**Overview:**

PA1 is designed to illustrate Probabilistic Analysis (PA). PA1 has a very simple structure which mimics the broad features of more complex population health microsimulation models such as OncoSim, HPVMM, GMM, POHEM. The same principles also apply to other microsimulation models. The model has 6 input parmeters, 3 derived parameters, and a single aggregate output measure (average life expectancy). It consists of about 100 lines of code which are reproduced below. The model illustrates the major kinds of input parameters:

* constant for purposes of the analysis
* subject to PA (sampled internally or externally)
* derived from other parameters
* specifying a health intervention

PA1 produces a single measure of health system performance (life expectancy), for base and intervention scenarios. An intervention scenario can be compared to a base scenario, using matched replicate simulation pairs to quantify the effect of parameter uncertainty and Monte Carlo Error. A typical PA1 scenario uses 100 replicates of a population of 100,000, simulating a total of 10,000,000 individuals. On a desktop, a PA1 scenario takes about 1 minute to complete, with additional diagnostic output tables. Larger numbers of replicates or individuals can be simulated with linear scaling in performance in OpenM++.

**Substance:**

In the absence of intervention, the population has an average age-independent mortality hazard Lambda. The mortality hazard of an individual depends additionally on the presence or absence of an underlying condition which is drawn at birth. The birth prevalence of the condition is given by the parameter Alpha, which can vary between 0.0 and 1.0. Individuals with the condition experience a higher-than-average relative risk of mortality RR\_with. Individuals without the condition have a lower relative risk RR\_without, which is typically slightly less than 1.0. RR\_without is derived by a formula which respects the average mortality hazard Lambda, and the implied average life expectancy. Thus Alpha and RR\_with can be varied without affecting the life expectancy of the population.

A health system intervention can identify and treat a proportion Treated\_proportion of those with the condition. A treated individual has a relative risk RR\_treated which is between RR\_with and RR\_without, depending on the parameter Treated\_effectiveness, which varies between 0.0 (treated is like untreated) and 1.0 (treated is like those without the condition).

**Probabilistic Analysis (sampled internally)**

The parameter RR\_with is assumed to have been estimated out-of-model by equation estimation using microdata (e.g. Cox regression). The estimation distribution of RR\_with is expressed analytically, specifically log(RR\_with) is Normal with mean log\_RR\_mu and standard deviation log\_RR\_sigma. In a real application these two parameters would be taken from the output of the regression procedure, but in PA1 they are set arbitrarily (see file PA.dat reproduced below for example values). When PA1 is run, a value from the estimation distribution is drawn for each simulation replicate (aka member). The simulation framework arranges that the same value for RR\_with is drawn for the same replicate in both base and variant scenarios. That produces a set of matched replicate pairs where correlated uncertainty can largely cancel out in base-variant comparisons. Internal sampling has the advantage that samples of arbitrary size can be drawn.

**Probabilistic Analysis (sampled externally)**

The parameter Alpha (the proportion at birth having the condition), is assumed to come out-of-model, but not from a regression procedure giving parameters of an estimation distribution. For example, Alpha might have been computed from a survey with 100 sets of bootstrap weights, which would produce a sample of 100 values of Alpha. That sample of 100 values would reflect the uncertainty in Alpha due to sample size and survey design. In PA1, a sample of 100 values of Alpha is provided through an external csv file. For details on how parmaeters can be supplied to ompp models, please see the section “How model finds parameters” in the ompp wiki page <https://ompp.sourceforge.io/wiki/index.php/Model_Run_Cycle#Sub-values:_sub-samples.2C_members.2C_replicas>.

**Applicability to more complex models:**

While PA1 is very simple, the techniques illustrated can be applied to more complex models, e.g.

* Parameters estimated using multivariate regression can be draw internally in the model (like RR\_with in PA1) by converting the variance-covariance matrix from the regression into a multivariate Normal distribution using Cholesky decomposition. This preserves the correlation structure of estimated parameters.
* Parameters estimated using multiple independent regressions can be drawn internally in the model by drawing from those independent estimation distributions.
* Multiple parameters estimated externally from a single dataset using bootstrap weights can be provided to the model as separate external csv files. In effect the sample is a sample of parameter vectors with correlated error, where each vector uses the same set of bootstrap weights.
* Multiple parameters estimated externally using calibration or whole-model estimation can be supplied as a sample of vectors from the (empirical) estimation distribution. Each parameter has its own csv file.

**Contents of module PA.mpp:**

// Copyright (c) 2013-2018 OpenM++ Contributors (see AUTHORS.txt for details)

// This code is licensed under the MIT license (see LICENSE.txt for details)

//LABEL(PA, EN) Substantive module

/\* NOTE(PA, EN)

Substantive module for PA1

\*/

#include "omc/omSimulation.h" // For IDE

#if 0 // Hide from IDE

parameters

{

//EN Mortality hazard

double Lambda;

//EN Relative risk of mortality with condition

model\_generated double RR\_with;

//EN Relative risk of mortality without condition

model\_generated double RR\_without;

//EN Relative risk of mortality with condition adn treated

model\_generated double RR\_treated;

//EN log(RR) estimation distribution (mu)

double log\_RR\_mu;

//EN log(RR) estimation distribution (sigma)

double log\_RR\_sigma;

//EN Proportion with condition at birth

double Alpha;

//EN Proportion with condition identified at birth and treated

double Treated\_proportion;

//EN Effectiveness of treatment

double Treated\_effectiveness;

};

actor Person //EN Individual

{

//EN alive

bool alive = { true };

//EN condition

bool condition = { false };

//EN treated

bool treated = { false };

void Start\_condition(); //EN Set condition at birth

hook Start\_condition, Start;

event timeMortalityEvent, MortalityEvent; //EN Mortality event

};

table Person A\_LifeExpectancy //EN Life expectancy

[trigger\_entrances(alive, false)]

{

{

value\_out(age) / unit //EN Age at death decimals=4

} //EN Measure

};

#endif // Hide from IDE

void PreSimulation()

{

// Draw value of RR with condition for current simulation member

// from estimation distribution.

double z = RandNormal(1);

RR\_with = exp(log\_RR\_mu + log\_RR\_sigma \* z);

// Compute RR without condition to maintain same average life expectancy (with no treatment)

RR\_without = RR\_with \* (1.0 - Alpha) / (RR\_with - Alpha);

// Compute RR with condition and treated.

// Is between RR\_without and RR\_with, a function of effectiveness.

RR\_treated = RR\_without \* Treated\_effectiveness + RR\_with \* (1.0 - Treated\_effectiveness);

};

void Person::Start\_condition()

{

// Draw condition at birth

condition = (RandUniform(2) < Alpha);

// If person has condition, draw if treated

if (condition) {

treated = (RandUniform(3) < Treated\_proportion);

}

// Allow other functions to hook to this function

hook\_Start\_condition();

}

TIME Person::timeMortalityEvent()

{

// Apply appropriate RR to baseline hazard

// depending on presence of condition and treatment.

double hazard = 0.0;

if (condition) {

if (treated) {

hazard = RR\_treated \* Lambda;

}

else {

hazard = RR\_with \* Lambda;

}

}

else {

hazard = RR\_without \* Lambda;

}

// Draw time to event, with constant hazard (exponential distribution).

TIME event\_time = WAIT(-TIME(log(RandUniform(4)) / hazard));

return event\_time;

}

void Person::MortalityEvent()

{

alive = false;

Finish();

}

**Example contents of module PA.dat (parameter input file):**

parameters {

/\*NOTE(Lambda, EN)

Gives average life expectancy of 50 years.

\*/

double Lambda = 0.0200; //EN Mortality hazard

/\*NOTE(log\_RR\_mu, EN)

Value is ln(2) which gives mean RR of 2.0.

\*/

double log\_RR\_mu = 0.69314718055994530941723212145818; //EN log(RR) estimation distribution (mu)

double log\_RR\_sigma = 0.100000; //EN log(RR) estimation distribution (sigma)

double Alpha = 0.20; //EN Proportion at birth with condition

double Treated\_proportion = 0.20;

double Treated\_effectiveness = 0.50;

};