Overview table of functions

DARTH

19-2-2019

Table

00_general_functions.R

This file contains functions that are not specific to the case example. This means they can be useful in any decision model. Separating these general functions in a file makes it easy to transfer these functions to new project where you probably need these functions again.
- `owsa_det()`
* argument: `parms`, `ranges`, `nsamps`, `params.basecase`, `FUN`, `outcome`, `strategies`
* returns: `df.owsa.lng`

This function runs a deterministic one-way sensitivity analysis (OWSA) on a given function that produces outcomes. The arguments of the function are well described in the function file. The function returns a dataframe with the results of the sensitivity analysis. These results can be visualized with functions `plot.owsa`, and `owsa_tornado`
- `twsa_det()`
* argument: `parm1`, `parm2`, `ranges`, `nsamps`, `params.basecase`, `FUN`, `outcome`, `strategies`
* returns: `df.twsa.lng`

This function runs a deterministic two-way sensitivity analysis (TWSA) on a given function that produces outcomes. The argumenst of the function are well described in the function file. The function returns a dataframe with the results of the sensitivity analysis. These results can be visualized with `plot.twsa`, and `twsa_tornado`

01_model-inputs_function.R

- `f.define_init_params()`
* argument: none
* returns: `df.params.init`

This function creates a dataframe with the base-case parameters of the model, called `df.params.init`. The function does not require any arguments. External data from a csv file is loaded into the environment and the parameters that need to be calibrated get an placeholder value to make it possible to run the model. These values will be replaced after the calibrated values are known. The dataframe `df.params.init` including all base-case parameters values is returned.

02_simulation-model_function.R

- `f.decision_model()`
* argument: `v.params`
* returns: `a.P` and `m.M`

02__simulation-model__function.R

The input for this function is the `df.params` dataframe including all base-case parameter values generated by the `f.define_init_params()`. Based on this dataframe the age-specific transition probabilities are calculated for all cycles. The second step is the initiation of the age-specific transition probability matrices in an array, called `a.P`. The functions includes two checks to make sure that each transition probability is a valid value as well as the transition probability matrix as a whole. The third part of the model initiate the cohort trace `m.M` that will be used to store the model results that are generated by iterating the STM over time.

03__calibration__function.R

- `f.calibration_out()`
* argument: `v.params.calib`
* returns: `l.out`

The function `f.calibration_out` the values of the uncalibrated parameters in the base-case dataframe are substituted by values of calibrated parameters in the vector `v.params.calib`. The function `f.decision_model(v.params = df.params)` is used again to run the model with the new values for the calibrated parameters. Based on the new model results the epidemiological measures,
overall survival, disease prevalence and the proportion of sick in the sick states is estimated. At three timepoints, `t = 11`, `t = 21` and `t = 31`, the values of all three the epidemiological measured are stored in the list `l.out`. This list is the output of the function.

- `f.log_lik()`
* argument: `v.params`
* returns: `v.llik.overall`

DESCRIPTION

- `likelihood()`
* argument: `v.params`
* returns:

DESCRIPTION

- `sample.prior()`
* argument: `n.samp`
* returns: `m.param.samp`

DESCRIPTION

- `f.log_prior()`
* argument: `v.params`
* returns:

DESCRIPTION

- `prior()`
* argument: `v.params`
* returns:

DESCRIPTION

- `f.log_post()`
* argument: `v.params`
* returns: `n.lpost`

DESCRIPTION

04__validation__function.R

- `f.data_summary()`
* argument: `data`, `varname`, `groupnames`
* returns: `data_sum`

04__validation__function.R

This function is used to calculate the mean, standard deviation and 95% Credible Interval for each group based on the data. We use the data predicted by the model for the `data` argument in the function. The `varname` is the name of the column of the values of interest. With the `groupnames` argument you can specify the column names to be used as grouping variables. The function returns a dataframe with summary information about the data.

05a__deterministic-analysis__function.R

- `f.generate_basecase_params()`
* argument: NONE
* returns:`v.params.basecase`
This functions generate a base-case set of CEA Parameters based on the 03_imin-output.RData stored in the data folder and the `f.define_init_params()` function spcified in the 01__model-inputs_function.R file. With this information the function creates a vector with basecase values for the parameters.

- `f.calculate_ce_out()`
* argument:`v.params` and `n.wtp`
* returns: `m.ce`
This function is used to calculate the cost-effectiveness of the strategies. The vector `v.params` informs the model about the values of the parameters to run the simulation with. With the argument `n.wtp` the user can specify the willingness-to-pay threshold. This value is used to compute the net benefits. Several calculations are done within the function, like computing the cohort trace, summing the results per cycle, apply discounting and summarizing the results. These summarized results are combined in the matrix `m.ce`. This matrix gives the discounted costs, effectiveness and net monetary benefit (NMB).

- `twsa.plot.det()`
* argument: `params`, `outcomes`, `strategyNames`, `outcomeName = "Outcome"`, `mx = TRUE`
* returns: `twsa.gg`
Interpreting results of is often much easier via a graph, especially two-way sensitivity analysis. This functions creates twsa plots for you based on the generated data for the two parameters. This information will be given to the `params` argument in the function. Further, the function needs to know which values to take as the `outcomes`, the `strategyNames`, the names of the XXXXX
The final result of the function is a graph generated using `ggplot`, called `twsa.gg`.

- `TornadoPlot()`
* argument: `Params`, `Outcomes`, `titleName`, `outcomeName`, `ylab = "$"`
* returns: `tor.gg`
Interpreting results of a deterministic analysis are much easier via a graph. Tornado plots are used to show the results of these analyses. This function helps you plotting Tornado diagrams using the generated data. The function needs a vector with the parameter names, `Parm`, a matrix including the parameter specific outcomes, `Outcomes`, a title you like to give to the plot, `titleName` and the name of the outcome shown in the plot `outcomeName`. The ylab sign are dollars by default, but you can change this when you use this function for results using another currency. The final result of the function is a graph generated using `ggplot`, called `tor.gg`.

05b__uncertainty-analysis__function.R

- `f.generate_psa_params()`
* argument: `seed` and `n.sim`
* returns: `df.psa.params`

05b__uncertainty-analysis__function.R

In a probabilistic sensitivity analysis (PSA) the model is evaluate using multiple random combination of parameter values sampled from their distributions. In order to run a model with multiple combinations of parameter values we need to create a psa data set. The function `f.generate_psa_params()` is able to do this for use. In the arguments of the function we have the option to set the seed, otherwise we use the default value, which allows us to generate the same results for this stochastic process. The other argument we need to specify is the number of simulations, `n.sim`, we would like to run our PSA. For each simulation the function creates a combination of parameter values based on their distributions. These values are combined in the dataframe `df.psa.params`.

06__value-of-information__function.R

- `function`
argument:
returns:
DESCRIPTION