Tutorial-3_BuildFunctions

June 13, 2024

1 Tutorial #3 - Build functions

This tutorial will introduce you to creating your own build functions so that you can modify the baseline country model.

1.1 What are build functions?

These are the functions that `EMODTask` calls to build the configuration files for each simulation in the experiement. They are called once for each simulation, so if you were to have a significant amount of processing in a build function, it will get multiplied by the number of simulations in your experiement. While you probably won't have extensive processing, it is something to be aware of.

Your build functions can have whatever name you wish, but users typically choose build_config, build_campaign, and build_demographics.

`build_config()` - This function will get called first to set the basic, simulation-wide configuration parameters. `build_campaign()` - This function is used for creating the logic of when, who, why, and how interventions are distributed. `build_demograpics()` - The demographics configuration determines the number of initial people in your scenario, fertility rates, mortality rates, and relationship parameters.

1.2 `build_config()`

In our example function, we are going to double the initial population of the baseline and reduce the duration of the simulation by 10 years. The increased population will make the simulations take longer, but the reduction in length should reduce that increase. (Simulations might take 5-7 minutes.)

[1]:
```
def build_config( config ):
    import emodpy_hiv.country_model as cm

    zambia = cm.Zambia()
    config = zambia.build_config( config )

    config.parameters.x_Base_Population = config.parameters.x_Base_Population *␣
↪2.0
    config.parameters.Simulation_Duration = config.parameters.
↪Simulation_Duration - (10 * 365)
```

```
    return config
```

1.3 build_campaign

In this example, we are going to add the distribution of a Long Lasting PrEP-like intervention. We will do a mass distribution every year for 15 years and increase the coverage each year.

One goal of this example is to demonstrate how we can use a `for-loop` in Python instead of copying and pasting a large amount of JSON.

[2]:
```
def build_campaign( ):
    import manifest
    import emodpy_hiv.country_model as cm
    import emodpy_hiv.interventions.cascade_helpers as helpers
    import emodpy_hiv.interventions.prep as prep

    zambia = cm.Zambia()
    campaign = zambia.build_campaign( manifest.schema_file )

    laprep = prep.new_intervention( campaign,
                                    efficacy_times=   [  0, 180, 210, 240, 270,␣
↪300, 330 ],
                                    efficacy_values=  [0.8, 0.8, 0.7, 0.5, 0.3,␣
↪0.1, 0.0 ],
                                    intervention_name="LA-PrEP",
                                    disqualifying_properties=None,
                                    new_property_value=None )

    # Target only people who have accessibility to healthcare AND who's Risk is␣
↪either HIGH or MEDIUM
    ip_restrictions = [
        { "Accessibility": "Yes", "Risk": "HIGH"  },
        { "Accessibility": "Yes", "Risk": "MEDIUM" }
    ]

    laprep_coverages = [ 0.1, 0.3, 0.5, 0.5, 0.5, 0.7, 0.7, 0.7, 0.8, 0.8, 0.8,␣
↪0.8, 0.9, 0.9, 0.9  ]
    start_year = 2025
    for coverage in laprep_coverages:
        start_day = (start_year - 1960.5) * 365
        helpers.add_scheduled_event( campaign,
                                     event_name=("Tutorial 3 LA-PrEP with␣
↪coverage="+str(coverage)),
                                     start_day=start_day,
                                     coverage=coverage,
                                     property_restrictions=ip_restrictions,
```

```
                                        out_iv=[laprep],
                                        node_ids=None )
        start_year = start_year + 1

    return campaign
```

1.4 build_demographics

In the demographics example, we are simply going to increase the accessibility to health care to 90%

[3]:
```
def build_demographics():
    import emodpy_hiv.country_model as cm

    zambia = cm.Zambia()
    demographics = zambia.build_demographics()

    demographics.AddIndividualPropertyAndHINT( Property="Accessibility",
                                               Values=[ "Yes", "No" ],
                                               InitialDistribution=[ 0.9, 0.1 ],
                                               overwrite_existing=True )

    return demographics
```

1.5 Code from Tutorial #2

The code below is from Tutorial #2, except that in `EMODTask` we will use the build functions we created above instead of the baseline country model's.

[4]:
```
# Will make the warnings off by default in 2.0
import emod_api.schema_to_class as s2c
s2c.show_warnings = False

from idmtools.core.platform_factory import Platform
from idmtools.entities.experiment import Experiment
from idmtools.builders import SimulationBuilder

import emod_hiv.bootstrap as dtk
import emodpy_hiv.emod_task as emod_task
import emodpy_hiv.country_model as cm
import manifest

def add_reports( task, mainifest ):
    import emodpy_hiv.reporters.builtin as rp

    rp.add_report_simulation_stats( task, manifest )
    rp.add_report_hiv_by_age_and_gender(task,
```

```
                                        start_year=1985, #avoid outbreak so␣
 ↪newly infected plot isn't overwhelmed
                                        end_year=2070,
                                        collect_gender_data=True,
                                        collect_age_bins_data=[15, 20, 25, 30,␣
 ↪35, 40, 45, 50],
                                        collect_circumcision_data=True,
                                        collect_hiv_stage_data=False,
                                        collect_ip_data=[],
                                        collect_intervention_data=[],
                                        add_transmitters=False,
                                        stratify_infected_by_cd4=False,
                                        event_counter_list=[],
                                        add_relationships=False,
                                        add_concordant_relationships=False)

    return


def process_results( experiment, platform ):
    import os, shutil
    from idmtools.analysis.analyze_manager import AnalyzeManager
    from idmtools.analysis.download_analyzer import DownloadAnalyzer

    # Clean up 'outputs' dir
    output_path = "tutorial_3_results"
    if os.path.exists( output_path ):
        shutil.rmtree( output_path )

    # files to be downloaded from each sim
    filenames = [
        'output/InsetChart.json',
        'output/ReportHIVByAgeAndGender.csv'
    ]
    analyzers = [ DownloadAnalyzer( filenames=filenames,␣
 ↪output_path=output_path ) ]

    manager = AnalyzeManager( platform=platform, analyzers=analyzers )
    manager.add_item( experiment )
    manager.analyze()
    return

def sweep_run_number( simulation, value ):
    simulation.task.config.parameters.Run_Number = value
    return { "Run_Number": value }
```

```
dtk.setup(local_dir=manifest.executables_dir)

#platform = Platform( "SLURM_LOCAL", job_directory="experiments" )
platform = Platform( "Calculon", node_group="idm_abcd", priority="Normal" )

task = emod_task.EMODHIVTask.from_default(
    eradication_path = manifest.eradication_path,
    schema_path      = manifest.schema_file,
    param_custom_cb  = build_config,
    campaign_builder = build_campaign,
    demog_builder    = build_demographics,
    ep4_path         = None
)
add_reports( task, manifest )
task.config.parameters.Report_HIV_Period = 365/6

#task.set_sif( path_to_sif=manifest.sif_path, platform=platform )
task.set_sif( path_to_sif=manifest.sif_path )

builder = SimulationBuilder()
builder.add_sweep_definition( sweep_run_number, [1,2,3] )

experiment = Experiment.from_builder( builder, task, name="Tutorial_3" )

experiment.run( wait_until_done=True, platform=platform )

# Check result
if experiment.succeeded:
    print(f"Experiment {experiment.uid} succeeded.")

    process_results( experiment, platform )

    print(f"Downloaded resuts for experiment {experiment.uid}.")
else:
    print(f"Experiment {experiment.uid} failed.\n")
```

```
/!\ WARNING: File 'idmtools.ini' Not Found! For details on how to configure
idmtools, see
https://docs.idmod.org/projects/idmtools/en/v1.7.1/configuration.html for
details on how to configure idmtools.
```

```
[Calculon]
{
   "endpoint": "https://comps.idmod.org",
   "environment": "Calculon"
}
```

```
Generating demographics file demographics.json.
Telling emod-api to use executables\schema.json as schema.


The created experiment can be viewed at https://comps.idmod.org/#explore/Simulat
ions?filters=ExperimentId=c39900a6-fe29-ef11-aa14-b88303911bc1
Simulations are still being created


Creating Simulations on Comps:
100%|                          | 3/3 [00:00<00:00,
4.94simulation/s]
Waiting on Experiment Tutorial_3 to Finish running:
100%|             | 3/3 [03:25<00:00, 68.46s/simulation]

Experiment c39900a6-fe29-ef11-aa14-b88303911bc1 succeeded.
Analyze Manager
 | 3 item(s) selected for analysis
 | partial_analyze_ok is False, max_items is None, and 0 item(s) are being

ignored
 | Analyzer(s):
 |  - DownloadAnalyzer File parsing: off / Use cache: off
 | Pool of 3 analyzing process(es)


100%|
     | 3/3 [00:04<00:00,  1.39s/it]
Running Analyzer Reduces:
100%|                                | 1/1
[00:00<00:00, 142.80it/s]

 | Analysis complete. Took 4.20 seconds (~ 1.401 per item)
Downloaded resuts for experiment c39900a6-fe29-ef11-aa14-b88303911bc1.
```

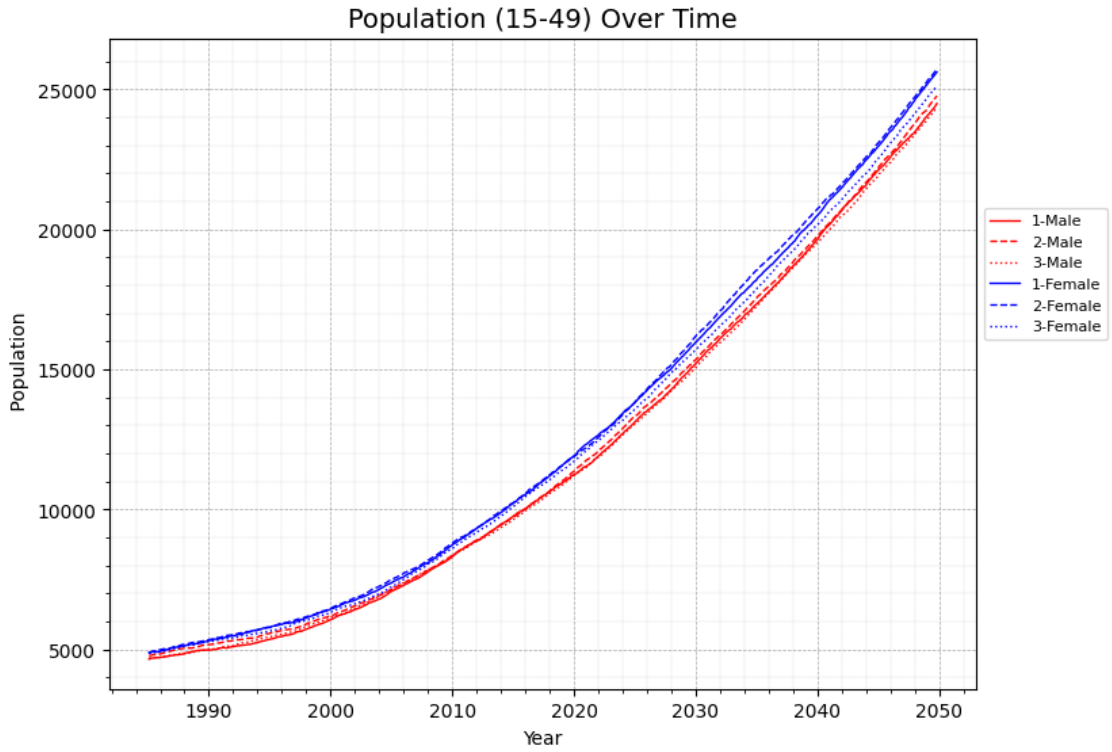
1.6 Plot The Resluts

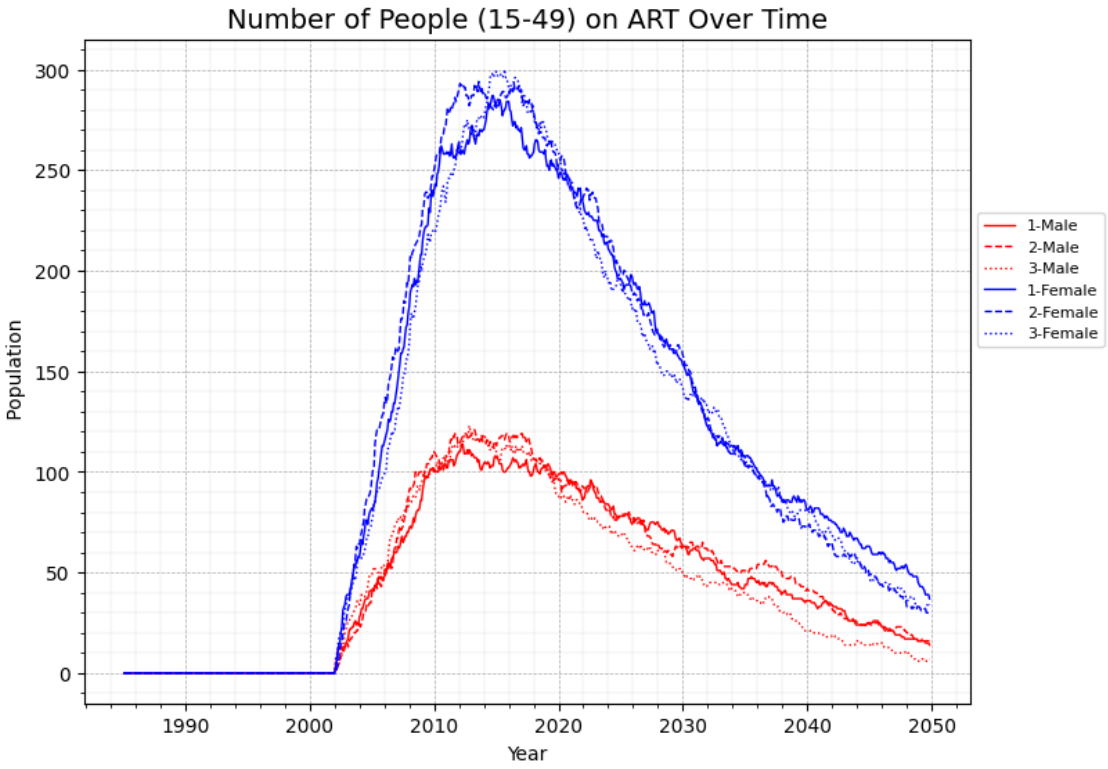
So how did our changes impact the results?

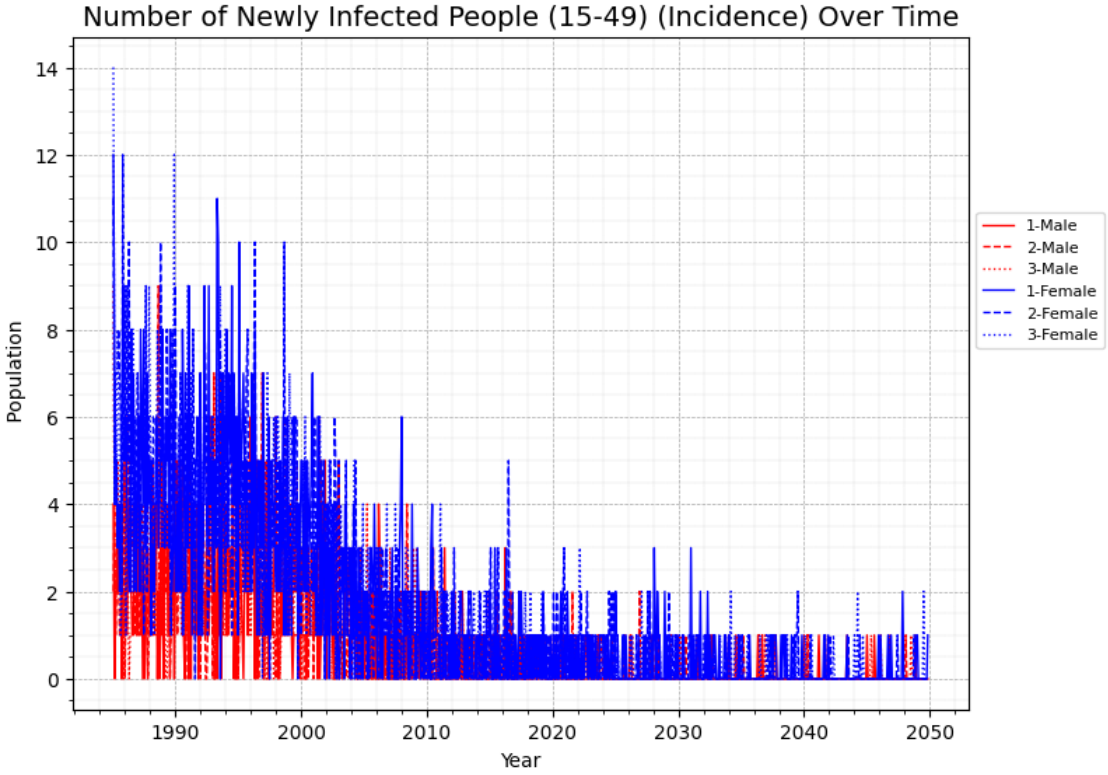
[5]:
```
import plot_report_hiv_by_age_and_gender as my_plt

report_filenames = my_plt.get_report_filenames( "tutorial_3_results",␣
 ↪"ReportHIVByAgeAndGender.csv" )
df = my_plt.create_dataframe_from_csv_reports( report_filenames )
num_runs = len(report_filenames)
my_plt.plot_age_based_data( df, num_runs,  "Population (15-49) Over Time", "␣
 ↪Population" )
```

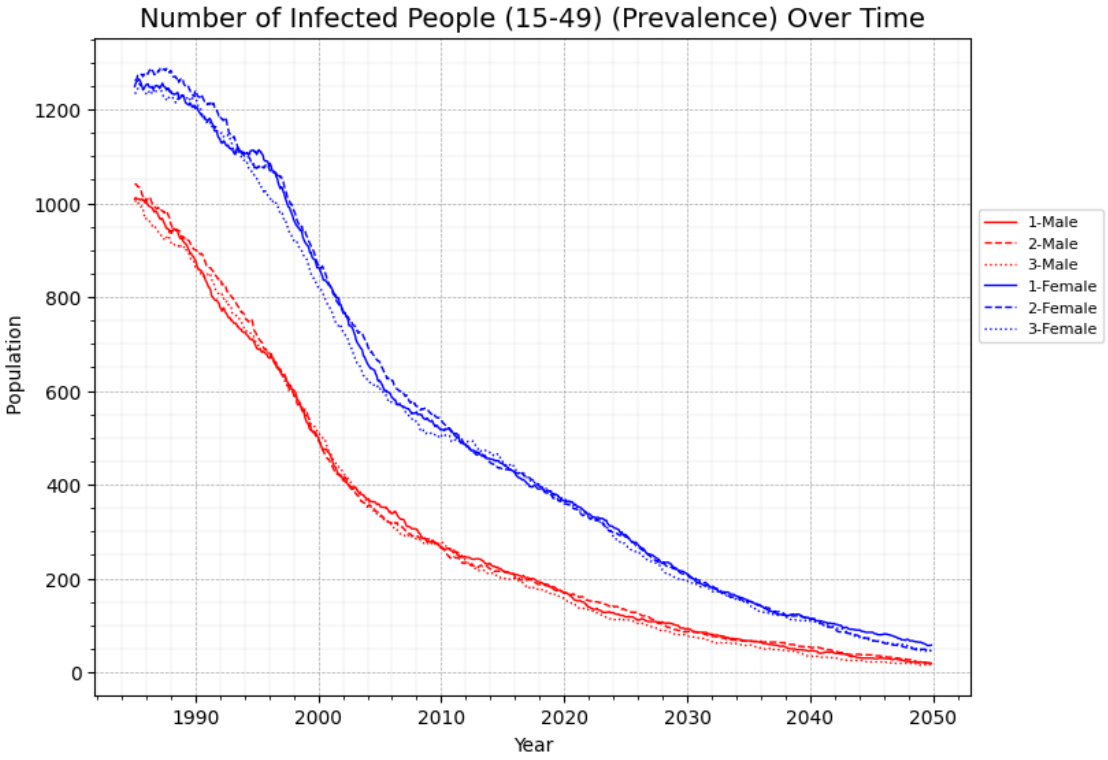
```
my_plt.plot_age_based_data( df, num_runs,  "Number of People (15-49) on ART␣
 ↪Over Time", " On_ART" )
my_plt.plot_age_based_data( df, num_runs,  "Number of Newly Infected People␣
 ↪(15-49) (Incidence) Over Time", " Newly Infected" )
my_plt.plot_age_based_data( df, num_runs,  "Number of Infected People (15-49)␣
 ↪(Prevalence) Over Time", " Infected" )
```

```
Reading tutorial_3_results\c49900a6-fe29-ef11-aa14-b88303911bc1\ReportHIVByAgeAn
dGender.csv
Reading tutorial_3_results\c59900a6-fe29-ef11-aa14-b88303911bc1\ReportHIVByAgeAn
dGender.csv
Reading tutorial_3_results\c69900a6-fe29-ef11-aa14-b88303911bc1\ReportHIVByAgeAn
dGender.csv
plotting Population (15-49) Over Time
plotting Number of People (15-49) on ART Over Time
plotting Number of Newly Infected People (15-49) (Incidence) Over Time
plotting Number of Infected People (15-49) (Prevalence) Over Time
```




Number of People (15-49) on ART Over Time
Population
Year
0
50
100
150
200
250
300
1990
2000
2010
2020
2030
2040
2050
1-Male
2-Male
3-Male
1-Female
2-Female
3-Female


Number of Newly Infected People (15-49) (Incidence) Over Time
Population
Year
0
2
4
6
8
10
12
14
1990
2000
2010
2020
2030
2040
2050
1-Male
2-Male
3-Male
1-Female
2-Female
3-Female



1.7 Next: Tutorial #4 - Overriding a state in the cascade of care

In this tutorial, our `build_campaign` method added some PrEP to our campaign. But what do you do when there is an existing state that you want to change? Tutorial #4 will provide an example of how to do it.