pw_out, wtp = 50000) %>%  
  format()
```

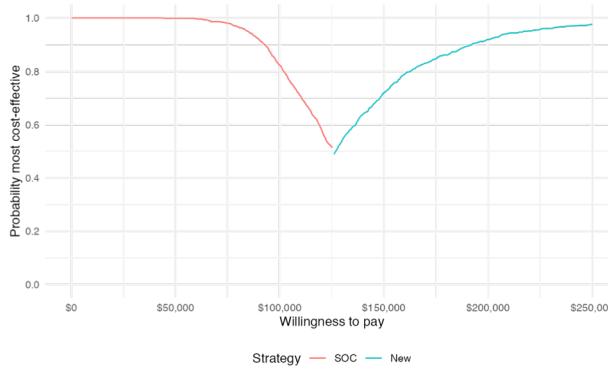
	Outcome	New
1: Incremental QALYs		2.09 (1.04, 3.14)
2: Incremental costs	261,594	(212,238, 291,654)
3: Incremental NMB	-156,981	(-215,676, -83,866)
4: ICER		125,029

Representing decision uncertainty with hesim

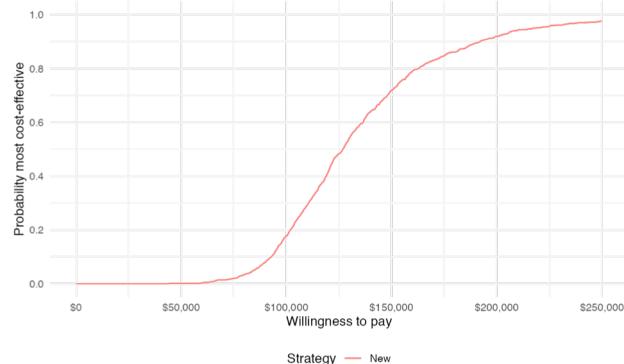
plot_ceplane(cea_pw_out)



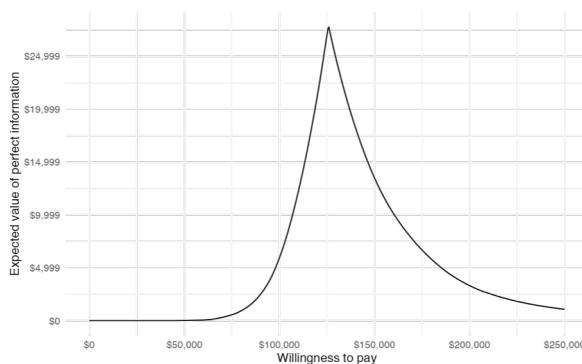
plot_ceaf(cea_out)



plot_ceac(cea_pw_out)



plot_evpri(cea_out)



Complete R script

```
06-cea.R x
Source on Save | Run | Source | Overview ▾
1 ## ---- Overview -----
2 ## @knitr R-setup
3 library("hesim")
4 library("ggplot2")
5 library("magrittr")
6 theme_set(theme_minimal()) # Set ggplot2 theme
7
8 ## ---- Application -----
9 ## @knitr load-ce
10 markov_ce <- readRDS("markov-cohort-ce_sim.rds")
11 markov_ce
12
13 ## @knitr load-hesim-ce
14 markov_hesim_ce <- readRDS("markov-cohort-hesim-ce_sim.rds")
15 hesim_dat <- readRDS("markov-cohort-hesim_data.rds")
16 markov_hesim_ce
17
18 ## @knitr conduct-cea
19 wtp <- seq(0, 250000, 500) # Willingness to pay per QALY
20 cea_pw_out <- cea_pw(markov_hesim_ce,
21                         comparator = 1, # Comparator is SOC (ID = 1)
22                         dr_qalys = 0.03, dr_costs = 0.03,
23                         wtp)
24 cea_out <- cea(markov_hesim_ce,
25                   dr_qalys = 0.03, dr_costs = 0.03,
26                   k = wtp)
27
28 ## @knitr conduct-cea-default
29 cea_pw_out2 <- cea_pw(markov_ce, comparator = "SOC",
30                         k = wtp,
31                         sample = "sample", strategy = "strategy",
32                         e = "dqalys", c = "dcosts")
33 cea_pw_out$summary
34 cea_pw_out2$summary
35
36 ## ---- Incremental cost-effectiveness ratio -----
37 ## @knitr icer
38 labs <- get_labels(hesim_dat)
39 icer(cea_pw_out, wtp = 50000, labels = labs) %>%
40   format()
41
42 ## ---- Cost-effectiveness plane -----
43 ## @knitr ceplane-plot
44 plot_ceplane(cea_pw_out, k = 100000, labels = labs)
45
```

Tutorial

rcea **0.1.2**

Reference

Tutorials ▾

Slides



Cost-effectiveness Analysis

2021-07-26

Contents

Overview

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

CUSTOM PUBLISHING WITH GITHUB

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.

In this analysis we will perform a CEA given the output of model from the “[Semi-Markov Multi-state Model](#)” tutorial. We will use the CEA functions from `hesim` to summarize decision uncertainty and `ggplot2` for visualization. The CEA will be performed for a single target population, but you can review the `hesim` tutorial on [CEA](#) and the references therein for an example of CEA in the context of multiple subgroups.

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

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

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

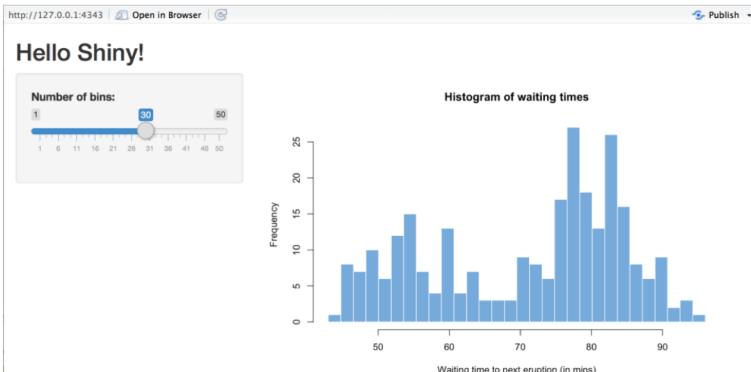
LESSON 1

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

Treatment Cost
200

PSA runs
1000

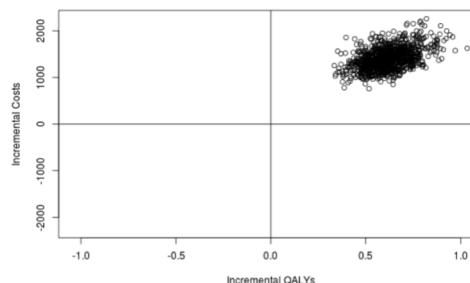
initial age
10 25 80

Run / update model

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](#)

Line of therapy

Treatment regimen

Sequence number

I II III IV V

1L

gefitinib

erlotinib

afatinib

dacomitinib

osimertinib

T790M Status + - + - + -

2L

gefitinib

erlotinib

afatinib

dacomitinib

osimertinib

pbdc

pbdc + bevacizumab

pbdc

pbdc + nivolumab

pbdc + pembrolizumab

pbdc + atezolizumab

pbdc + bevacizumab

pbdc + bevacizumab + nivolumab

Cost-effectiveness plane ⓘ

Sequence-comparator

I

II

III

IV

Expected outcomes

ICER

CE PLANE

CEAC

● Sequence II ● Sequence III ● Sequence IV — Value of a QALY

\$ 1,800,000

\$ 1,000,000

\$ 0

-\$ 1,000,000

-\$ 1,800,000

Incremental costs

-8.00 -6.00 -3.00 0.0 3.00 6.00 8.00

Incremental QALYs

Expected value of perfect information

 Restore defaults

Value of a QALY ⓘ \$ 150,000

 Save Run simulationR Code 

```
22 pats <- create_patients(n = 100)

23

24

25 txseq1 <- txseq(
26   first = c("gefitinib"),
27   second = c("osimertinib", "PBDC"),
28   second_plus = c("PBDC + atezolizumab", "PBDC + atezolizumab")
29 )

30

31 txseq2 <- txseq(
32   first = c("erlotinib"),
33   second = c("osimertinib", "PBDC"),
34   second_plus = c("PBDC + atezolizumab", "PBDC + atezolizumab")
35 )

36

37

38 txseq3 <- txseq(
39   first = c("afatinib"),
40   second = c("osimertinib", "PBDC"),
41   second_plus = c("PBDC + atezolizumab", "PBDC + atezolizumab")
42 )

43

44

45 txseq4 <- txseq(
46   first = c("osimertinib"),
47   second = c("PBDC", "PBDC"),
48   second_plus = c("PBDC + atezolizumab", "PBDC + atezolizumab")
```

pbdc + bevacizumab

pbdc + bevacizumab + nivolumab

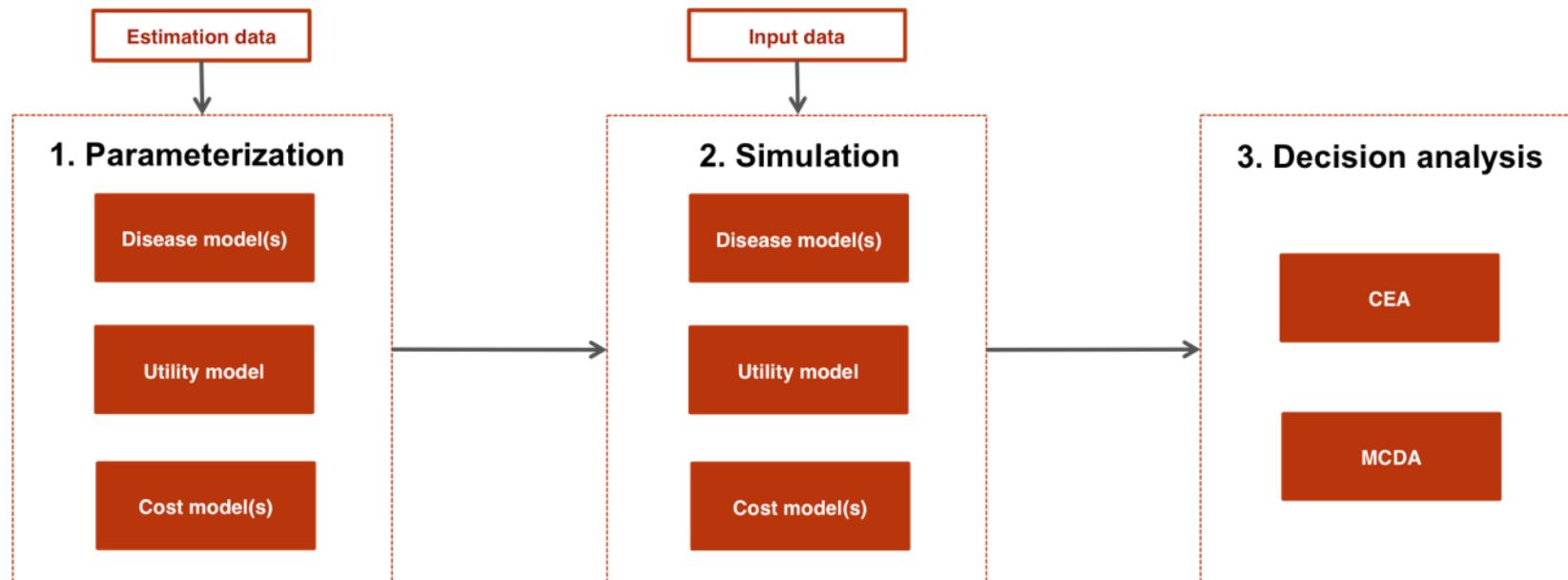


Thank you

devin.incerti@gmail.com

jeroen.jansen@ucsf.edu

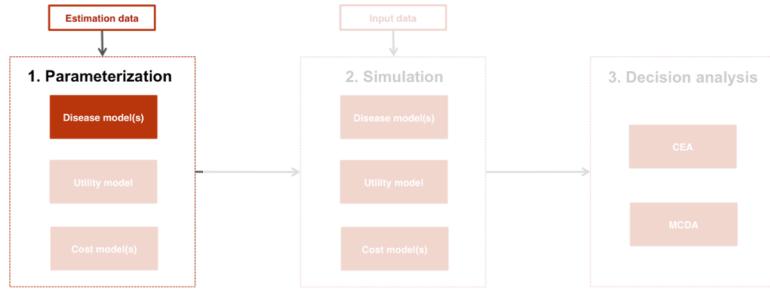
Appendix – Building a model with **hesim**



hesim overview

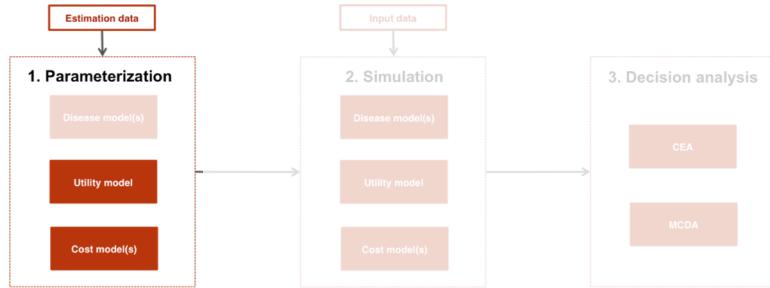
Economic model	<i>Object</i>	<i>Function</i>
Cohort discrete time state transition model (cDTSTM)	<code>CohortDtstm</code>	<code>create_CohortDtstm()</code> <code>CohortDtstm\$new()</code>
Individual-level continuous time state transition model (iCTSTM)	<code>IndivCtstm</code>	<code>IndivCtstm\$new()</code>
N-state partitioned survival model (PSM)	<code>Psm</code>	<code>Psm\$new()</code>

hesim - Parameterization



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

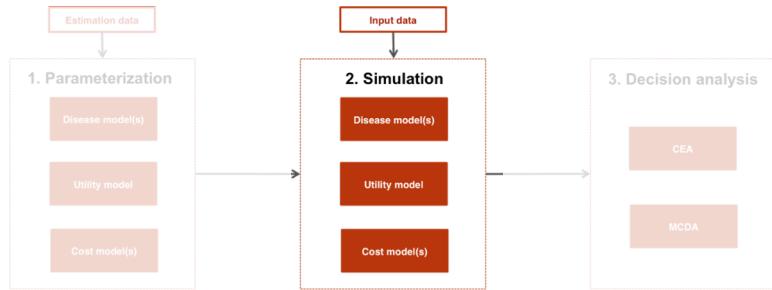
hesim - Parameterization



Cost and utility		
<i>Statistical model</i>	<i>Parameter Object</i>	<i>Model fit Function</i>
Predicted means	<code>tparams_mean</code>	<code>stateval_tb1()</code>
	<code>tparams_mean</code>	<code>define_model()</code>
Linear model	<code>params_lm</code>	<code>Stats::lm()</code>

hesim - Simulation

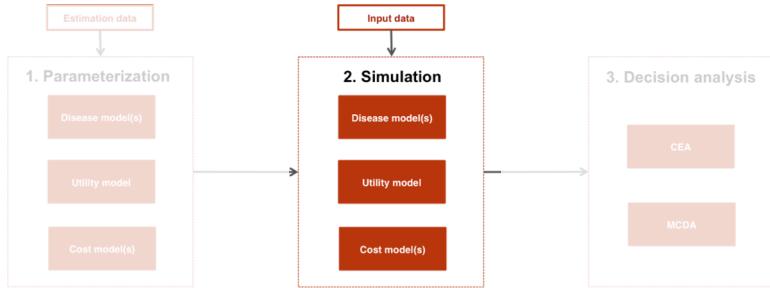
Constructing an economic model



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()