vivarium$_{u}nimelb_{t}obacco_{i}ntervention_{c}omparisonD$

Release 1.0.0

Rob Moss

Jan 21, 2020

CONTENTS:

Research repository for the Vivarium MSLT Tobacco Intervention Comparison project.

CHAPTER

ONE

INSTALLATION

To set up a new research environment, open up a terminal and run:

```
$> conda create --name=mslt_tobacco python=3.6 hdf5
...standard conda install stuff...
$> conda activate mslt_tobacco
(mslt_tobacco) $> git clone git@github.com:ihmeuw/vivarium_unimelb_tobacco_
↪intervention_comparison.git
(mslt_tobacco) $> cd vivarium_unimelb_tobacco_intervention_comparison
(mslt_tobacco) $> pip install -e .
```

See the *Getting started* section of the tutorial for further details.

1.1 Concepts

Multi-state life tables (MSLT) are a tool that can be used to predict the impact of preventative interventions on chronic disease morbidity and mortality, by interventions acting through changes in risk factors that affect multiple disease incidence rates (hence "multi-state" life tables). Metrics such as health-adjusted life years (HALYs) and health-adjusted life expectancy (HALE) can be used to quantify intervention impacts.

To demonstrate how a MSLT works, we begin by showing a life table can be used to estimate HALYs and HALE before any intervention is applied, and then show to simulate simple intervention effects.

Table 1: A simple life table example, which shows how morbidity and mortality data are used to calculate life expectancy and life year statistics. Input data are shown in **bold text**, everything else is calculated within the life table.

Year	Age	Sex	Population	Mortality rate	Probability of death	Number of deaths	Number of survivors	Person years lived	Life expectancy	YLD rate	HALYs	HALE
2011	**52**	**male**	**129,850**	**0.0030**	0.0030	390	129,460	129,655	33.12	**0.1122**	115,103	36.00
2012	53	male	129,460	**0.0032**	0.0032	413	129,047	129,254	32.23	**0.1122**	114,747	35.18
…	…	…	…	…	…	…	…	…	…	…	…	…
2067	108	male	221	**0.4811**	0.3819	84	136	179	1.62	**0.3578**	115	1.04
2068	109	male	136	**0.4811**	0.3819	52	84	110	1.31	**0.3578**	71	0.84
2069	110	male	84	**0.4812**	0.3820	32	52	68	0.81	**0.3578**	44	0.52

The above table shows a life table for the population cohort who were 52 years old at the start of the year 2011. The inputs for this life table (shown in bold, above) are:

1. The cohort age after the first time-step (52), sex (male), and initial population size (129,850);

2. The age-specific, sex-specific mortality rate; and
3. The age-specific, sex-specific years lost due to disability (YLD) rate.

For each future year, the following calculations are performed:

1. The (age-specific) mortality rate is converted into a mortality risk (i.e., the probability that an individual will die in that year);
2. The risk is multiplied by the population size to calculate the number of deaths that occur in that year, which also determines the number of survivors;
3. The person-years lived are calculated under the assumption that the deaths occur at a constant rate, and so this the mean of the starting population and the surviving population;
4. The life expectancy is defined as the sum of all future life years, divided by the starting population size; and
5. The years lost due to disability (YLD) rate is used to discount the person-years lived and the life expectancy, which yields the health-adjusted life years (HALYs) and health-adjusted life expectancy (HALE) for this cohort.

The above life table simulated the lifespan of the 52 year old male cohort. Within Vivarium, the same calculations are performed in parallel for multiple cohorts. In the simulations presented here we divide the population into five-year age-group cohorts for each sex, under the assumption that, e.g., males aged 50-54 can be reasonably approximated as a single cohort aged 52 years.

The above examples is also called the "business as usual" (BAU) scenario, and uses reference values for the mortality and YLD rates. A simple intervention that lowers mortality rates by, say, 5% would generate more LYs and HALYs, and longer LEs and HALEs, than those obtained in the BAU scenario. These difference between the BAU and intervention life tables comprise the intervention effect. However, in the MSLT model the intervention effect is typically not modelled directly as a change in the all-cause mortality and morbidity rates. Rather, we construct multiple disease-specific life tables and allow interventions to affect disease incidence rates. Changes to disease incidence will result in changes to disease-specific mortality and morbidity rates. The sum of these differences across all diseases is then subtracted from the all-cause mortality and morbidity rates in the intervention life table. We now address each of these concepts in turn.

1.1.1 Chronic disease

To capture intervention effects, we set up multiple parallel diseases as separate lifetables. We consider chronic diseases as being independent (i.e., the prevalence of one disease does not affect the incidence or case fatality rate of another). The reason for setting up the parallel disease states is that we simulate intervention effects (through risk factor changes) as changes in disease incidence rates. We thus need "BAU" and "intervention" lifetables for all diseases impacted by the intervention.

The outputs of the chronic disease life tables are:

- A disease-specific mortality rate, for each cohort at each year; and
- A disease-specific YLD rate, for each cohort at each year.

These outputs are generated for both the BAU and intervention scenarios, with the difference between BAU and intervention (across all of the disease life tables) then being subtracted from the BAU all-cause mortality and morbidity rates, to create the "intervention" life table. We can then measure the intervention effect in terms of the differences in LYs, HALYs, LE, and HALE, between the BAU and intervention life tables.

A chronic disease is characterised in terms of:

- Incidence rate (i);
- Remission rate (r);
- Case fatality rate (f);

- Initial prevalence ($C(0)$); and
- Disability rate.

The equations for chronic disease prevalence, remission, and mortality come from Barendregt et al., 2003. A key assumption in their derivation is the independence of mortality from all causes:

> "If it is assumed that mortality from all other causes is independent of the disease, i.e., that it is the same for healthy and diseased people, this implies that the transition hazards for incidence, remission and case fatality are not affected by the value of the 'all other causes' mortality. Therefore we can set the value of mortality from all other causes to 0 (i.e., leave it out of the equations) and still derive the right values for the disease rates."

With this simplifying assumption, the system of equations are:

$$\frac{dS_a}{da} = -i_a S_a + r_a C_a \tag{1.1}$$

$$\frac{dC_a}{da} = -(f_a + r_a)C_a + i_a S_a \tag{1.2}$$

$$\frac{dD_a}{da} = f_a C_a \tag{1.3}$$

Table 2: Definition of symbols used in the chronic disease equations.

Symbol	Definition
i_a	Disease incidence rate for people of age a.
r_r	Disease remission rate for people of age a.
f_a	Case fatality rate for people of age a.
S_a	Number of healthy people at age a.
C_a	Number of diseased people at age a.
D_a	Number of dead people at age a (due to the disease).

This is a system of linear ordinary differential equations (ODEs), for which an analytic solution can be obtained (see equations (4)–(6) in Barendregt et al., 2003).

1.1.2 Acute disease and other events

The MSLT models presented here use a time-step size of 1 year. So it is not sensible to talk about the **prevalence** of acute diseases (such as lower respiratory tract infections) and acute events (such as road traffic accidents), which may affect the all-cause mortality and YLD rate, but whose duration is significantly less than that of a single time-step.

The acute events are therefore characterised in terms of two rates:

- Their excess mortality rate; and
- Their YLD rate.

An intervention could affect either or both of these rates.

For example, regular use of face masks could reduce the transmission of respiratory infections, which would reduce the mortality and YLD rates for respiratory infections.

Conversely, encouraging people to undertake short trips on foot could increase the rate of pedestrian-vehicle collisions, which would increase the mortality and YLD rates for road traffic accidents.

1.1.3 Risk factors

We will use tobacco smoking as an example of a risk factor that:

a) Increases the incidence risk for a number of diseases; and

b) Can be mitigated by interventions that reduce smoking prevalence.

Tobacco smoking

Similar to chronic diseases, we can define the prevalence of tobacco smoking in terms of initial prevalence, incidence ("uptake") and remission ("cessation").

The simplest risk factor will have two categories of exposure:

- No exposure; and
- Exposure.

However, since cessation of tobacco smoking does not immediately reverse all effects of exposure, we will increase the number of exposure categories so that the exposure can gradually return to baseline over a period of 20 years. In other words, we assume that it takes 20 years after quitting to recover the health risks associated with having never smoked. The exposure categories will therefore be:

- No exposure (never smoked);
- Exposure (currently smoking);
- 0 years post-cessation;
- 1 year post-cessation;
- 2 years post-cessation;
- ...
- 19 years post-cessation;
- 20 years post-cessation; and
- 21+ years post-cessation.

Upon cessation, an individual will progress through the post-cessation exposure levels and, 21 years later, their exposure category will be **21+ years post-cessation** and they will have the same incidence risks as those individuals who have never smoked.

For each exposure level, we need the relative risk (or risk ratio) for each disease of interest. This is how the prevalence of the exposure will affect disease incidence, which in turn will affect the mortality and YLD rates in the MSLT.

1.1.4 Interventions

We will consider three different interventions that affect the prevalence of tobacco smoking.

Each of these interventions will affect the exposure distribution of the risk factor (tobacco smoking). This will be done by modifying any of the rates that affect the exposure (i.e.,, the uptake and remission rates), or by moving people from one exposure category to another.

Note: Another option, not explored here, is to modify the relative risk(s) associated with an exposure category (the "relative risk shift" method, Barendregt and Veerman, 2009). With this method, proportions of the cohort do not transition between exposure states. Rather, each exposure category has a shift in its average exposure which is

modelled as a shift in its relative risk. We use this method for BMI categories, but for smoking we explicitly model transitions between smoking states.

Tobacco eradication

For this intervention, we assume that tobacco is no longer available from some specific year Y. This will have two effects:

- From year Y, the uptake rate will be zero; and
- At year Y, all current smokers will cease to smoke and their exposure category will be changed to **0 years post-cessation**. They will then progress through the post-cessation exposure categories and, 20 years later, they will have the same disease incidence rates as the **never smoked** exposure category.

Tobacco-free generation

For this intervention, we assume that individuals born after a certain year Y will be unable to purchase tobacco and therefore will never smoke. This will have one effect (where we assume that all uptake occurs at age 20):

- From year $Y + 20$, the uptake rate will be zero.

Tobacco tax

This is a more complex intervention, where we assume that there will be a gradual tax increase that affects the price of cigarette packs, and that tobacco uptake and cessation will be affected by the annual cost increase.

While the underlying details are more complex than the other interventions outlined above, the effects of this intervention on tobacco smoking prevalence are themselves simple:

- The uptake rate will be reduced by some proportion; and
- The cessation rate will be increased by some proportion.

The reduction in uptake will grow larger over time, since the tobacco price will increase over time. However, the impact on cessation rates is only felt in the year of tax increase (Blakely et al., 2015).

Note: The size of these effects is determined by price elasticities, which can vary by sex and age (and other strata of heterogeneity, as required).

1.1.5 Input data requirements

The data required for an MSLT model depend on the model components. Here, we define the data requirements for each type of component.

In general, rates and values are stored in tables with the following columns:

Note: For convenience, all of these input data can be collected into a single data artifact. For each of the tables described below, we identify the name under which it should be stored in a data artifact.

We will see how to use data artifacts in the *MSLT tutorials*.

Core MSLT

The cohorts and their population sizes are defined in the `population.structure` table:

year	age	sex	population	bau_population
2011	2	female	108970.000	108970.000
2011	2	male	114970.000	114970.000
2011	7	female	105600.000	105600.000
2011	7	male	110470.000	110470.000
…	…	…	…	…
2011	102	female	1035.000	1035.000
2011	102	male	433.125	433.125
2011	107	female	207.000	207.000
2011	107	male	86.625	86.625

The age-specific, sex-specific mortality rates are defined in the `cause.all_causes.mortality` table:

year_start	year_end	age_start	age_end	sex	rate
2011	2012	0	1	female	0.003586
2011	2012	0	1	male	0.004390
2011	2012	1	2	female	0.000330
2011	2012	1	2	male	0.000340
…	…	…	…		
2120	2121	109	110	female	0.524922
2120	2121	109	110	male	0.529281

Note: Rates and other values that apply to specific cohorts during the simulation (i.e., all input data except for the initial cohort population sizes and initial disease/risk factor prevalence) are indexed by time intervals and age intervals.

In the mortality rate table shown above, the rate in each row applies:

- From the time in **year_start** up to (but not including) the time in **year_end**; and
- To cohorts whose age lies between **age_start** (inclusive) and **age_end** (exclusive).

Similarly, the age-specific, sex-specific disability rates are defined in the `cause.all_causes.disability_rate` table:

year_start	year_end	age_start	age_end	sex	rate
2011	2012	0	1	female	0.014837
2011	2012	0	1	male	0.020674
2011	2012	1	2	female	0.022379
2011	2012	1	2	male	0.026409
…	…	…	…		
2120	2121	109	110	female	0.366114
2120	2121	109	110	male	0.357842

Chronic diseases

For each chronic disease, the initial prevalence and disease-specific rates are stored in the following tables (where the disease name is `NAME`).

The incidence rate (i) is stored in `chronic_disease.NAME.incidence`:

year_start	year_end	age_start	age_end	sex	NAME_i
2011	2012	0	1	female	0.0
…	…	…	…		

The disability rate (DR) is stored in `chronic_disease.NAME.morbidity`:

year_start	year_end	age_start	age_end	sex	NAME_DR
2011	2012	0	1	female	0.0
…	…	…	…		

The mortality rate (f) is stored in `chronic_disease.NAME.mortality`:

year_start	year_end	age_start	age_end	sex	NAME_f
2011	2012	0	1	female	0.0
…	…	…	…		

The initial prevalence is stored in `chronic_disease.NAME.prevalence`:

year	age	sex	NAME_prev
2011	0	female	0.0
…	…	…	…

The remission rate (r) is stored in `chronic_disease.NAME.remission`:

year_start	year_end	age_start	age_end	sex	NAME_r
2011	2012	0	1	female	0.0
…	…	…	…		

Note: Note that the column names are different in each table.

Acute diseases and other events

For each acute disease/event, the morbidity and mortality rates are stored in the following tables (where the disease/event names is `NAME`).

The morbidity rate is stored in `acute_disease.NAME.morbidity`:

year_start	year_end	age_start	age_end	sex	NAME_disability_rate
2011	2012	0	1	female	0.000301
…	…	…	…		

The mortality rate is stored in `acute_disease.NAME.mortality`:

year_start	year_end	age_start	age_end	sex	NAME_excess_mortality
2011	2012	0	1	female	0.000032
…	…	…	…		

Note: Note that the column names are different in each table.

Risk factors

The tobacco risk factor (as implemented by the `DelayedRisk` component) requires several data tables.

The incidence rate is stored in `risk_factor.tobacco.incidence`:

year_start	year_end	age_start	age_end	sex	incidence
2011	2012	0	1	female	0.000301
…	…	…	…		

The remission rate is stored in `risk_factor.tobacco.remission`:

year_start	year_end	age_start	age_end	sex	remission
2011	2012	0	1	female	0.000301
…	…	…	…		

The initial prevalence for each exposure category is stored in `risk_factor.tobacco.prevalence`:

year	age	sex	tobacco.no	tobacco.yes	tobacco.0	tobacco.1	…	tobacco.20	tobacco.21
2011	0	female	1.0	0.0	0.0	0.0	…	0.0	0.0
…	…	…	…	…	…	…	…	…	…

The relative risk of mortality for each exposure category (defined separately for the BAU and intervention scenarios) is stored in `risk_factor.tobacco.mortality_relative_risk`:

year_start	year_end	age_start	age_end	sex	tobacco.no	tobacco.yes	…	tobacco.21	tobacco_intervention.no	tobacco_intervention.yes	…	tobacco_intervention.21
2011	2012	0	1	female	1.0	1.0	…	1.0	1.0	1.0	…	1.0
…	…	…	…	…	…	…	…	…	…	…		

The relative risk of chronic disease incidence for each exposure category is stored in `risk_factor.tobacco.disease_relative_risk`, which contains separate columns for each chronic disease. Shown here is an example for two chronic diseases, called `DiseaseA` and `DiseaseB`:

year_start	year_end	age_start	age_end	sex	DiseaseA_no	DiseaseA_yes	…	DiseaseA_21	DiseaseB_no	DiseaseB_yes	…	DiseaseB_21
2011	2012	0	1	female	1.0	1.0	…	1.0	1.0	1.0	…	1.0
…	…	…	…	…	…	…	…	…	…	…		

Interventions

The `TobaccoEradication` and `TobaccoFreeGeneration` interventions don't have any data requirements. The tobacco tax intervention, however, is characterised in terms of its effect on the incidence (i.e., uptake) and remission (i.e., cessation) rates.

The incidence effect is stored in `risk_factor.tobacco.tax_effect_incidence`:

year_start	year_end	age_start	age_end	sex	incidence_effect
2011	2012	0	1	female	1.0
2011	2012	0	1	male	1.0
2011	2012	1	2	female	1.0
2011	2012	1	2	male	1.0
…	…	…	…		
2120	2121	108	109	female	0.866004
2120	2121	108	109	male	0.866004
2120	2121	109	110	female	0.866004
2120	2121	109	110	male	0.866004

The remission effect is stored in `risk_factor.tobacco.tax_effect_remission`:

year_start	year_end	age_start	age_end	sex	remission_effect
2011	2012	0	1	female	1.0
2011	2012	0	1	male	1.0
2011	2012	1	2	female	1.0
2011	2012	1	2	male	1.0
…	…	…	…		
2031	2032	22	23	female	0.975724
2031	2032	22	23	male	0.975724
2031	2032	23	24	female	0.975724
2031	2032	23	24	male	0.975724
…	…	…	…		
2120	2121	108	109	female	1.0
2120	2121	108	109	male	1.0
2120	2121	109	110	female	1.0
2120	2121	109	110	male	1.0

1.1.6 Recording life table outputs

The multi-state life table contains a vast amount of information for each population cohort at each time-step of a model simulation. Since the primary objective of MSLT models is to predict the impact of preventative interventions on population morbidity and mortality, only some of these data are relevant and worth recording.

The core concepts are:

1. MSLT components, such as diseases, risk factors, and interventions, will record quantities of interest as columns in the population table;
2. Observers will record the values of these columns (and also those of columns that identify each cohort, such as their age and sex) at each time-step; and
3. At the end of the simulation, observers will concatenate the values observed at each time-step into a single table, calculate summary statistics (if required), and save the resulting table to disk.

The MSLT framework provides a number of "observers" that record tailored summary statistics during a model simulation. We now introduce each of the provided observers in turn.

Note: Typically, each observer will record summary statistics for the "business-as-usual" (BAU) scenario **and** for the intervention scenario.

Population morbidity and mortality

The `MorbidityMortality` observer records the core life table quantities (as shown in the *example table*) at each year of the simulation. This includes calculating quantities such as the life expectancy and health-adjusted life expectancy (HALE) for each cohort at each time-step.

Chronic disease incidence, prevalence, and mortality

The `Disease` observer records the chronic disease incidence and prevalence, and the number of deaths caused by this disease, at each year of the simulation. For example, with an intervention that *reduces the incidence of chronic heart disease (CHD) by 5%* for all cohorts at all time-steps, it will produce the following output:

disease	year	age	sex	BAU incidence	Incidence	BAU prevalence	Prevalence	BAU deaths	Deaths	Change in incidence	Change in prevalence
…	…	…	…	…	…	…	…	…	…	…	…
CHD	2011	53	male	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
CHD	2012	54	male	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
…	…	…	…	…	…	…	…	…	…	…	…
CHD	2066	108	male	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0019732946424867795	3.643709743081369e-05
CHD	2067	109	male	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0019732946424867795	0.0005170621228446082
…	…	…	…	…	…	…	…	…	…	…	…

Risk factor prevalence

The `TobaccoPrevalence` observer records the smoking status of each cohort at each time-step. Note that all of the post-cessation exposure categories are summed together.

year	age	sex	BAU never smoked	BAU currently smoking	BAU previously smoked	BAU population	Never smoked	Currently smoking	Previously smoked	Population
…	…	…	…	…	…	…	…	…	…	…
2011	53	male	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
2012	54	male	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
…	…	…	…	…	…	…	…	…	…	…
2066	108	male	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
2067	109	male	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
…	…	…	…	…	…	…	…	…	…	…

1.1.7 Uncertainty analyses

In order to account for uncertainties in the input data, assumptions about the business-as-usual scenario, the effects of interventions, etc, we can run many model simulations and vary the input data. In each simulation we randomly draw values for each input parameter from some probability distribution (e.g., a normal distribution, where we set the mean to the input value used in the BAU, and define the standard deviation). Accordingly, each simulation will generate different intervention effects. We then define the 95% uncertainty interval for each output (LYs, HALYs, LE, HALE) as the 2.5% and 97.5% percentiles of the values obtained over all of these simulations.

The basic process is:

1. Identify rate(s) and/or value(s) for which uncertainties exist;
2. Define a probability distribution to characterise the uncertainty for each rate/value.
3. Identify whether the samples drawn from each distribution should be independent, or correlated in some way. For example, you may wish to correlate the samples for each rate across all cohorts (e.g., by age, sex, and ethnicity).
4. Draw N samples for each of the rate(s) and/or value(s).
5. Store these samples according to the same table structure as per the *original data*, with each sample represented as a separate row, and with one additional column (`"draw"`) that identifies the draw number ($1 \ldots N$).

This will result in a single, larger data artifact that contains all of the draws. In a model specification, you can then identify both the data artifact **and** the draw number, and when the simulation is run it will automatically select the correct values from all data tables that contain multiple draw.

See *Uncertainty analyses* for an example of running such an analysis.

1.2 Tutorials

In these tutorials, you will learn how to reproduce each of the simulations presented in the paper "Multistate lifetable modelling of preventive interventions: Concept and Python code".

After completing these tutorials, you will be able to adapt and modify these simulations, to explore the impact of different model assumptions and interventions, and to capture different simulation outputs of interest.

Important: These tutorials are intended to be followed in order, from first to last.

Note: The input data requirements for each of the MSLT components introduced in these tutorials are *described here*.

1.2.1 Getting started

1. You need to have Python 3.6 installed. If you don't already have this version of Python installed, the easiest option is to use Anaconda. Once Anaconda is installed:
    1. Create a new virtual environment:

    ```
    conda create --name=mslt_tobacco python=3.6
    ```

    2. Activate this Conda environment:

    ```
    conda activate mslt_tobacco
    ```

2. Download the Vivarium MSLT Tobacco Intervention Comparison project.
    - You can clone this project using `git`; this will create a new directory called **vivarium_unimelb_tobacco_intervention_comparison**.

    ```
    git clone https://github.com/population-interventions/vivarium_unimelb_
    ↪tobacco_intervention_comparison.git
    ```

    - Alternatively, you download the project as a zip archive and unzip its contents; this will create a new directory called **vivarium_unimelb_tobacco_intervention_comparison-master**.
3. Open a terminal and install the project using `pip`.
    - If you used `git` to clone the repository:

    ```
    cd vivarium_unimelb_tobacco_intervention_comparison
    pip install -e
    ```

    - If you downloaded the zip archive:

    ```
    cd vivarium_unimelb_tobacco_intervention_comparison-master
    pip install -e
    ```

4. Create the data artifacts, which will be stored in the `artifacts` directory:

```
make_artifacts minimal
```

5. Create the model specification files, which will be stored in the `model_specifications` directory:

```
make_model_specifications
```

Once you have completed these steps, you will be able to run all of the simulations described in these tutorials. For each simulation there will be a model specification file, whose file name ends in `.yaml`. These are plain text files, that you can edit in any text editor. To run the simulation described in one of these files, run the following command in a command prompt or terminal, from within the project directory:

```
simulate run model_specifications/model_file.yaml
```

Note: Each simulation will produce one or more output CSV files. You can then extract relevant subsets from these data files and plot them using your normal plotting tools. This allows you to easily examine outcomes of interest for specific cohorts and/or over specific time intervals.

The figures shown in these tutorials were created using external tools, not included in the Vivarium Public Health package and not documented here. Any plotting software could be used to produce similar figures.

1.2.2 The business-as-usual (BAU) scenario

The business-as-usual (BAU) scenario characterises what we expect to occur in the absence of any intervention. It comprises a population that are subject to BAU morbidity and mortality rates, and is the baseline against which we quantitatively evaluate the impact of different interventions.

Defining a simulation

Model simulations are defined by text files that describe all of the simulation components and configuration settings. These details are written in the YAML markup language, and the file names typically have a `.yaml` extension.

In the intervention example presented below, we provide a step-by-step description of the contents of these YAML files.

In brief, these files will contain three sections:

- The `plugins` section, where we load the plugins that allow us to make use of data artefacts;
- The `components` section, where we list the simulation components that define the population demographics, the BAU scenario, and the intervention; and
- The `configuration` section, where we identify the relevant data artefact, and define component-specific configuration settings and other simulation details.

Listing 1: An example simulation definition.

```
plugins:
    optional:
        data:
            controller: "vivarium_public_health.dataset_manager.ArtifactManager"
            builder_interface: "vivarium_public_health.dataset_manager.
↪ArtifactManagerInterface"

components:
    vivarium_public_health:
        mslt:
            population:
                - BasePopulation()
```

(continues on next page)

(continued from previous page)

```
                - Mortality()
                - Disability()
            intervention:
                - ModifyAllCauseMortality('reduce_acmr')
            observer:
                - MorbidityMortality()

configuration:
    input_data:
        # Change this to "mslt_tobacco_maori_20-years.hdf" for the Maori
        # population.
        artifact_path: artifacts/mslt_tobacco_non-maori_20-years.hdf
        input_draw_number: 0
    population:
        # The population size here is the number of cohorts.
        # There are 22 age bins (0-4, 5-9, ..., 105-109) for females and for
        # males, making a total of 44 cohorts.
        population_size: 44
    time:
        start:
            year: 2011
        end:
            year: 2120
        step_size: 365  # In days
    intervention:
        reduce_acmr:
            # Reduce the all-cause mortality rate by 5%.
            scale: 0.95
    observer:
        output_prefix: results/mslt_reduce_acmr
```

Data artefacts

Data artefacts collect all of the required *input data tables* into a single file. The input data files that were used to generate the data artefacts for this tutorial are stored in the `src/vivarium_unimelb_tobacco_intervention_comparison/external_data/` directory. If you modify any of the input data files, you can rebuild these artefacts by running the provided script:

```
make_artifacts minimal
```

This will update the follow data artefacts:

- `mslt_tobacco_maori_20-years.hdf`: data for the Maori population, where cessation of smoking results in gradual recovery over the next 20 years.
- `mslt_tobacco_non-maori_20-years.hdf`: data for the non-Maori population, where cessation of smoking results in gradual recovery over the next 20 years.
- `mslt_tobacco_maori_0-years.hdf`: data for the Maori population, where cessation of smoking results in immediate recovery.
- `mslt_tobacco_non-maori_0-years.hdf`: data for the non-Maori population, where cessation of smoking results in immediate recovery.

Intervention: a reduction in mortality rate

In this section, we describe how to use the MSLT components to define a model simulation that will evaluate the impact of reducing the all-cause mortality rate. We then show how to run this simulation and interpret the results.

Note: All of the MSLT components are contained within the `vivarium_public_health.mslt` module. This module is divided into several sub-modules; we will use the `population`, `intervention`, and `observer` modules in this example.

Defining the model simulation

Because we are reading all of the necessary input data tables from a preexisting data artifact, we need to load two Vivarium plugins:

Listing 2: Load the necessary Vivarium plugins.

```
plugins:
    optional:
        data:
            controller: "vivarium_public_health.dataset_manager.ArtifactManager"
            builder_interface: "vivarium_public_health.dataset_manager.
↪ArtifactManagerInterface"
```

We then need to specify the location of the data artifact in the configuration settings:

Listing 3: Define the input data artifact.

```
configuration:
    input_data:
        # Change this to "mslt_tobacco_maori_20-years.hdf" for the Maori
        # population.
        artifact_path: artifacts/mslt_tobacco_non-maori_20-years.hdf
```

The core components of the simulation are the population demographics (`BasePopulation`), the mortality rate (`Mortality`), and the years lost due to disability (YLD) rate (`Disability`). These components are located in the `population` module, and so we identify them as follows:

Listing 4: The core population components.

```
components:
    vivarium_public_health:
        mslt:
            population:
                - BasePopulation()
                - Mortality()
                - Disability()
```

We define the number of population cohorts, and the simulation time period, in the configuration settings:

Listing 5: Define the number of cohorts and the simulation time period.

```
configuration:
    population:
        # The population size here is the number of cohorts.
```

(continues on next page)

(continued from previous page)

```
        # There are 22 age bins (0-4, 5-9, ..., 105-109) for females and for
        # males, making a total of 44 cohorts.
        population_size: 44
    time:
        start:
            year: 2011
```

We also add a component that will reduce the all-cause mortality rate (`ModifyAllCauseMortality`, which is located in the `intervention` module) and give this intervention a name (`reduce_acmr`). We define the reduction in all-cause mortality rate in the configuration settings, identifying the intervention by name (`reduce_acmr`) and defining the mortality rate scaling factor (`scale`):

Listing 6: The core population components.

```
components:
    vivarium_public_health:
        mslt:
            intervention:
                - ModifyAllCauseMortality('reduce_acmr')

configuration:
    intervention:
        reduce_acmr:
            # Reduce the all-cause mortality rate by 5%.
            scale: 0.95
```

Finally, we need to record the core life table quantities (as shown in the *example table*) at each year of the simulation, by using the `MorbidityMortality` observer (located in the `observer` module) and specifying the prefix for output files (`mslt_reduce_acmr`):

Listing 7: The core population components.

```
components:
    vivarium_public_health:
        mslt:
            observer:
                - MorbidityMortality()

configuration:
    observer:
        output_prefix: results/mslt_reduce_acmr
```

Putting all of these pieces together, we obtain the following simulation definition:

Listing 8: The simulation definition for the BAU scenario and the intervention.

```
plugins:
    optional:
        data:
            controller: "vivarium_public_health.dataset_manager.ArtifactManager"
            builder_interface: "vivarium_public_health.dataset_manager.
↪ArtifactManagerInterface"

components:
    vivarium_public_health:
```

(continues on next page)

(continued from previous page)

```
        mslt:
            population:
                - BasePopulation()
                - Mortality()
                - Disability()
            intervention:
                - ModifyAllCauseMortality('reduce_acmr')
            observer:
                - MorbidityMortality()

configuration:
    input_data:
        # Change this to "mslt_tobacco_maori_20-years.hdf" for the Maori
        # population.
        artifact_path: artifacts/mslt_tobacco_non-maori_20-years.hdf
        input_draw_number: 0
    population:
        # The population size here is the number of cohorts.
        # There are 22 age bins (0-4, 5-9, ..., 105-109) for females and for
        # males, making a total of 44 cohorts.
        population_size: 44
    time:
        start:
            year: 2011
        end:
            year: 2120
        step_size: 365  # In days
    intervention:
        reduce_acmr:
            # Reduce the all-cause mortality rate by 5%.
            scale: 0.95
    observer:
        output_prefix: results/mslt_reduce_acmr
```

Running the model simulation

The above simulation is already defined in `mslt_reduce_acmr.yaml`. Run this simulation with the following command:

```
simulate run model_specifications/mslt_reduce_acmr.yaml
```

When this has completed, the output recorded by the `MorbidityMortality` observer will be saved in the file `mslt_reduce_acmr_mm.csv`. The contents of this file will contain the following results:

Table 3: An extract of the simulation results, showing a subset of rows for the cohort of males aged 50-54 in 2010.

Year of birth	Sex	Age	Year	Survivors	BAU Survivors	Population	BAU Population	ACMR	BAU ACMR	Probability of death	BAU Probability of death	Deaths	BAU Deaths	YLD rate	BAU YLD rate	Person years	BAU Person years	HALYs	BAU HALYs	LE	BAU LE	HALE	BAU HALE
...																							
1959	male	52	2011	129,4	129,4	139,4	139,4	0.00	0.00	0.00	0.00	370.1	389.6	0.11	0.11	129,6	129,6	115,2	115,2	33.1	33.1	26.3	26.0
1959	male	53	2012	129,0	129,0	139,0	139,0	0.00	0.00	0.00	0.00	392.9	413.5	0.11	0.11	129,2	129,2	114,7	114,7	32.2	32.2	25.5	25.2
1959	male	54	2013	128,6	128,6	138,5	138,6	0.00	0.00	0.00	0.00	417.7	439.5	0.11	0.11	128,8	128,8	114,3	114,3	31.3	31.3	24.7	24.4
...																							
1959	male	108	2067	192.3	136.4	303.7	220.7	0.457	0.481	0.136	0.138	91.1	84.3	0.357	0.357	48.0	78.6	59.2	14.7	1.7	1.6	1.1	1.0
1959	male	109	2068	121.7	84.3	192.3	136.4	0.457	0.481	0.136	0.138	79.5	52.1	0.357	0.357	57.0	110.4	100.8	70.9	1.3	1.3	0.9	0.8
1959	male	110	2069	77.1	52.1	121.7	84.3	0.457	0.481	0.236	0.238	44.7	32.2	0.357	0.357	99.4	68.2	63.8	43.8	0.8	0.8	0.5	0.5
...																							

We can examine the impact of this intervention on a single cohort (e.g., non-Maori males aged 50-54 in 2011) by filtering the rows by **Year of birth** and **Sex**. We can then plot columns of interest, such as the LE and HALE for both the BAU and intervention scenarios:

With some further data processing, we can also plot the survival of this cohort in both the BAU and intervention scenarios, relative to the starting population, and see how the survival rate has increased as a result of this intervention.

1.2.3 Chronic heart disease

Intervention: a reduction in CHD incidence

Note: In this example, we will also use components from the `vivarium_public_health.mslt.disease` module.

Compared to the *previous simulation*, we will now add a chronic disease component, and replace the all-cause mortality rate intervention with an intervention that affects CHD incidence.

To add CHD as a separate cause of morbidity and mortality, we use the `Disease` component:

Listing 9: Add a chronic disease.

```
components:
    vivarium_public_health:
        mslt:
            disease:
                - Disease('CHD')
```

We then replace the `ModifyAllCauseMortality` intervention with the `ModifyDiseaseIncidence` intervention. We give this intervention a name (`reduce_chd`) and identify the disease that it affects (`CHD`). In the configuration settings, we identify this intervention by name (`reduce_chd`) and specify the scaling factor for CHD incidence (`CHD_incidence_scale`).

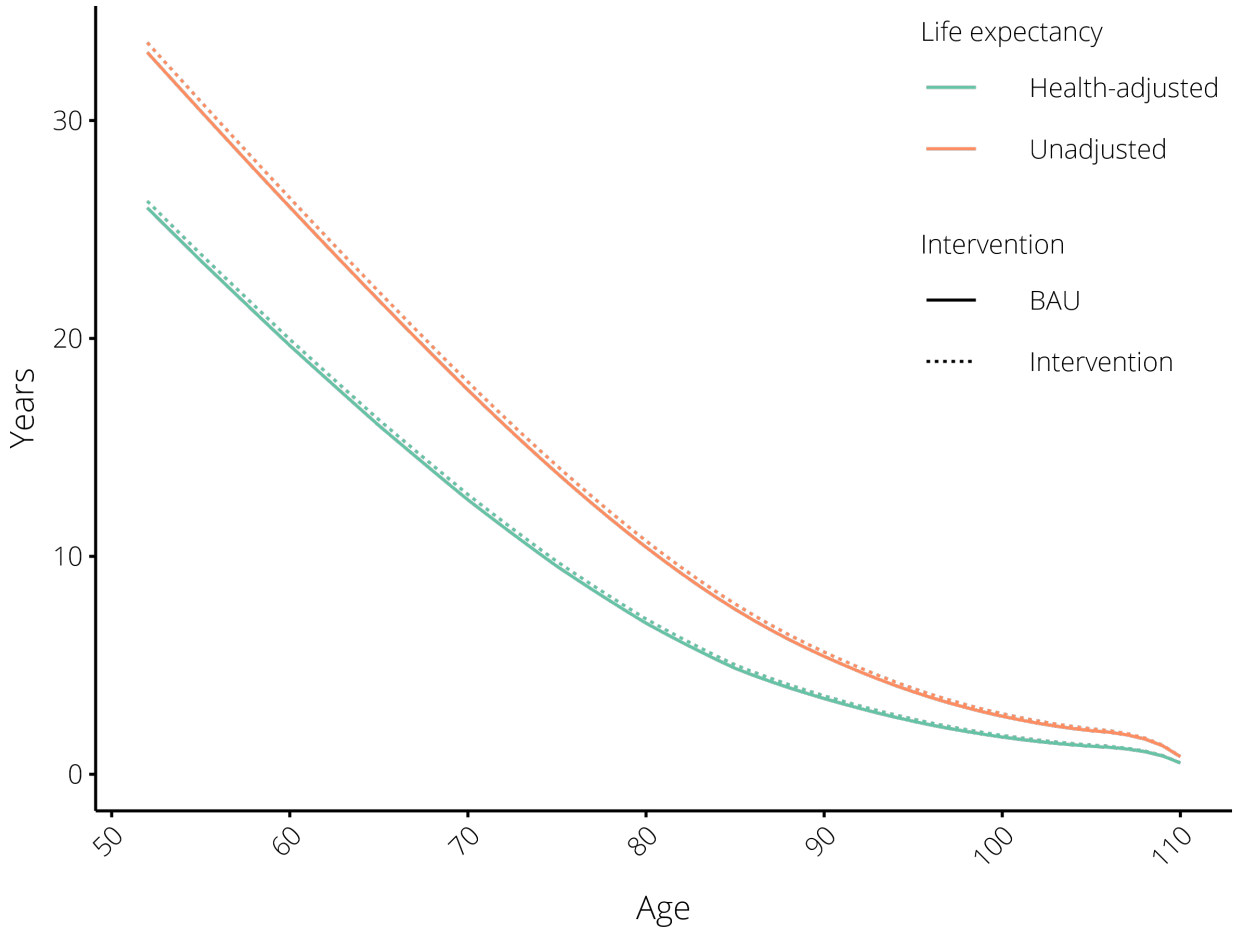

Fig. 1: The impact of reducing the all-cause mortality rate by 5% on life expectancy. Results are shown for the cohort of males aged 50-54 in 2011.

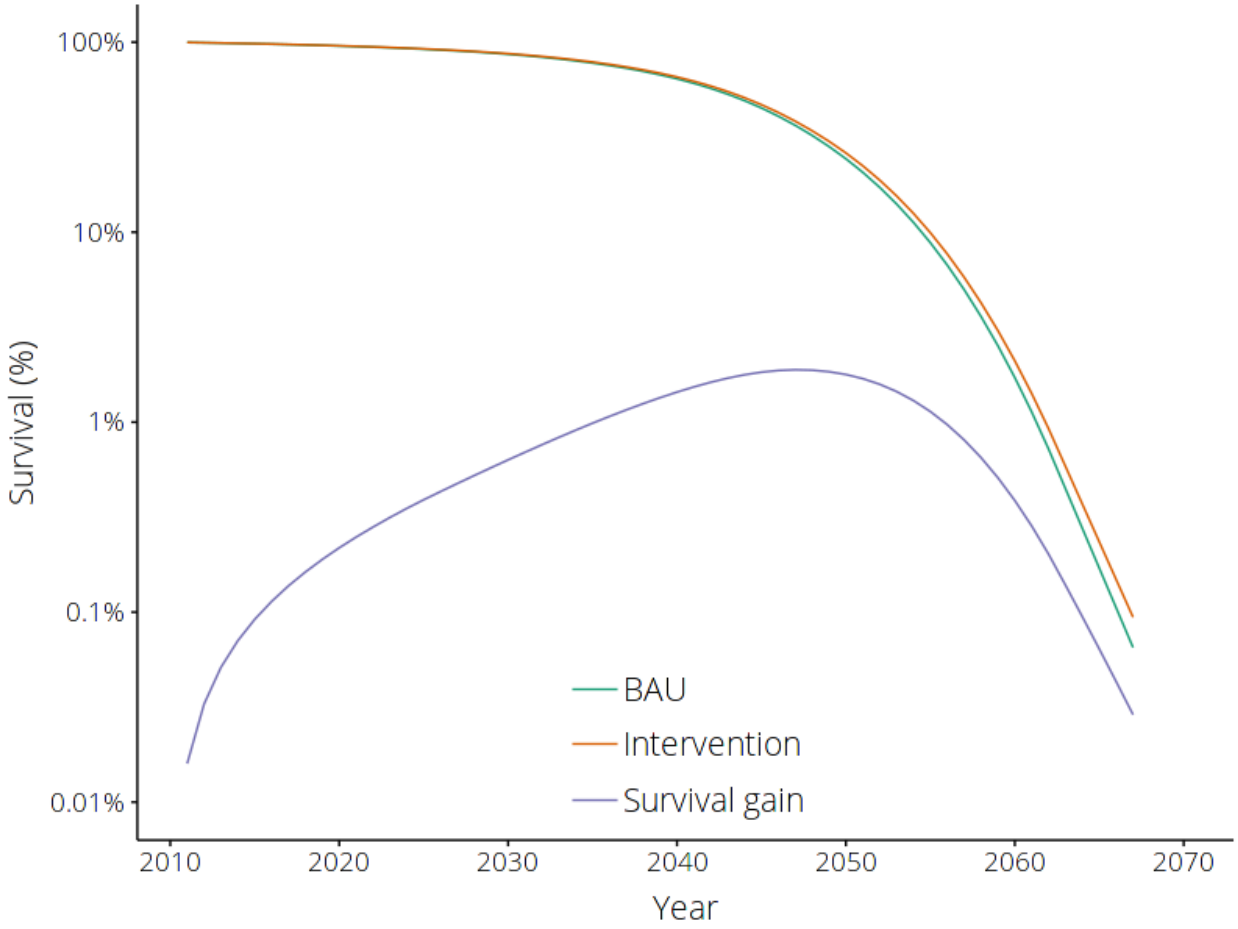

Fig. 2: The impact of reducing the all-cause mortality rate by 5% on survival rate. Results are shown for the cohort of males aged 50-54 in 2011.

Listing 10: Add an intervention that reduces CHD incidence.

```
components:
    vivarium_public_health:
        mslt:
            intervention:
                - ModifyDiseaseIncidence('reduce_chd', 'CHD')
configuration:
    intervention:
        reduce_chd:
            # Reduce the CHD incidence rate by 5%.
            CHD_incidence_scale: 0.95
```

Finally, we add an observer to record CHD incidence, prevalence, and deaths, in both the BAU scenario and the intervention scenario. We use the `Disease` observer, identify the disease of interest by name (`CHD`), and specify the prefix for output files (`mslt_reduce_chd`).

Listing 11: Record CHD incidence, prevalence, and deaths.

```
components:
    vivarium_public_health:
        mslt:
            observer:
                - Disease('CHD')

configuration:
    observer:
        output_prefix: results/mslt_reduce_chd
```

Putting all of these pieces together, we obtain the following simulation definition:

Listing 12: The simulation definition for the BAU scenario and the intervention.

```
plugins:
    optional:
        data:
            controller: "vivarium_public_health.dataset_manager.ArtifactManager"
            builder_interface: "vivarium_public_health.dataset_manager.
↪ArtifactManagerInterface"

components:
    vivarium_public_health:
        mslt:
            population:
                - BasePopulation()
                - Mortality()
                - Disability()
            disease:
                - Disease('CHD')
            intervention:
                - ModifyDiseaseIncidence('reduce_chd', 'CHD')
            observer:
                - MorbidityMortality()
                - Disease('CHD')

configuration:
```

(continues on next page)

(continued from previous page)

```
    input_data:
        # Change this to "mslt_tobacco_maori_20-years.hdf" for the Maori
        # population.
        artifact_path: artifacts/mslt_tobacco_non-maori_20-years.hdf
        input_draw_number: 0
    population:
        # The population size here is the number of cohorts.
        # There are 22 age bins (0-4, 5-9, ..., 105-109) for females and for
        # males, making a total of 44 cohorts.
        population_size: 44
    time:
        start:
            year: 2011
        end:
            year: 2120
        step_size: 365  # In days
    intervention:
        reduce_chd:
            # Reduce the CHD incidence rate by 5%.
            CHD_incidence_scale: 0.95
    observer:
        output_prefix: results/mslt_reduce_chd
```

Running the model simulation

The above simulation is already defined in `mslt_reduce_chd.yaml`. Run this simulation with the following command:

```
simulate run model_specifications/mslt_reduce_chd.yaml
```

When this has completed, the output recorded by the `MorbidityMortality` observer will be saved in the file `mslt_reduce_chd_mm.csv`.

We can now plot the survival of this cohort in both the BAU and intervention scenarios, relative to the starting population, and see how the survival rate has increased as a result of this intervention.

The output recorded by the `Disease` observer will be saved in the file `reduce_chd_disease.csv`. The contents of this file will contain the following results:

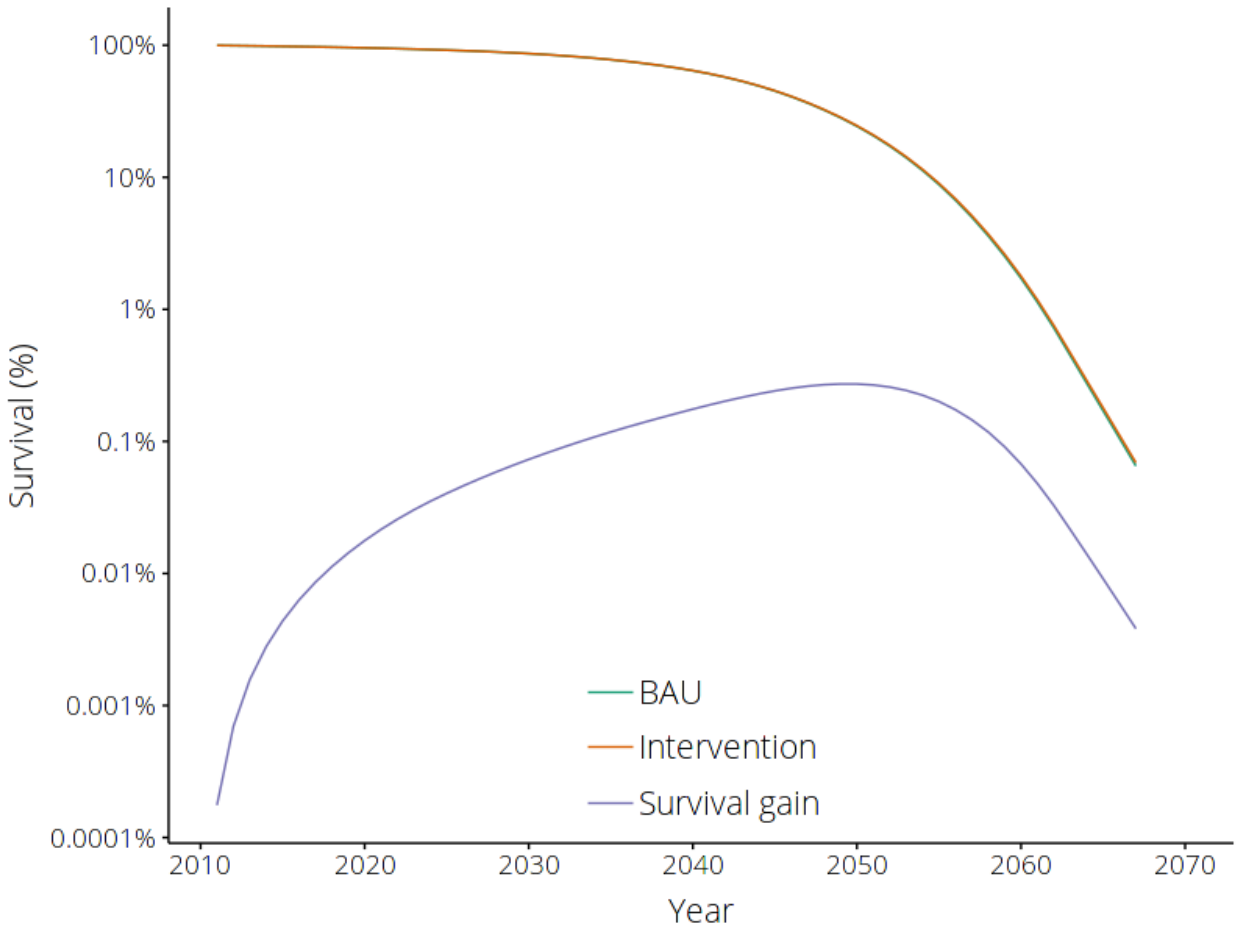

Fig. 3: The impact of reducing the CHD incidence rate by 5% on survival rate. Results are shown for the cohort of males aged 50-54 in 2010. Compare this to the impact of *reducing all-cause mortality rate by 5%*.

Table 4: An extract of the CHD statistics, showing a subset of rows for the cohort of males aged 50-54 in 2010.

Disease	Year of birth	Sex	Age	Year	BAU Incidence	Incidence	BAU Prevalence	Prevalence	BAU Deaths	Deaths	Change in incidence	Change in prevalence
...												
CHD	1959	male	52	2011	0.004984	0.004736	0.03664	0.03664	0.0	0.0	-0.000249	0.0
CHD	1959	male	53	2012	0.005339	0.005072	0.04121	0.040955	0.6	0.6	-0.000267	-0.000254
CHD	1959	male	54	2013	0.005698	0.005413	0.046049	0.04553	1.2	1.2	-0.000285	-0.000519
...												
CHD	1959	male	108	2067	0.038684	0.03675	0.185506	0.18575	692.3	674.8	-0.001934	0.000243
CHD	1959	male	109	2068	0.038684	0.03675	0.181912	0.182607	705.3	687.8	-0.001934	0.000695
CHD	1959	male	110	2069	0.038684	0.03675	0.178933	0.180058	717.5	700.1	-0.001934	0.001126
...												

1.2.4 Tobacco smoking: effect of interventions

Each chronic and acute disease that is affected by tobacco smoking is modelled as a separate component, so that interventions on tobacco smoking can affect the morbidity and mortality of these diseases. We also need to inform the tobacco component which diseases it should affect; this is done in the configuration section. The resulting simulation definition is quite long, simply because there are many diseases to include.

Listing 13: The simulation definition for the BAU scenario.

```
plugins:
    optional:
        data:
            controller: "vivarium_public_health.dataset_manager.ArtifactManager"
            builder_interface: "vivarium_public_health.dataset_manager.
↪ArtifactManagerInterface"

components:
    vivarium_public_health:
        mslt:
            population:
                - BasePopulation()
                - Mortality()
                - Disability()
            delay:
                - DelayedRisk('tobacco')
            disease:
                - Disease('CHD')
                - Disease('Stroke')
                - Disease('LungCancer')
                - Disease('HeadNeckCancer')
                - Disease('OesophagusCancer')
                - Disease('StomachCancer')
```

(continues on next page)

(continued from previous page)

```
                - Disease('LiverCancer')
                - Disease('ColorectalCancer')
                - Disease('PancreasCancer')
                - Disease('CervicalCancer')
                - Disease('BladderCancer')
                - Disease('KidneyCancer')
                - Disease('EndometrialCancer')
                - Disease('Melanoma')
                - Disease('ThyroidCancer')
                - Disease('COPD')
                - AcuteDisease('LRTI')
            observer:
                - MorbidityMortality()
                - TobaccoPrevalence()

configuration:
    input_data:
        # Change this to "mslt_tobacco_maori_20-years.hdf" for the Maori
        # population.
        artifact_path: artifacts/mslt_tobacco_non-maori_20-years.hdf
        input_draw_number: 0
    population:
        population_size: 44 # Male and female 5-year cohorts, aged 0 to 109.
    time:
        start:
            year: 2011
        end:
            year: 2120
        step_size: 365  # In days
    tobacco:
        delay: 20  # The delay (in years) between cessation and normal risks.
        affects:
            # This is where the affected diseases should be listed.
            CHD:
            COPD:
            BladderCancer:
            CervicalCancer:
            ColorectalCancer:
            EndometrialCancer:
            KidneyCancer:
            LiverCancer:
            LungCancer:
            OesophagusCancer:
            PancreasCancer:
            StomachCancer:
            ThyroidCancer:
            LRTI:
            Melanoma:
            Stroke:
    observer:
        output_prefix: mslt_tobacco_bau # The prefix for output files.
```

Tobacco eradication

We add the `TobaccoEradication` component, and specify at what year it comes into effect. Shown below are the new lines that are added to the simulation definition *for the BAU scenario*.

```
components:
    vivarium_public_health:
        mslt:
            # Other components ...
            intervention:
                TobaccoEradication()

configuration:
    # Other configuration settings ...
    tobacco_eradication:
        year: 2011
```

These simulations are already defined in the following files:

- `mslt_tobacco_maori_20-years_decreasing_erad.yaml`
- `mslt_tobacco_non-maori_20-years_decreasing_erad.yaml`

Tobacco-free generation

We add the `TobaccoFreeGeneration` component, and specify at what year it comes into effect. Shown below are the new lines that are added to the simulation definition *for the BAU scenario*.

```
components:
    vivarium_public_health:
        mslt:
            # Other components ...
            intervention:
                TobaccoFreeGeneration()

configuration:
    # Other configuration settings ...
    tobacco_free_generation:
        year: 2011
```

These simulations are already defined in the following files:

- `mslt_tobacco_maori_20-years_decreasing_tfg.yaml`
- `mslt_tobacco_non-maori_20-years_decreasing_tfg.yaml`

Tobacco tax

We enable the `tobacco_tax` option of the tobacco risk factor (`DelayedRisk`). Shown below are the new lines that are added to the simulation definition *for the BAU scenario*.

```
configuration:
    # Other configuration settings ...
    tobacco:
        tobacco_tax: True
```

These simulations are already defined in the following files:

- `mslt_tobacco_maori_20-years_decreasing_tax.yaml`
- `mslt_tobacco_non-maori_20-years_decreasing_tax.yaml`

Intervention comparison

If you run all of these simulations, you can then compare them by the gains that they provide in LYs and HALYs, and the reductions that they provide in ACMR and YLDR, using the data analysis software of your choice.

As an example, here are some of the results obtained for non-Maori males aged 50-54 in 2011, for the tobacco eradication intervention:

Table 5: Results for the tobacco eradication intervention, which yields gains in LYs, HALYs, ACMR, and YLDR.

Year of birth	Sex	Age	Year	Survivors	BAU Survivors	Population	BAU Population	ACMR	BAU ACMR	Probability of death	BAU Probability of death	Deaths	BAU Deaths	YLD rate	BAU YLD rate	Person years	BAU Person years	HALYs	BAU HALYs	LE	BAU LE	HALE	BAU HALE
…																							
1959	male	52	2011	129,461	129,461	129,850	129,850	0.003	0.003	0.003	0.003	389.6	389.6	0.1	0.1	129,655	129,655	103,434	103,434	33.1	26.3	26.0	
1959	male	53	2012	129,071	129,071	129,460	129,460	0.003	0.003	0.003	0.003	389.2	389.2	0.1	0.1	129,265	129,265	103,045	103,045	32.2	25.5	25.2	
1959	male	54	2013	128,681	128,681	129,071	129,071	0.003	0.003	0.003	0.003	388.5	389.5	0.1	0.1	128,876	128,876	102,655	102,655	31.3	24.7	24.4	
…																							
1959	male	108	2067	149.5	136.4	241.4	220.7	0.479	0.448	0.138	0.138	92.0	84.3	0.355	0.355	195.5	178.6	126.0	114.7	1.6	1.6	1.0	1.0
1959	male	109	2068	92.5	84.3	149.5	136.4	0.479	0.548	0.138	0.138	56.9	52.1	0.355	0.355	121.0	110.4	78.0	70.9	1.3	1.3	0.8	0.8
1959	male	110	2069	57.3	52.1	92.5	84.3	0.479	0.748	0.238	0.138	35.3	32.2	0.355	0.355	74.9	68.2	48.3	43.8	0.8	0.8	0.5	0.5
…																							

1.2.5 Conclusion

Now that you have completed these tutorials, you can extract subsets of the simulation outputs and plot the effect of interventions on individual cohorts over specific time periods. You can also change parameters in the model specification files — such as the time at which the tobacco eradication or the tobacco-free generation intervention comes into effect — and re-run these simulations to see how your changes affect the intervention impact.

For further explorations, such as providing your own input data tables and implementing custom risk factors and interventions, see the *Advanced Topics*.

1.3 Advanced topics

Here we describe how the multi-state life tables (MSLT) components are implemented.

Note that when you run a simulation using the simulate command:

```
simulate run reduce_acmr.yaml
```

The following sequence of operations will be performed:

1. The model specification will be read (in this case, from the file `reduce_acmr.yaml`) and a simulation object will be created.

2. The `simulation.setup()` method will call the `setup()` method for each of the MSLT components defined in the model specification.

   Note: This is where components will load data tables, register event handlers, etc.

3. The initial population is created (typically by the `BasePopulation` component).
4. The time-steps will be simulated, with each time-step triggering the following events in turn:
   1. `"time_step__prepare"`: The `DelayedRisk` component uses this event to account for transitions between exposure categories (i.e., uptake, cessation, and transitions between tunnel states). The `Disease` component uses this event to update disease prevalence and mortality for both the BAU and intervention scenarios, so that mortality and morbidity adjustments can be calculated. The `TobaccoEradication` component uses this event to move current smokers to the **0 years post-cessation** exposure category when tobacco is eradicated.
   2. `"time_step"`: The `BasePopulation` component uses this event to remove cohorts once they've reached the maximum age (110 years). The `Mortality` component uses this event to calculate the number of deaths and survivors at each time-step. The `Disability` component uses this event to calculate the HALYs for each cohort for both the BAU and intervention scenarios.
   3. `"time_step__cleanup"`: no MSLT components respond to this event.
   4. `"collect_metrics"`: the observer components will record relevant population details at the end of each time-step.
5. The simulation will trigger the `"simulation_end"` event and finish. The observer components use this event to write output tables to disk.

1.3.1 Uncertainty analyses

Uncertainty analyses are a **bespoke process**, because you need to decide which input data should be correlated (e.g., the incidence rate for a single disease, across all age groups, sex, and ethnicity). To build data artifacts that contain 2000 draws for each input rate/value, run the following command:

```
make_artifacts uncertainty
```

Note: This can take a long time to complete, and generates data artifacts that are around **3 GB** in size.

We have also provided a command that runs multiple simulations for a single model specification file, where each simulation uses a different draw from the data artifact. This script can be used as follows:

```
run_uncertainty_analysis --draws 2000 --spawn 16 modelA.yaml modelB.yaml [...]
```

This will run 2000 simulations for each of the model specifications (modelA.yaml, modelB.yaml, etc) and will simultaneously run 16 simulations at a time. Each simulation will produce distinct output files (`modelA_mm_1.csv`, `modelA_mm_2.csv`, etc).

1.3.2 Alternative BAU: Immediate recovery upon cessation

We now consider the case where cessation of smoking results in **immediate** recovery, rather than taking 20 years for the tobacco-associated relative risks to decrease back to 1.0. The purpose here is to highlight how our assumptions about the BAU scenario can affect the predicted impact of an intervention.

The only changes that we need to make to the simulation definition are:

1. To use a different data artifact for these simulations, where the initial prevalence of tobacco use is only defined for 3 exposure levels: never smoked, current smoker, and former smoker; and
2. Set the recovery delay to 0 years.

Note: We could have used the same data artifact as in previous simulations, but then the tobacco component would have to manipulate the input data into the appropriate form. We instead choose to perform all input data manipulation *before* generating the data artifacts.

```
configuration:
    input_data:
        # Change this to "mslt_tobacco_maori_data_0-years.hdf" for the Maori
        # population.
        artifact_path: artifacts/mslt_tobacco_non-maori_0-years.hdf
    # Other configuration settings ...
    tobacco:
        delay: 0
```

These simulations are already defined in the following files:

- Tobacco eradication:
  - `mslt_tobacco_maori_0-years_decreasing_erad.yaml`
  - `mslt_tobacco_non-maori_0-years_decreasing_erad.yaml`
- Tobacco tax:
  - `mslt_tobacco_maori_0-years_decreasing_tax.yaml`
  - `mslt_tobacco_non-maori_0-years_decreasing_tax.yaml`
- Tobacco-free generation:
  - `mslt_tobacco_maori_0-years_decreasing_tfg.yaml`
  - `mslt_tobacco_non-maori_0-years_decreasing_tfg.yaml`

Intervention comparison

If you run all of these simulations, you can then compare their effects (and how these differ to those obtained with the original BAU scenario), using the data analysis software of your choice.

As an example, here are some of the results obtained for non-Maori males aged 50-54 in 2011, for the tobacco eradication intervention:

Table 6: Results for the tobacco eradication intervention, which yields gains in LYs, HALYs, ACMR, and YLDR.

Year of birth	Sex	Age	Year	Sur-vivors	BAU Sur-vivors	Pop-u-la-tion	BAU Pop-u-la-tion	ACMR	BAU ACMR	Prob-a-bil-ity of death	BAU Prob-a-bil-ity of death	Deaths	BAU Deaths	YLD rate	BAU YLD rate	Per-son years	BAU Per-son years	HALYs	BAU HALYs	LE	BAU LE	HALE	BAU HALE
…																							
1959	male	52	2011	129,4	129,4	129,4	129,4	0.003	0.003	0.003	0.003	389.6	389.6	0.11	0.11	129,6	129,6	115,2	115,2	33.1	33.1	26.5	26.0
1959	male	53	2012	129,0	129,0	129,0	129,0	0.003	0.003	0.003	0.003	406.3	413.5	0.11	0.11	129,2	129,2	114,7	114,6	32.2	32.2	25.7	25.2
1959	male	54	2013	128,6	128,6	128,6	129,0	0.003	0.003	0.003	0.003	422.4	439.5	0.11	0.11	128,8	128,8	114,2	114,3	31.3	31.3	24.9	24.4
…																							
1959	male	108	2067	151.4	136.4	244.5	220.7	0.479	0.448	0.138	0.238	93.1	84.3	0.355	0.355	135.7	98.0	78.6	57.7	1.6	1.6	1.0	1.0
1959	male	109	2068	93.7	84.3	151.4	136.4	0.479	0.448	0.138	0.238	57.7	52.1	0.355	0.355	83.5	72.6	46.4	40.7	1.3	1.3	0.8	0.8
1959	male	110	2069	58.0	52.1	93.7	84.3	0.479	0.448	0.238	0.382	35.7	32.2	0.355	0.355	51.7	45.9	29.8	26.9	0.8	0.8	0.5	0.5
…																							

Note that these results differ to those obtained with the *original BAU scenario*.

1.3.3 Alternative BAU: Constant tobacco prevalence

We now consider the case where the prevalence of tobacco use in each cohort remains constant over time — in other words, the cessation rate is zero. As per the previous tutorial, the purpose here is to highlight how our assumptions about the BAU scenario can affect the predicted impact of an intervention.

The only change that we need to make to the simulation definition is:

- Set the remission rate to zero.

This is done by adding the following configuration option:

```
configuration:
    tobacco:
        constant_prevalence: True
```

These simulations are already defined in the following files:

- Tobacco eradication:
  - `mslt_tobacco_maori_20-years_constant_erad.yaml`
  - `mslt_tobacco_non-maori_20-years_constant_erad.yaml`
- Tobacco tax:
  - `mslt_tobacco_maori_20-years_constant_tax.yaml`
  - `mslt_tobacco_non-maori_20-years_constant_tax.yaml`
- Tobacco-free generation:
  - `mslt_tobacco_maori_20-years_constant_tfg.yaml`
  - `mslt_tobacco_non-maori_20-years_constant_tfg.yaml`

Intervention comparison

If you run all of these simulations, you can then compare their effects (and how these differ to those obtained with the original BAU scenario), using the data analysis software of your choice.

As an example, here are some of the results obtained for non-Maori males aged 50-54 in 2011, for the tobacco eradication intervention:

Table 7: Results for the tobacco eradication intervention, which yields gains in LYs, HALYs, ACMR, and YLDR.

Year of birth	Sex	Age	Year	Survivors	BAU Survivors	Population	BAU Population	ACMR	BAU ACMR	Probability of death	BAU Probability of death	Deaths	BAU Deaths	YLD rate	BAU YLD rate	Person years	BAU Person years	HALYs	BAU HALYs	LE	BAU LE	HALE	BAU HALE
...																							
1959	male	52	2011	129,469	129,469	129,850	129,850	0.003	0.003	0.003	0.003	389.6	389.6	0.112	0.112	129,655	129,655	115,403	115,403	33.1	33.1	26.6	26.0
1959	male	53	2012	129,079	129,079	129,460	129,460	0.003	0.003	0.003	0.003	413.2	413.5	0.112	0.112	129,252	129,253	114,750	114,746	32.2	32.2	25.7	25.2
1959	male	54	2013	128,666	128,666	129,047	129,047	0.003	0.003	0.003	0.003	438.5	439.5	0.112	0.112	128,828	128,828	114,337	114,336	31.3	31.3	24.9	24.4
...																							
1959	male	108	2067	169.8	136.4	273.6	220.7	0.476	0.748	0.137	0.238	190.3	384.3	0.352	0.352	82.1	178.6	143.7	114.7	1.6	1.6	1.1	1.0
1959	male	109	2068	105.4	84.3	169.8	136.4	0.477	0.748	0.137	0.238	194.4	52.1	0.352	0.352	87.6	110.4	89.1	70.9	1.3	1.3	0.9	0.8
1959	male	110	2069	65.4	52.1	105.4	84.3	0.477	0.748	0.137	0.238	240.0	32.2	0.352	0.352	55.4	68.2	55.3	43.8	0.8	0.8	0.5	0.5
...																							

Note that these results differ to those obtained with the *original BAU scenario*.

1.3.4 Writing a custom intervention

As *explained earlier*, an intervention will typically affect the exposure distribution of a risk factor by modifying one (or more) of:

- The rate(s) that affect the exposure (e.g., uptake of tobacco smoking);
- The prevalence of exposure categories (e.g., moving people from one exposure category to another); and
- The relative risk(s) associated with an exposure category.

Note: Interventions may also directly affect chronic and acute diseases, by modifying any of the rates associated with those diseases. This is very similar to modifying any of the rates that affect the exposure of a risk factor; the **only** difference is the choice of which rate(s) will be affected.

Structure of an intervention component

An intervention component will comprise the following methods:

- A constructor (`__init__`) that will **normally** accept two arguments:
  - The `self` parameter (a reference to the component *instance*); and
  - A `name` that will be used to identify this intervention, and which may be used in the configuration section of a simulation definition in order to define settings for this intervention.
- A `setup(self, builder)` method that will:
  - Load any required input value or rate tables.
  - Read any intervention-specific settings from `builder.configuration`, such as the year at which the intervention will come into effect.
  - Register value and/or rate modifiers (if required).
  - Register a time-step event handler to, e.g., move people from one exposure category to another (if required).
- Some number of value and/or rate modifiers (if required).
- The time-step event handler (if required).

Example of an intervention component

As an example, we will walk through each of these methods for the `TobaccoEradication` intervention. This intervention is quite simple, because it doesn't need to load any input data tables, and it has an all-or-nothing effect on a risk factor rate.

Recall that this intervention is controlled by a single configuration setting:

```
configuration:
    tobacco_eradication:
        year: 2011
```

The constructor

This intervention is currently hard-coded to modify the `'tobacco'` risk factor, which it stores in `self.exposure`.

The setup method

The `setup()` method performs a several necessary house-keeping tasks:

- It retrieves the year at which the intervention comes into effect (specified in the configuration section, as shown above) and stores it in `self.year`.
- It stores the simulation clock in `self.clock`, so that it can detect when this intervention comes into effect.
- It registers a modifier for the `tobacco_intervention.incidence` rate (i.e., the uptake rate in the intervention scenario).
- It registers a modifier for the `tobacco_intervention.remission` rate (i.e., the cessation rate in the intervention scenario).

The incidence modifier

The `adjust_inc_rate()` method, which was registered as a modifier for the `tobacco_intervention.incidence` rate, will set the rate to zero once the intervention is active. Recall that `self.year` is the year at which this intervention comes into effect.

Note: Once this intervention becomes active, this rate modifier applies an effect on every time-step.

The remission modifier

The `adjust_rem_rate()` method, which was registered as a modifier for the `tobacco_intervention.remission` rate, will set the rate to one once the intervention is active. This will have the effect of moving all of the people in the **currently smoking** exposure category to the **0 years post-cessation** exposure category. Recall that `self.year` is the year at which this intervention comes into effect.

Note: Once this intervention becomes active, this rate modifier applies an effect on every time-step.

1.3.5 Writing a custom observer

As *explained earlier*, an observer will typically record the values in a specific subset of columns at each time-step of a simulation, and save these data as a single table. There are three primary concerns when writing a custom observer:

- Deciding which columns to record;
- Recording the data in these columns at each time-step; and
- Collating these data and saving them to an output file.

Structure of an observer component

An observer component will comprise the following methods:

- A constructor (`__init__`).
- A `setup(self, builder)` method that will:
  - Identify which columns to record.
  - Register a time-step event handler to record values at each time-step.
  - Register an end-of-simulation event handler to write the recorded data to an output file.
- A time-step event handler (`on_collect_metric`).
- An end-of-simulation handler (`write_output`).

Example of an observer component

As an example, we will walk through each of these methods for the `MorbidityMortality` observer. This observer records the core life table quantities (as shown in the *example table*) at each year of the simulation.

The constructor

This component has one required argument for the constructor, which is the name of the file to which the data will be saved at the end of the simulation:

```
components:
    vivarium_public_health:
        mslt:
            observer:
                MorbidityMortality('output_file.csv')
```

So the `__init__` method takes two arguments, and stores the name of the output file in `self.output_file`.

The setup method

The `setup()` method performs a several necessary house-keeping tasks:

- It identifies the columns that it will observe.
- It then informs the framework that it will need access to these columns, and stores this "view" in `self.population_view`.
- It stores a reference to the simulation clock in `self.clock`, so that it can determine the current year at each time-step.
- It registers an event handler that will be called **after** each time-step (by selecting the "on_collect_metrics" event) that will record the current population state.
- It registers an event handler that will be called at the end of the simulation (by selecting the "simulation_end" event) that will write the recorded data to the output file.
- It creates an empty list, which will contain the data tables recorded at each time-step, and stores it in `self.tables`.
- It defines the column ordering for the output table, and stores it in `self.table_cols`.

The time-step event handler

The `on_collect_metrics()` method records the current values in the specified columns, which is achieved by:

- Retrieving those columns from the underlying population table, using the `get` method of `self.population_view`;
- Checking whether this table contains at least one population cohort;
- Adding a new column, `year`, to record the current year; and
- Adding this table to the list of recorded tables, `self.tables`.

The end-of-simulation event handler

The `write_output()` method saves the recorded data, by performing the following steps:

- Concatenating the tables recorded at each time-step into a single table;
- Calculating the year of birth for each cohort, so that individual cohorts can be identified by two columns: year of birth, and sex;
- Sorting the table rows so that they are grouped by cohort and arranged chronologically;
- Calculating the life expectancy and the health-adjusted life expectancy (HALE) for each cohort at each time-step; and
- Writing the sorted table to the specified output file.

Note: This is also the appropriate method in which to perform any post-processing of the data (e.g., calculating life expectancy and other summary statistics).

1.3.6 Writing a custom risk factor

As *explained earlier*, a risk factor will typically define a number of exposure categories, and each category will be assigned one or more relative risks (e.g., for chronic disease incidence). The primary concerns when writing a custom risk factor are:

- Identifying appropriate exposure categories;
- Identifying which rates will be modified by these exposure categories;
- Defining the relative risks for each rate, for each exposure category; and
- Defining transition rates between exposure categories (if applicable).

Once these concerns have been addressed, the input data requirements can be identified, and input data tables can be prepared.

Structure of a risk factor component

A risk factor component will comprise the following methods:

- A constructor (`__init__`) that will **normally** accept two arguments:
  - The `self` parameter (a reference to the component *instance*); and
  - A `name` that will be used to identify this risk factor, and which may be used in the configuration section of a simulation definition in order to define settings for this risk factor.
- A `setup(self, builder)` method that will:
  - Load any required input value or rate tables, including the initial prevalence for each exposure category.
  - Read any risk-factor-specific settings from `builder.configuration`.
  - Register value and/or rate modifiers.
  - Register a time-step event handler to, e.g., move people from one exposure category to another (if required).
- Some number of value and/or rate modifiers.
- The time-step event handler (if required).

- If you anticipate applying interventions that affect the prevalence of exposure categories, it may be convenient to also include a method that returns the column name for each exposure category.

Example of an intervention component

As an example, we will walk through each of these methods for the `DelayedRisk` risk factor, which was created to model the effects of tobacco smoking.

The constructor

This component has one required argument for the constructor, which is the name of the risk factor. Note that the constructor defines the default configuration settings for this component, because this depends on knowing the risk factor's name.

The setup method

The `setup()` method performs several necessary house-keeping tasks:

- It reads the configuration settings.
- It loads the initial prevalence for each exposure category.
- It loads the incidence and remission rates, and registers these rates so that they will be available at each time-step.
- It request access to the all-cause mortality rate, and loads the relative risk of mortality for each exposure category, so that it can determine the excess mortality produced by this risk factor.
- It registers rate modifiers for each disease that is affected by this risk factor (implemented by the `register_modifier` method, described below).
- It loads the disease-specific relative risks for each exposure category.
- It adds an initialization handler to create a column for each exposure category, and populate them with the initial prevalence.
- It loads the effects that a tobacco tax would have on the incidence and remissions rates.
- It registers an event handler that will be called **before** each time-step (by selecting the "time_step__prepare" event) that will move people from one post-cessation category to the next.
- It defines the columns that it will need to access, and stores this view in `self.population_view`.

The initialization method

The `on_initialize_simulants()` method creates a column for each of the exposure categories (with separate columns for the BAU and intervention scenarios), populates them with the initial prevalence values, and updates the underlying table.

The rate modifiers

This risk factor can affect an arbitrary number of diseases, and so this component includes the `register_modifier()` method, which registers modifiers for:

- The incidence rate of a chronic disease;
- The excess mortality rate of an acute disease/event; and
- The disability rate of an acute disease/event.

This approach was used because the component is currently unable to identify whether each disease that it affects is a chronic disease or an acute disease.

The `incidence_adjustment()` method calculates the mean relative risk in the BAU and intervention scenarios, from which it then calculates the PIF, and modifies the un-adjusted rate accordingly.

The prevalence modifier

The `on_time_step_prepare()` method modifies the prevalence, so that it takes effect **before** the time-step itself, and accounts for the normal transitions between exposure categories in both the BAU and intervention scenarios:

- The incidence rate moves people from the **never smoked** category to the **currently smoking** category;
- The remission rate moves people from the **currently smoking** category to the **0 years post-cessation** category; and
- People move from the **N years post-cessation** category to the **N+1 years post-cessation** category, until they reach **21+ years post-cessation**.
- It also accounts for mortality in each exposure category.

Note: The order in which these transitions are performed is important. First, we accumulate people in the final category, **21+ years post-cessation**. Second, we move people from the **N years post-cessation** category to the **N+1 years post-cessation** category in *reverse-chronological order*. Finally, we account for incidence and remission. This will account for the effects of a tobacco tax in the intervention scenario, if the `tobacco_tax` configuration setting was set to `True`, and the remission rate in the intervention scenario will be set to zero if the `constant_prevalence` configuration setting was set to `True`.

The column name method

For convenience, this component provides the `get_bin_names()` method that returns a list of the column names for each exposure category, for both the BAU and intervention scenarios.