Unlocking Discrete Event Simulation Modelling in R using WARDEN

Javier Sanchez Alvarez

Associate Director, Value Analytics Center, AbbVie

June 6th 2025

Why WARDEN?

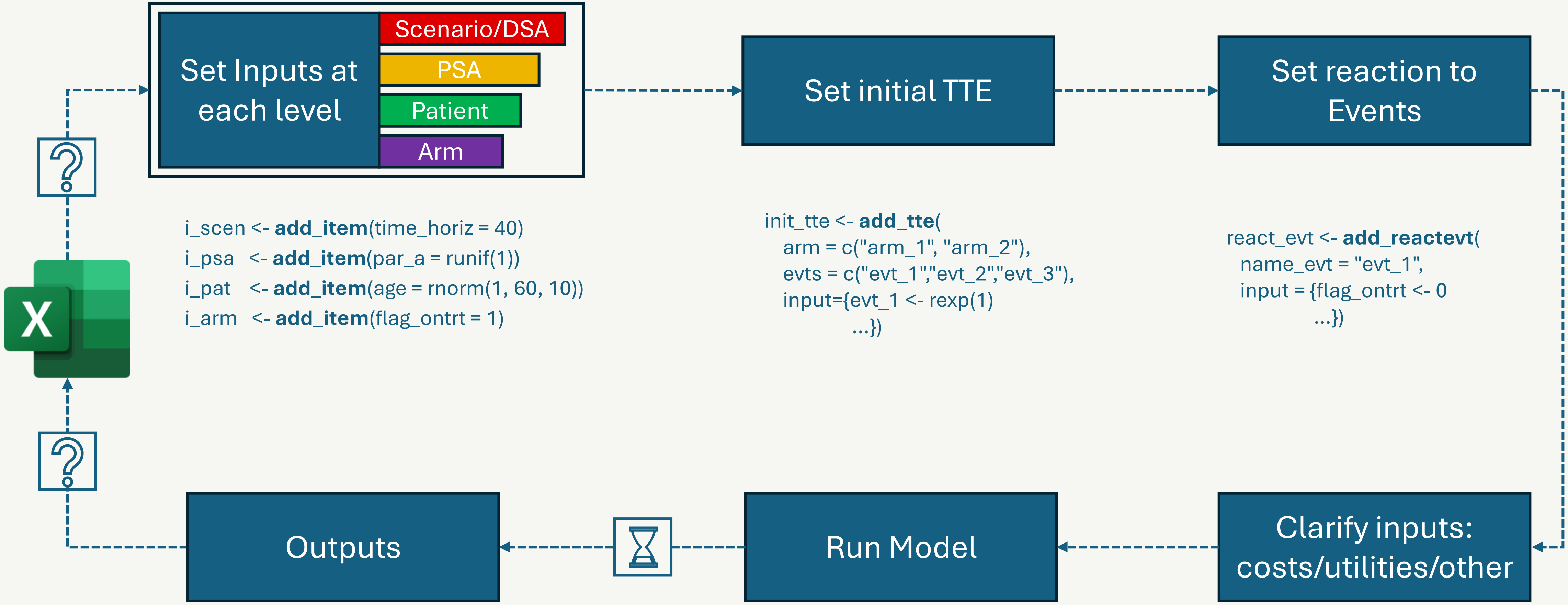
- DES and R are increasingly relevant for HTA submissions, however existing packages
  - Have non-HE focus
  - May require knowledge of C++
  - Require major work to use in HTA submissions

- WARDEN: DES without resource constraints - R package (CRAN)
  - Evolved over Roche, Evidera and AbbVie thanks to the support for **open source and copyleft** license
  - HE focus (comparison of interventions, QALYs, LYs, Costs...)
  - **HTA submission ready** (full toolset: PSA, OWSA, scenarios...)
  - “Beginner” friendly (no C++), grammar tidyverse style (pipes)
  - Focus on **flexibility** and **transparency**

QR code to WARDEN website

How to work with WARDEN?

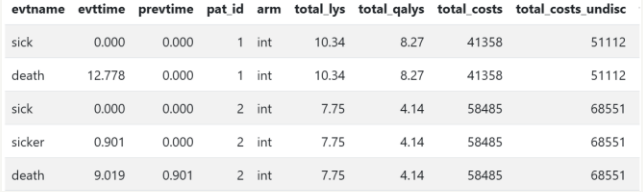
Nested Loops and Delayed Execution



```
i_scen <- add_item(time_horiz = 40)
i_psa   <- add_item(par_a = runif(1))
i_pat   <- add_item(age = rnorm(1, 60, 10))
i_arm   <- add_item(flag_ontrt = 1)
```

```
init_tte <- add_tte(
   arm = c("arm_1", "arm_2"),
   evts = c("evt_1","evt_2","evt_3"),
   input={evt_1 <- rexp(1)
              ...})
```

```
react_evt <- add_reactevt(
  name_evt = "evt_1",
  input = {flag_ontrt <- 0
              ...})
```



evtname	evttime	prevtime	pat_id	arm	total_lys	total_qalys	total_costs	total_costs_undisc
sick	0.000	0.000	1	int	10.34	8.27	41358	51112
death	12.778	0.000	1	int	10.34	8.27	41358	51112
sick	0.000	0.000	2	int	7.75	4.14	58485	68551
sicker	0.901	0.000	2	int	7.75	4.14	58485	68551
death	9.019	0.901	2	int	7.75	4.14	58485	68551

```
run_sim(
i_scen,
...,
init_tte,
...)
```

```
util_ongoing   <- "q_default“
cost_ongoing <- "c_default“
other_instant <- “o_inst_ob”
```

QR code to WARDEN website

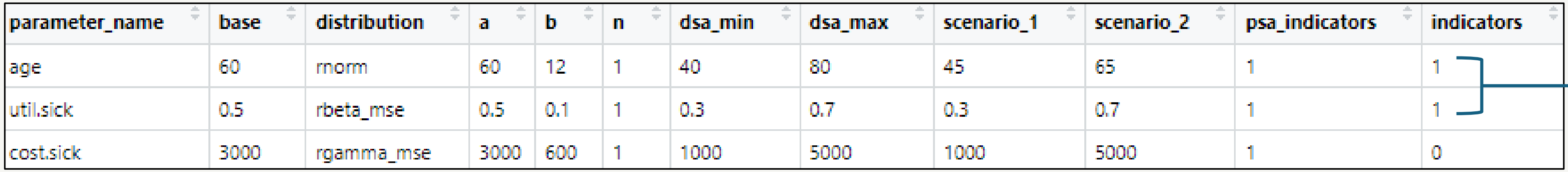
Advantages

- Massive improvement over descem (presented in 2021)
  - **2-3x faster**
  - New analyses **(deterministic/probabilistic): base case, scenarios**, **OWSA**
  - Numerous **helper functions** (automatic input handling, profile replication, reaction summary...)
  - **Statistical functions** for DES (conditional quantiles, luck adjustment...)
  - **Debug mode** with **log** file (even on error)
  - Fully **documented** with many examples
  - Improved outputs
    - Option: **aggregation** levels: full IPD, IPD but aggregating events, only summary
    - Option: evolution of cohort **outputs over time**

QR code to WARDEN website

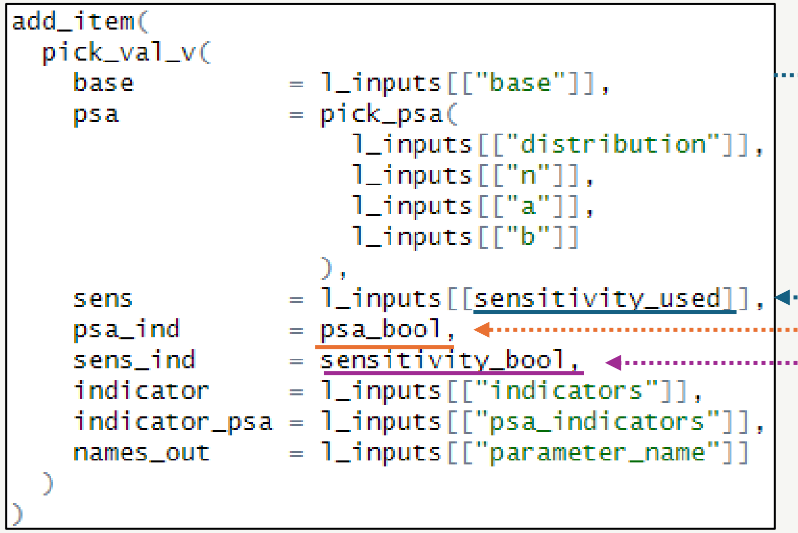
Showcase

How to handle deterministic/PSA/OWSA/Scenarios in a single function?

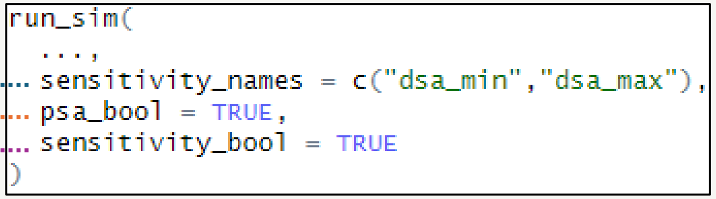


parameter_name	base	distribution	a	b	n	dsa_min	dsa_max	scenario_1	scenario_2	psa_indicators	indicators
age	60	rnorm	60	12	1	40	80	45	65	1	1
util.sick	0.5	rbeta_mse	0.5	0.1	1	0.3	0.7	0.3	0.7	1	1
cost.sick	3000	rgamma_mse	3000	600	1	1000	5000	1000	5000	1	0

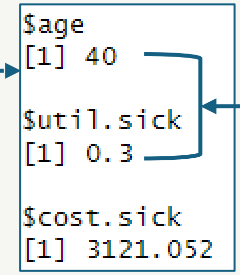
- Once model is written, user should only interact with **run_sim()** to change analysis type
- **pick_val_v() automatically** obtains the right values and iterates as needed over all type of analyses
- Accepts **vector parameters** (e.g., multivariate normal) and **multiple parameters covaried** (e.g., in DSA)



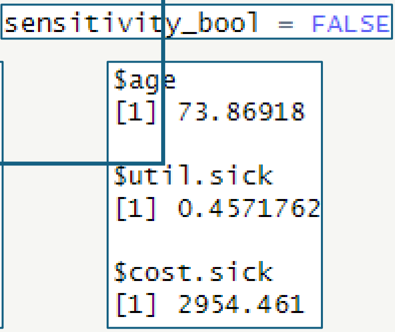
```
add_item(
  pick_val_v(
    base             = l_inputs[["base"]],
    psa              = pick_psa(
                          l_inputs[["distribution"]],
                          l_inputs[["n"]],
                          l_inputs[["a"]],
                          l_inputs[["b"]]
                        ),
    sens             = l_inputs[[sensitivity_used]],
    psa_ind          = psa_bool,
    sens_ind         = sensitivity_bool,
    indicator        = l_inputs[["indicators"]],
    indicator_psa = l_inputs[["psa_indicators"]],
    names_out        = l_inputs[["parameter_name"]]
  )
)
```



```
run_sim(
  ...,
  sensitivity_names = c("dsa_min","dsa_max"),
  psa_bool = TRUE,
  sensitivity_bool = TRUE
)
```



```
$age
[1] 40

$util.sick
[1] 0.3

$cost.sick
[1] 3121.052
```



```
sensitivity_bool = FALSE
```

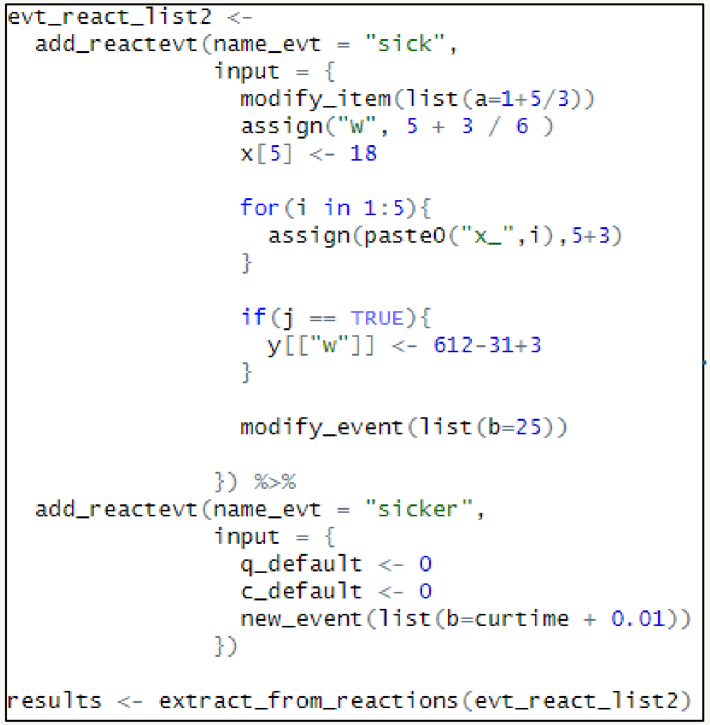
```
$age
[1] 73.86918

$util.sick
[1] 0.4571762

$cost.sick
[1] 2954.461
```

QR code to WARDEN website

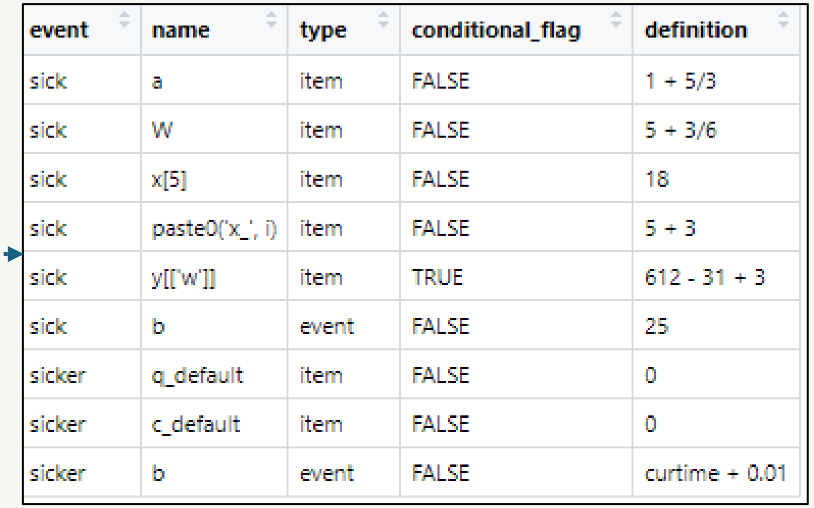
Showcase

How to summarize all interactions in reactions?

- If the model is complex, event reactions can become hard to read
  - **Summarizing all items/events interactions by event**, with **definitions** and whether they are **conditional** can be useful
- **extract_from_reactions() summarizes** all **interactions** using **abstract syntax trees** for easy inspection/debugging/documentation



```
evt_react_list2 <-
  add_reactevt(name_evt = "sick",
               input = {
                 modify_item(list(a=1+5/3))
                 assign("W", 5 + 3 / 6 )
                 x[5] <- 18

                 for(i in 1:5){
                   assign(paste0("x_",i),5+3)
                 }

                 if(j == TRUE){
                   y[["w"]] <- 612-31+3
                 }

                 modify_event(list(b=25))

               }) %>%
  add_reactevt(name_evt = "sicker",
               input = {
                 q_default <- 0
                 c_default <- 0
                 new_event(list(b=curtime + 0.01))
               })

results <- extract_from_reactions(evt_react_list2)
```



event	name	type	conditional_flag	definition
sick	a	item	FALSE	1 + 5/3
sick	W	item	FALSE	5 + 3/6
sick	x[5]	item	FALSE	18
sick	paste0('x_', i)	item	FALSE	5 + 3
sick	y[['w']]	item	TRUE	612 - 31 + 3
sick	b	event	FALSE	25
sicker	q_default	item	FALSE	0
sicker	c_default	item	FALSE	0
sicker	b	event	FALSE	curtime + 0.01

QR code to WARDEN website

Conclusion

- WARDEN **massively expands** upon previous package and **provides necessary tools** for HE modelling

- WARDEN: **CRAN** package for **DES ready for HTA submissions**
  - Focus on **transparency, flexibility, and being easy to use**
  - New version with improvements coming to CRAN in ~ 2 weeks.

- **Future development: resource constraints**, efficiency improvements, other features (as needed)

- Your **feedback** would be welcome!

QR code to WARDEN website

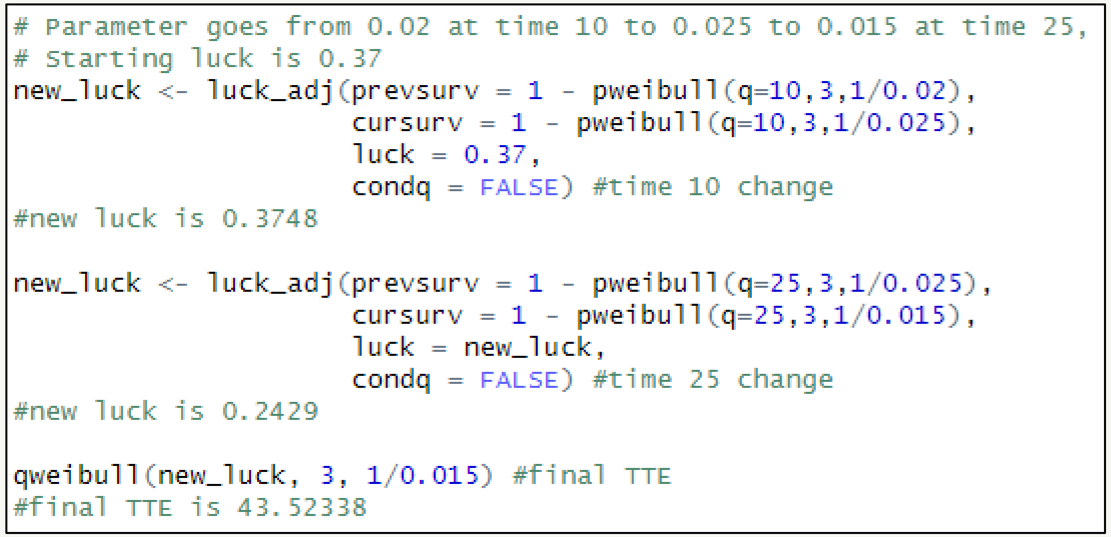
Q&A

QR code to WARDEN website

Appendix

Appendix - Showcase
Luck Adjustment

- Patient draws survival from **Weibull(3 ,50)** with **“luck”** (random draw number) **0.37** → Time to event = 38.65 (**qweibull(0.37, 3, 50)**)
  - At **time 10**, patient has an event and switches to **Weibull(3,4)** (worsens TTE).
  - At **time 25**, patient has another event and switches to **Weibull(3, 66.67)** (improves TTE).
  - What would be the final time to event of this patient?
  - We cannot draw directly from the last Weibull (TTE = 51.54), as the patient spent most of the time on another parametrization. Solution: adjust the patients’ “luck”.
- Use of **luck_adj()** function facilitates this computation



```
# Parameter goes from 0.02 at time 10 to 0.025 to 0.015 at time 25,
# Starting luck is 0.37
new_luck <- luck_adj(prevsurv = 1 - pweibull(q=10,3,1/0.02),
                     cursurv = 1 - pweibull(q=10,3,1/0.025),
                     luck = 0.37,
                     condq = FALSE) #time 10 change
#new luck is 0.3748

new_luck <- luck_adj(prevsurv = 1 - pweibull(q=25,3,1/0.025),
                     cursurv = 1 - pweibull(q=25,3,1/0.015),
                     luck = new_luck,
                     condq = FALSE) #time 25 change
#new luck is 0.2429

qweibull(new_luck, 3, 1/0.015) #final TTE
#final TTE is 43.52338
```

QR code to WARDEN website