OPEN SOURCE SOFTWARE FOR BUILDING HEALTH ECONOMIC MODELS

Erik Dasbach
Economic and Data Sciences, Merck & Co.

Joseph Levy
Postdoctoral Fellow, University of Maryland School of Pharmacy

Fernando Alarid-Escudero
Post-Doctoral Associate, University of Minnesota

Learning Objectives

At the end of this workshop attendees should gain an understanding of how new software modeling packages can

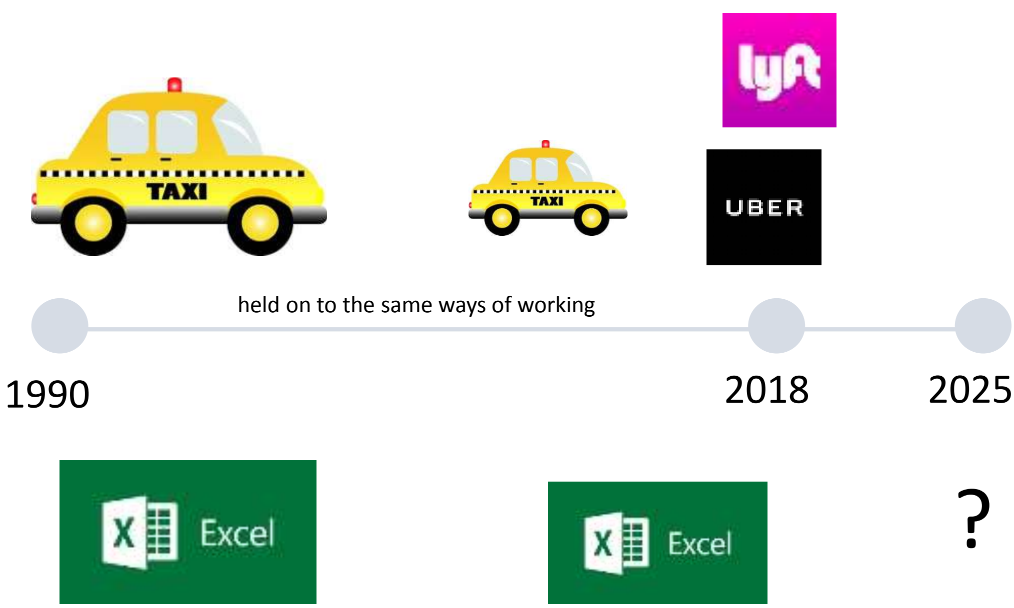
- accelerate model development,
- decrease rework, and
- improve model transparency and verification

Outline of Workshop

1. The case for why model development in our field needs to evolve
2. Markov models using open source software
3. Microsimulation modeling

Part 1

The case for why model development needs to evolve



lyft
UBER
held on to the same ways of working
1990
2018
2025
Excel
Excel
?

or
data scientist

1990

2018

"Why does the spreadsheet remain the model development platform of choice in the pharmacoeconomics field?"

Data Scientist to the Health Economist

Payers and reimbursement agencies and modelers favor the spreadsheet because

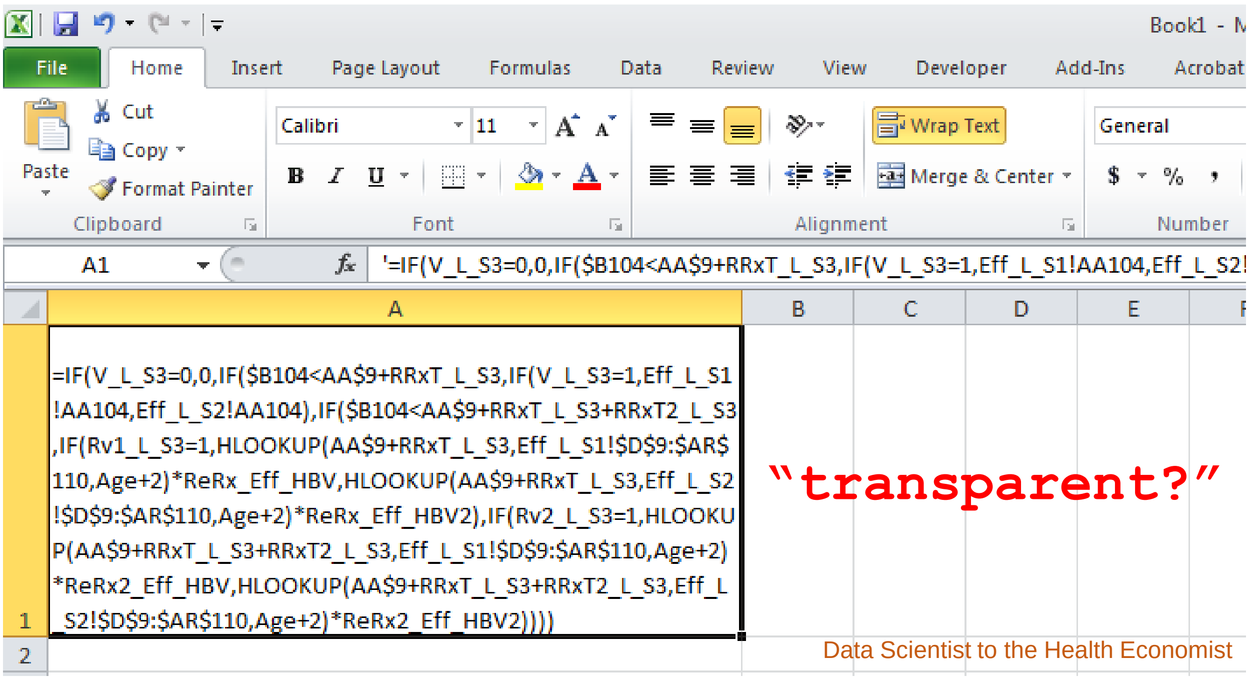
broad accessibility

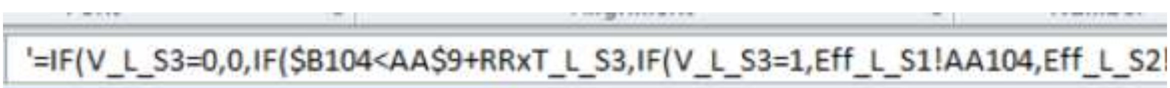
full stack platform

transparency

the ability to examine cell formula

Health Economist to the Data Scientist





“cell references are not transparent” — spaghetti code

“violates the DRY principle of coding” — Don’t Repeat Yourself

embraces WET code — Write Everything Twice+

“code is hard to reuse” — requires *shotgun surgery* to reuse

“lacks a testing framework”

“What do you mean by a testing framework?”

Health Economist to the Data Scientist

“How do you know your model is correct?”

Data Scientist to the Health Economist

“Well, I test edge cases and I have a colleague review the model.”

Health Economist to the Data Scientist

“A testing framework documents your tests?”

Data Scientist to the Health Economist

Unit Tests

- software testing method by which individual units of code are isolated and tested to demonstrate that the individual parts are correct (Kolowa & Huzinga, 2007)

	A	B	C	D
1	**Input Parameter Unit Tests**			
2				
3	**name**	**value**	**assertion**	**test**
4	annual drug cost	$1,500.00	$1,500.00	Pass
5	disease cost	$10,000.00	$10,000.00	Pass
6	disease probability	0.01	0.02	Fail
7	disease utility	0.80	0.80	Pass
8				
9				

Data Scientist to the Health Economist

Integration Tests

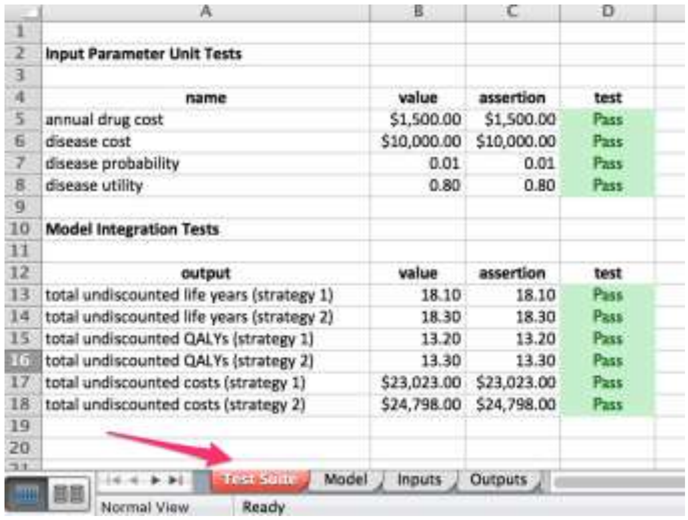
- the phase in software testing in which individual software modules are combined and tested as a group
  - https://en.wikipedia.org/wiki/Integration_testing

	A	B	C	D
1	**Model Integration Tests**			
2				
3	**output**	**value**	**assertion**	**test**
4	total undiscounted life years (strategy 1)	18.10	18.10	Pass
5	total undiscounted life years (strategy 2)	18.30	18.30	Pass
6	total undiscounted QALYs (strategy 1)	13.20	13.20	Pass
7	total undiscounted QALYs (strategy 2)	13.30	13.27	Fail
8	total undiscounted costs (strategy 1)	$23,023.00	$23,023.00	Pass
9	total undiscounted costs (strategy 2)	$24,798.00	$24,798.00	Pass
10				

Data Scientist to the Health Economist

Test Suite

- a collection of all the test cases



	A	B	C	D
1				
2	**Input Parameter Unit Tests**			
3				
4	**name**	**value**	**assertion**	**test**
5	annual drug cost	$1,500.00	$1,500.00	Pass
6	disease cost	$10,000.00	$10,000.00	Pass
7	disease probability	0.01	0.01	Pass
8	disease utility	0.80	0.80	Pass
9				
10	**Model Integration Tests**			
11				
12	**output**	**value**	**assertion**	**test**
13	total undiscounted life years (strategy 1)	18.10	18.10	Pass
14	total undiscounted life years (strategy 2)	18.30	18.30	Pass
15	total undiscounted QALYs (strategy 1)	13.20	13.20	Pass
16	total undiscounted QALYs (strategy 2)	13.30	13.30	Pass
17	total undiscounted costs (strategy 1)	$23,023.00	$23,023.00	Pass
18	total undiscounted costs (strategy 2)	$24,798.00	$24,798.00	Pass
19				
20				

Data Scientist to the Health Economist

Let me show you what you have been missing out on…

Excel

Excel

1990

2018

Agile

GitHub

RStudio

IDE

BASECASE

TDD

data viz

jupyter Notebook

EviDICE

hēRo3

IVI INNOVATION AND VALUE INITIATIVE

R

python

JavaScript

heemod

BCEA

DARTH

Data Scientist to the Health Economist

Examples of Software for Economic Evaluations

Part II

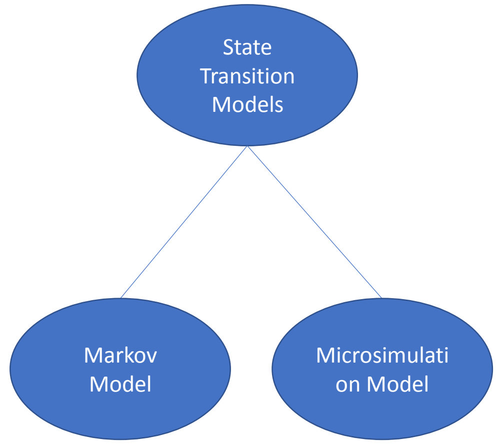
Joe Levy

Outline

- Briefly review State Transition Modeling
- Introduce HEEMOD and DICE for Markov Modeling
- Describe Sick Sicker Model
- Show syntax and model builds
- Compare anecdotal experiences

State Transition Models

- Representations of clinical scenarios by
  - Time in states
  - Transitions between states
  - Accrue costs and effects from being in states
  - Transition (and cost/effects) differentially by treatment
- Markov Cohort
  - Cohort transitions as percentage
- Microsimulation
  - Individuals progress with first order uncertainty



Siebert, Uwe, et al. "State-transition modeling: a report of the ISPOR-SMDM modeling good research practices task force-3." Value in Health 15.6 (2012): 812-820.

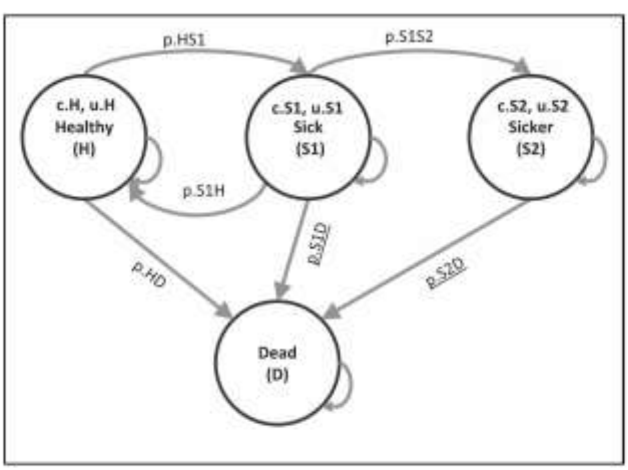
Software 1: HEEMOD

- Markov Models for Health Economic Evaluation (HEEMOD) R-Package
- Objective: Simple, declarative syntax to specify and execute Markov models and partitioned survival models
- Define Strategies, Model Parameters, Transitions, State Values
- Can perform deterministic and probabilistic sensitivity analysis
- Built in functions to discount, convert rates to probability, hazard, probability over time etc.
- Models are stored as objects, generate graphics in R (ggplot2)

Software 2: DICE

- Discretely Integrated Condition Event simulation (DICE).
- A modeling technique designed for general decision-analytic modeling, conceptualizes a disease process and its management in terms of **conditions** and **events**.
  - Conditions: Aspect of model that persist over time, have levels which can be modified by other conditions or events
  - Events: Aspects of the model that happen at any point in time, can effect level of conditions or other events
- Algorithm/engine which can construct markov, microsimulation and discrete event simulation.
- Algorithm has been implemented in excel, R and pyton

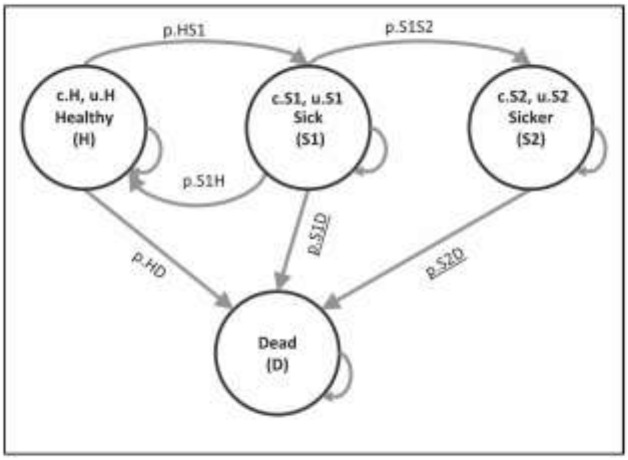
Example: Sick Sicker Markov



Krijkamp, Eline M., et al. "Microsimulation Modeling for Health Decision Sciences Using R: A Tutorial." *Medical Decision Making* 38.3 (2018): 400-422.

- Compare Treatment to No Treatment
- 4 State Model
- Treatment Modifies Cost of Sick, Sicker and Utility of Sick
- Transitions Probabilities are the Same between treatment groups
- Time horizon: 30 years

Example: Sick Sicker Markov



Parameter	Treat	No Treat
p.HS1	0.15	0.15
p.S1S2	0.105	0.105
p.S1H	0.5	0.5
p.HDie	0.005	0.005
RR.SickDie (vs H)	3	3
RR.SickerDie (vs H)	10	10
cost.H	2000	2000
cost.S1	4000	4000+12000
cost.S2	15000	15000+12000
Utility.H	1	1
utility.S1	0.75	0.95
Utility.S2	0.5	0.5
Discount Rate	3%	3%

define_transition

```
> Treat<- define_transition(state_names = c("healthy","sick","sicker","dead"),
+                                            C      ,p.HS1,0      ,p.HD,
+                                            p.S1H,C      ,p.S1S2,p.S1D,
+                                            0      ,0      ,C      ,p.S2D,
+                                            0      ,0      ,0      ,1)
> Treat
A transition matrix, 4 states.

        healthy sick  sicker dead
healthy C       p.HS1        p.HD
sick    p.S1H   C     p.S1S2 p.S1D
sicker                C      p.S2D
dead                         1
```

Parameter	No Treat	Treat
p.HS1	0.15	0.15
p.S1S2	0.105	0.105
p.S1H	0.5	0.5
p.HDie	0.005	0.005
RR.SickDie (vs H)	3	3
RR.SickerDie (vs H)	10	10
cost.H	2000	2000
cost.S1	4000	4000+12000
cost.S2	15000	15000+12000
Utility.H	1	1
utility.S1	0.75	0.95
Utility.S2	0.5	0.5
Discount Rate	3%	3%

define_parameters

```r
param<-define_parameters(
  dr=0.03,
  p.HD = 0.005, # probability to die when healthy
  p.HS1 = 0.15, # probability to become sick when healthy
  p.S1H =  0.5, # probability to become healthy when sick
  p.S1S2 = 0.105, # probability to become sicker when sick
  rr.S1 = 3 , # rate ratio of death when sick vs healthy
  rr.S2 = 10,  # rate ratio of death when sicker vs healthy
  r.HD = -log(1 - p.HD), # rate of death when healthy
  r.S1D = rr.S1 * r.HD, # rate of death when sick
  r.S2D = rr.S2 * r.HD, # rate of death when sicker
  p.S1D = 1-exp(-r.S1D), # probability to die when sick
  p.S2D = 1-exp(-r.S2D), # probability to die when sicker

  # Cost and utility inputs
  c.H = 2000, # cost of remaining one cycle healthy
  c.S1 = 4000 ,# cost of remaining one cycle sick
  c.S2 = 15000 ,# cost of remaining one cycle sicker
  c.Trt = 12000, # cost of treatment (per cycle)
  u.H = 1,
  u.S1 = .75,
  u.S2 = .5,
  u.Trt = .95
  )
```

Parameter	No Treat	Treat
p.HS1	0.15	0.15
p.S1S2	0.105	0.105
p.S1H	0.5	0.5
p.HDie	0.005	0.005
RR.SickDie (vs H)	3	3
RR.SickerDie (vs H)	10	10
cost.H	2000	2000
cost.S1	4000	4000+12000
cost.S2	15000	15000+12000
Utility.H	1	1
utility.S1	0.75	0.95
Utility.S2	0.5	0.5
Discount Rate	3%	3%

define_parameters


```r
param<-define_parameters(
  dr=0.03,
  p.HD = 0.005, # probability to die when healthy
  p.HS1 = 0.15, # probability to become sick when healthy
  p.S1H =  0.5, # probability to become healthy when sick
  p.S1S2 = 0.105, # probability to become sicker when sick
  rr.S1 = 3 , # rate ratio of death when sick vs healthy
  rr.S2 = 10,  # rate ratio of death when sicker vs healthy
  r.HD = -log(1 - p.HD), # rate of death when healthy
  r.S1D = rr.S1 * r.HD, # rate of death when sick
  r.S2D = rr.S2 * r.HD, # rate of death when sicker
  p.S1D = 1-exp(-r.S1D), # probability to die when sick
  p.S2D = 1-exp(-r.S2D), # probability to die when sicker

  # Cost and utility inputs
  c.H = 2000, # cost of remaining one cycle healthy
  c.S1 = 4000 ,# cost of remaining one cycle sick
  c.S2 = 15000 ,# cost of remaining one cycle sicker
  c.Trt = 12000, # cost of treatment (per cycle)
  u.H = 1,
  u.S1 = .75,
  u.S2 = .5,
  u.Trt = .95
  )
```


Parameter	No Treat	Treat
p.HS1	0.15	0.15
p.S1S2	0.105	0.105
p.S1H	0.5	0.5
p.HDie	0.005	0.005
p.S1Die	0.01492512	0.01492512
p.S2Die	0.04888987	0.04888987
cost.H	2000	2000
cost.S1	4000	4000+12000
cost.S2	15000	15000+12000
Utility.H	1	1
utility.S1	0.75	0.95
Utility.S2	0.5	0.5
Discount Rate	3%	3%

define_state

```
healthy<-define_state(cost=discount(c.H,dr),utility=discount(u.H,dr))

sick_t<-define_state(cost=discount(c.S1+c.Trt,dr),utility=discount(u.Trt,dr))

sicker_t<-define_state(cost=discount(c.S2+c.Trt,dr),utility=discount(u.S2,dr))

sick<-define_state(cost=discount(c.S1,dr),utility=discount(u.S1,dr))

sicker<-define_state(cost=discount(c.S2,dr),utility=discount(u.S2,dr))

dead<-define_state(cost=0,utility=0)
```

Parameter	No Treat	Treat
p.HS1	0.15	0.15
p.S1S2	0.105	0.105
p.S1H	0.5	0.5
p.HDie	0.005	0.005
p.S1Die	0.01492512	0.01492512
p.S2Die	0.04888987	0.04888987
cost.H	2000	2000
cost.S1	4000	4000+12000
cost.S2	15000	15000+12000
Utility.H	1	1
utility.S1	0.75	0.95
Utility.S2	0.5	0.5
Discount Rate	3%	3%

define_strategy

```
strat_trt<-define_strategy(
  transition=transition_Treat,healthy=healthy,sick=sick_t,
  sicker=sicker_t,dead=dead)

strat_ctrl<-define_strategy(
  transition=transition_NoTreat,healthy=healthy,sick=sick,
  sicker=sicker,dead=dead)
```


Parameter	No Treat	Treat
p.HS1	0.15	0.15
p.S1S2	0.105	0.105
p.S1H	0.5	0.5
p.HDie	0.005	0.005
p.S1Die	0.01492512	0.01492512
p.S2Die	0.04888987	0.04888987
cost.H	2000	2000
cost.S1	4000	4000+12000
cost.S2	15000	15000+12000
Utility.H	1	1
utility.S1	0.75	0.95
Utility.S2	0.5	0.5
Discount Rate	3%	3%

Run_model

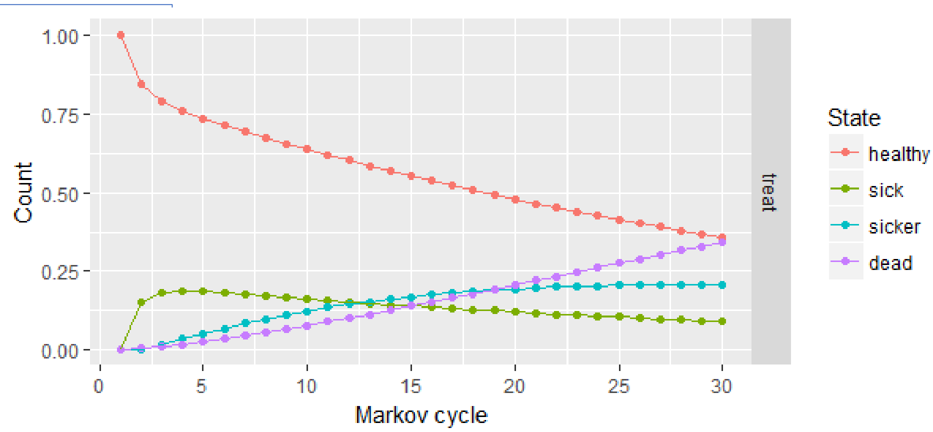
```
model_ss<-run_model(NoTreat=strat_ctrl, Treat=strat_trt, cycles=30, method="end",cost=cost,
                    effect=utility,parameters = param, init = c(1,0,0,0))
```

```
> model_ss
2 strategies run for 30 cycles.

Initial state counts:

healthy = 1
sick = 0
sicker = 0
dead = 0

Counting method: 'end'.

Values:

              cost  utility
NoTreat   72103.75 15.17023
Treat    134422.99 15.70836

Efficiency frontier:

NoTreat -> Treat

Differences:

      Cost Diff. Effect Diff.   ICER    Ref.
Treat   62319.24    0.5381302 115807 NoTreat
```

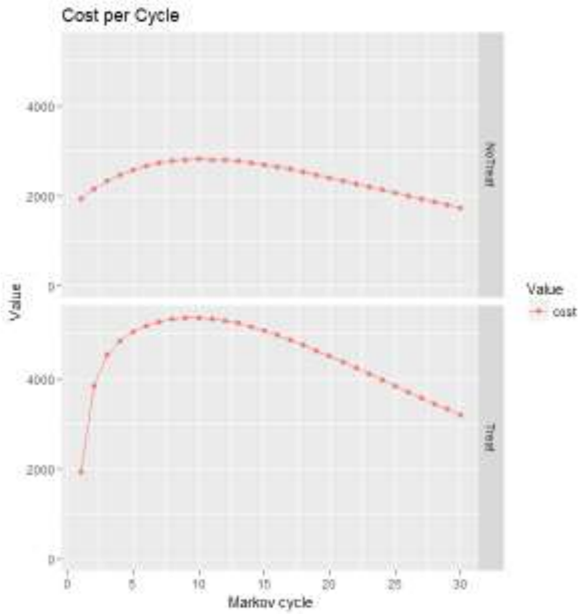
Run_model

```
model_ss<-run_model(NoTreat=strat_ctrl, Treat=strat_trt, cycles=30, method="end",cost=cost,
                    effect=utility,parameters = param, init = c(1,0,0,0))
```

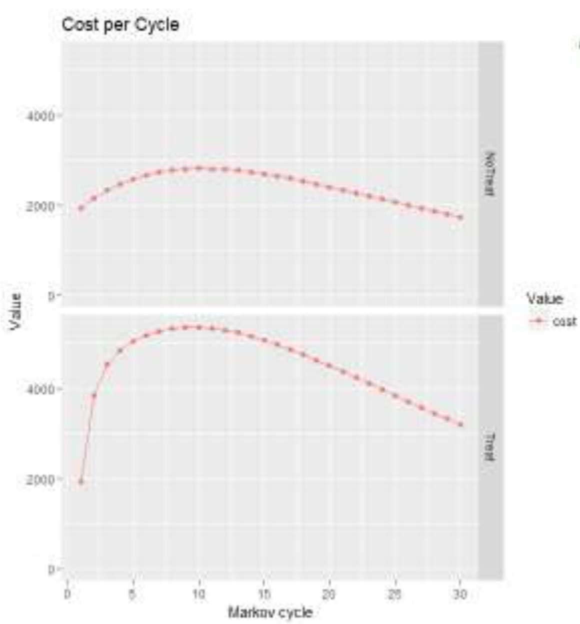
```
> model_ss
2 strategies run for 30 cy

Initial state counts:

healthy = 1
sick = 0
sicker = 0
dead = 0

Counting method: 'end'.

Values:

              cost  utilit
NoTreat   72103.75 15.1702
Treat    134422.99 15.7083

Efficiency frontier:

NoTreat -> Treat

Differences:

      Cost Diff. Effect Diff.   ICER    Ref.
Treat   62319.24    0.5381302 115807 NoTreat
```

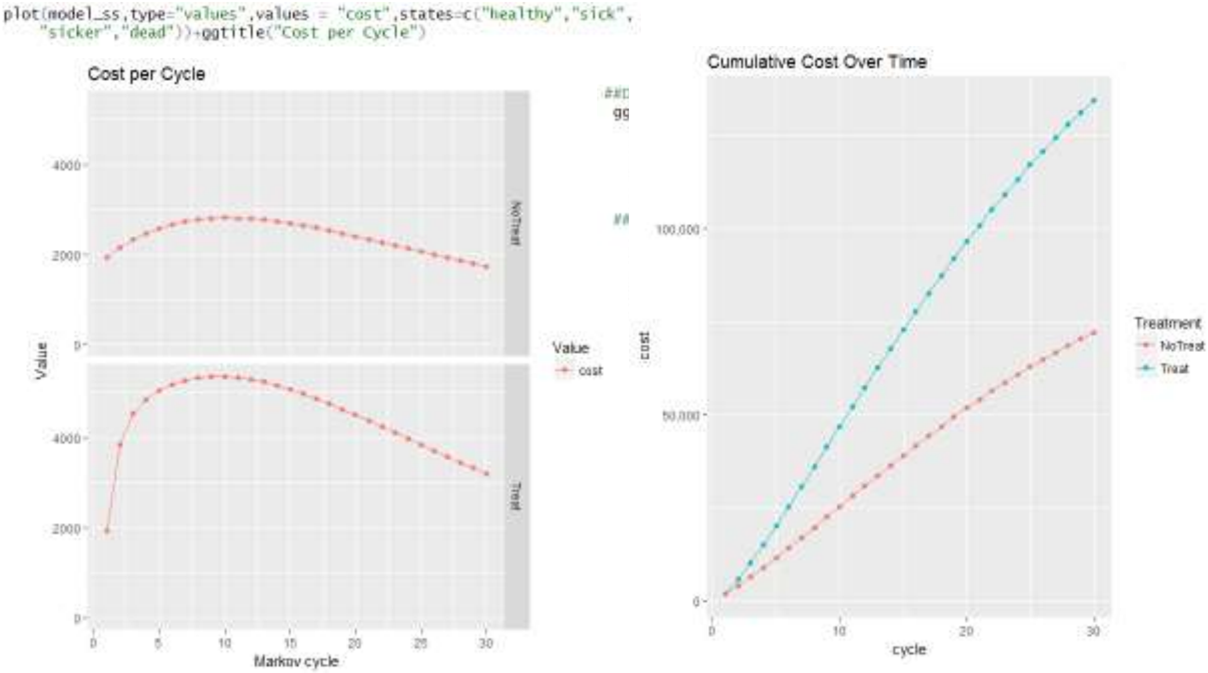
```r
plot(model_ss,type="values",values = "cost",states=c("healthy","sick",
     "sicker","dead"))+ggtitle("Cost per Cycle")
```




```r
plot(model_ss,type="values",values = "cost",states=c("healthy","sick",
     "sicker","dead"))+ggtitle("Cost per Cycle")
```

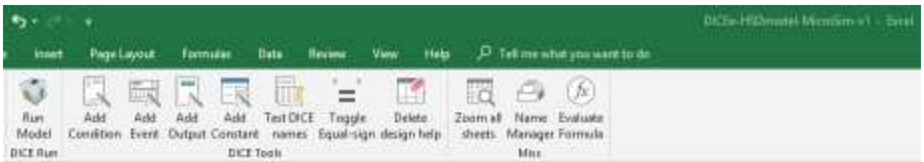

```r
##Data Frame
 gg<-data.frame(
   cycle=c(model_ss$eval_strategy_list$Treat$values$markov_cycle,
           model_ss$eval_strategy_list$Treat$values$markov_cycle),
   cost= c(cumsum(model_ss$eval_strategy_list$Treat$values$cost),
           cumsum(model_ss$eval_strategy_list$NoTreat$values$cost)),
   Treatment=c(rep("Treat",30),rep("NoTreat",30)))
 ##Plot
   ggplot(data=gg, aes(x=cycle, y=cost, group=Treatment)) +
   geom_line(aes(color=Treatment))+
   geom_point(aes(color=Treatment))+
   scale_y_continuous(labels = comma)+
   ggtitle("Cumulative Cost Over Time")
```



DICE

- Dice is a way to conceptualize any model type, at its core it is an algorithm that iteratively evaluates conditions and events
- Conditions and Events can be coerced to recreate Markov or microsimulations
- Will show only Excel implementation
- DICE Demo workbooks, and the engine available at Evidera.com, several papers and demos serve as starting points to comprehend syntax

DICE-Conditions



DICE-Conditions

Conditions:

	Name	CurCond Level	Initial Level
Process	ID		0
	Time		0
	TimeHorizon		30
	IntervNum		1
	Cycle		1
States	Healthy		100%
	Sick		0
	Sicker		0
	Dead		0
Transitions	HealthyDead		0
	HealthySick		0
	SickHealthy		0
	SickSicker		0
	SickDead		0
	SickerDead		0
Probabilities	pHD		0.005
	pHSick		0.15
	pSickH		0.5
	pSickDead		0.014925125
	pSickerDead		0.04888987
	pSickSicker		0.105
Costs	cH		2000
	cSick		4000
	cSicker		15000
	cTrt		12000
Utilities	uH		1
	uSick		0.75
	uSicker		0.5
	uTrt		0.95
Process	DiscountRate		3.00%
	NextEventTime		0
	NextEvent		1

Dice-Events

All Events

Name	CurEventTime	Initial Time To Event	Table
Start	99999999	Now	tblStart
Transition	31	Cycle	tblTransition
End	30	TimeHorizon	tblEnd

Event: Start

Type	Name	Expression	Notes
Condition	Time	Start	To reset the clock to zero
Event	Start	Never	To avoid infinite loop
Output	Tmt	CHOOSE(IntervNum,"NoTreat","Treat")	
Output	QALYs	0	Initialize to zero
Output	Cost	0	Initialize to zero
Output	dQALYs	0	Initialize to zero
Output	dCosts	0	Initialize to zero
Condition	HealthyDead	0	Set according to treatment
Condition	HealthySick	0	
Output	CostTmt	0	
Condition	NextEventTime	MIN(CurEventTime)	Find next event time
Condition	NextEvent	MATCH(NextEventTime,CurEventTime,0)	Find next event

Event: End

Type	Name	Expression	Notes
Condition	Time	End	To update the clock

Event: Transition (name: tblTransition)

Type	Name	Expression	Notes
Condition	Time	Transition	
Output	QALYs	QALYs+(Healthy*uH+Sick*Choose(IntervNum,uSick,uTrt)+Sicker*uSicker)	
Output	Cost	Cost+(Healthy*cH+Sick*CHOOSE(IntervNum,cSick,cSick+cTrt)+Sicker*CHOOSE(IntervNum,cSicker,cSicker+cTrt))	
Condition	HealthyDead	pHD*Healthy	
Condition	HealthySick	pHSick*Healthy	
Condition	SickHealthy	pSickH*Sick	
Condition	SickSicker	pSickSicker*Sick	
Condition	SickDead	pSickDead*Sick	
Condition	SickerDead	pSickerDead*Sicker	
Condition	Healthy	Healthy-HealthySick-HealthyDead+SickHealthy	
Condition	Sick	Sick+HealthySick-SickHealthy-SickDead-SickSicker	
Condition	Sicker	Sicker+SickSicker-SickerDead	
Condition	Dead	Dead+SickDead+SickerDead+HealthyDead	
Event	Transition	Time+Cycle	
Condition	NextEventTime	Min(CurEventTime)	Find next event time
Condition	NextEvent	Match(NextEventTime,CurEventTime,0)	Find next event

Dice-Events

Event: Transition (name: tblTransition)

Type	Name	Expression
Condition	Time	Transition
Output	QALYs	QALYs+(Healthy*uH+Sick*Choose(IntervNum,uSick,uTrt)+Sicker*uSicker)
Output	Cost	Cost+(Healthy*cH+Sick*CHOOSE(IntervNum,cSick,cSick+cTrt)+Sicker*CHOOSE(IntervNum,cSicker,cSicker+cTrt))
Output	dQALYs	dQALYs+(Healthy*uH+Sick*Choose(IntervNum,uSick,uTrt)+Sicker*uSicker)/(1+DiscountRate)^Time
Output	dCosts	dCosts+(Healthy*cH+Sick*CHOOSE(IntervNum,cSick,cSick+cTrt)+Sicker*CHOOSE(IntervNum,cSicker,cSicker+cTrt))/(1+DiscountRate)^Time
Condition	HealthyDead	pHD*Healthy
Condition	HealthySick	pHSick*Healthy
Condition	SickHealthy	pSickH*Sick
Condition	SickSicker	pSickSicker*Sick
Condition	SickDead	pSickDead*Sick
Condition	SickerDead	pSickerDead*Sicker
Condition	Healthy	Healthy-HealthySick-HealthyDead+SickHealthy
Condition	Sick	Sick+HealthySick-SickHealthy-SickDead-SickSicker
Condition	Sicker	Sicker+SickSicker-SickerDead
Condition	Dead	Dead+SickDead+SickerDead+HealthyDead
Event	Transition	Time+Cycle
Condition	NextEventTime	Min(CurEventTime)
Condition	NextEvent	Match(NextEventTime,CurEventTime,0)

HEEMOD

```
Values:

              cost  utility
NoTreat   72103.75 15.17023
Treat    134422.99 15.70836

Efficiency frontier:

NoTreat -> Treat

Differences:

       Cost Diff. Effect Diff.   ICER    Ref.
Treat    62319.24    0.5381302 115807 NoTreat
```

Runtime: 0.33 Seconds

DICE

Tmt	QALYs	Cost	dQALYs	dCosts
NoTreat	22.20284	114982.7	15.17023	72103.75
Treat	22.99945	214738.7	15.70836	134423

Runtime: <1 Second

HEEMOD

Advantages	Disadvantages
-Easy to learn (especially for R Users) -Replicated examples from Decision Modelling for Health Economic Evaluation -Can write model with scripting only OR using tabular inputs (excel based) -Unit tests are built into code, fully transparent -Sensitivity analysis, half-cycle correction, discounting, rate to probability, all easy to implement -Can be run in parallel (multi-core)	-Syntax may be hard to learn for non-R users -Markov Only, without plans to implement additional features -Probabilistic Sensitivity Analysis is slow depending on complexity (3-6 minutes)

DICE

Advantages	Disadvantages
-Any type of decision analytic model can be built this way—unifying -Familiar Excel syntax (if using) -Structure and implementation are consistent across model types	-Less worked examples -PSA is slow (excel) -Similar pitfalls to excel transcription errors -Similar time to learn DICE than general excel, setting up PSA similar -No graphics