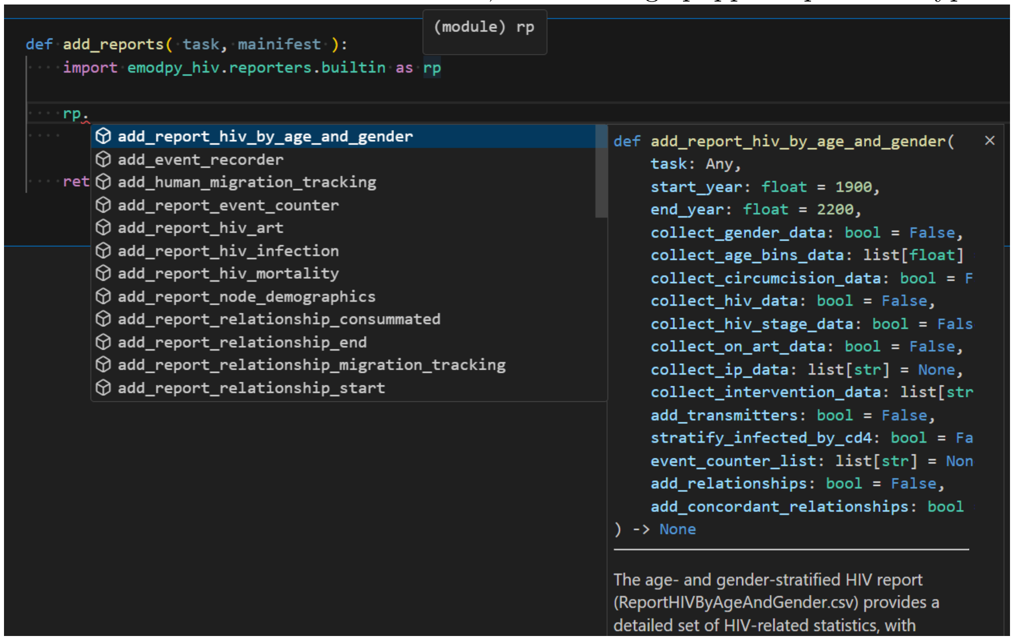
Tutorial-2_AddingReportsAndPlottingData

June 13, 2024

1 Tutorial #2 - Add reports, download the reports, and plot the data

The objective of this tutorial is to give you a very simple way of getting the results from your experiement. It will start with the code we used in Tutorial #1 and: - Show you how to add new reports - Show you how to download the reports - Plot the data in the reports so we can see how the simulation is performing

Below is an example of using an IDE to see what functions are available, what they do, and what the parameters are. In Visual Studio Code, these dialogs popped up after I typed the period in "rp.".


(module) rp
add_report_hiv_by_age_and_gender
add_event_recorder
add_human_migration_tracking
add_report_event_counter
add_report_hiv_art
add_report_hiv_infection
add_report_hiv_mortality
add_report_node_demographics
add_report_relationship_consummated
add_report_relationship_end
add_report_relationship_migration_tracking
add_report_relationship_start

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

    rp.

    ret
```

```
def add_report_hiv_by_age_and_gender(
    task: Any,
    start_year: float = 1900,
    end_year: float = 2200,
    collect_gender_data: bool = False,
    collect_age_bins_data: list[float]
    collect_circumcision_data: bool = F
    collect_hiv_data: bool = False,
    collect_hiv_stage_data: bool = Fals
    collect_on_art_data: bool = False,
    collect_ip_data: list[str] = None,
    collect_intervention_data: list[str
    add_transmitters: bool = False,
    stratify_infected_by_cd4: bool = Fa
    event_counter_list: list[str] = Non
    add_relationships: bool = False,
    add_concordant_relationships: bool
) -> None
```

The age- and gender-stratified HIV report (ReportHIVByAgeAndGender.csv) provides a detailed set of HIV-related statistics, with

1.1 Adding reports

To organize our logic, we will create a method that configures the reports we want EMOD to produce. EMOD is already generating the default InsetChart.json (by setting `config.parameters.Enable_Default_Reporting = 1`). We will add two more reports so you can see how it is done and get everyone's favorite `ReportHIVByAgeAndGender`.

[1]:
```
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
```

1.2 Downloading the reports

We add another function to call that will use the `idmtools` concept of an "analyzer". Analyzers are intended to be Python logic that you use to process the output of your simulations. In this tutorial, we will use the built-in `DownloadAnalyzer` to copy the reports to a directory called `tutorial_2_results`.

In this method, we will also use the `AnalyzeManager` to execute the `DownloadAnalyzer`. One could have multiple analyzers. Imagine you have multiple report files and want to summarize each of those reports separately. You could create an analyzer for each report.

[2]:
```
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_2_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
```

1.3 Code from Tutorial #1 plus `add_reports` and `process_results()`

The following code is the code we used in Tutorial #1, but all bunched together. Please note the following: - The imports have been moved to the top. - The `sweep_run_number()` function was placed right after the reports. - `add_reports()` is called right after the creation of the EMOD-Task. - Logic is added after `experiment.run()` to check if the experiement succeeded and call 'process_results()'.

[3]:
```
# 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 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" )

zambia = cm.Zambia()
task = emod_task.EMODHIVTask.from_default(
    eradication_path = manifest.eradication_path,
    schema_path      = manifest.schema_file,
    param_custom_cb  = zambia.build_config,
    campaign_builder = zambia.build_campaign,
    demog_builder    = zambia.build_demographics,
    ep4_path         = None
)
```

```python
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_2" )

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"

}
setting schema_path: executables\schema.json to campaign_builder function.
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=20b17de6-fc29-ef11-aa14-b88303911bc1

Simulations are still being created



Creating Simulations on Comps:
100%|                         | 3/3 [00:00<00:00,
3.46simulation/s]
Waiting on Experiment Tutorial_2 to Finish running:
100%|             | 3/3 [01:53<00:00, 37.98s/simulation]

Experiment 20b17de6-fc29-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 [00:05<00:00,  1.72s/it]
Running Analyzer Reduces:
100%|                             | 1/1
[00:00<00:00, 143.03it/s]

 | Analysis complete. Took 5.17 seconds (~ 1.723 per item)
Downloaded resuts for experiment 20b17de6-fc29-ef11-aa14-b88303911bc1.
```

1.4 Results found in tutorial_2_results

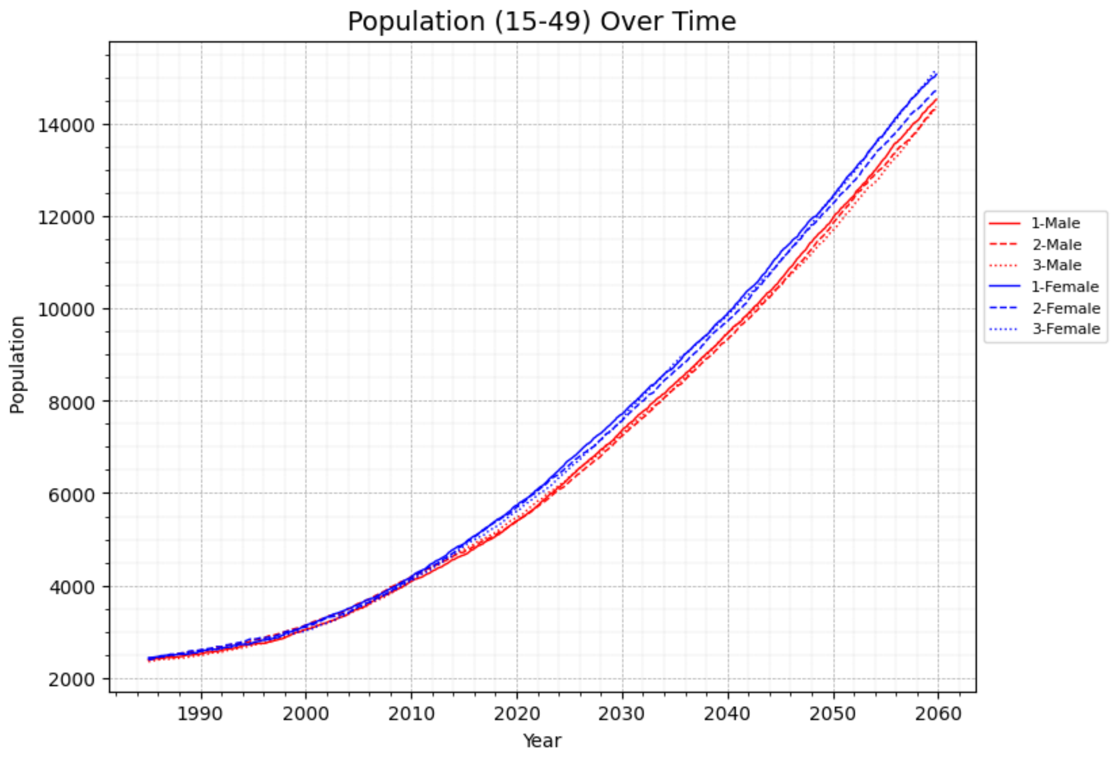
The DownloadAnalyzer should have created a directory called tutorial_2_results, which should contain three folders. The names of the folders are the IDs (GUIDs) of the simulations. Each simulation folder should contain the InsetChart.json and ReportHIVByAgeAndGender.csv reports.

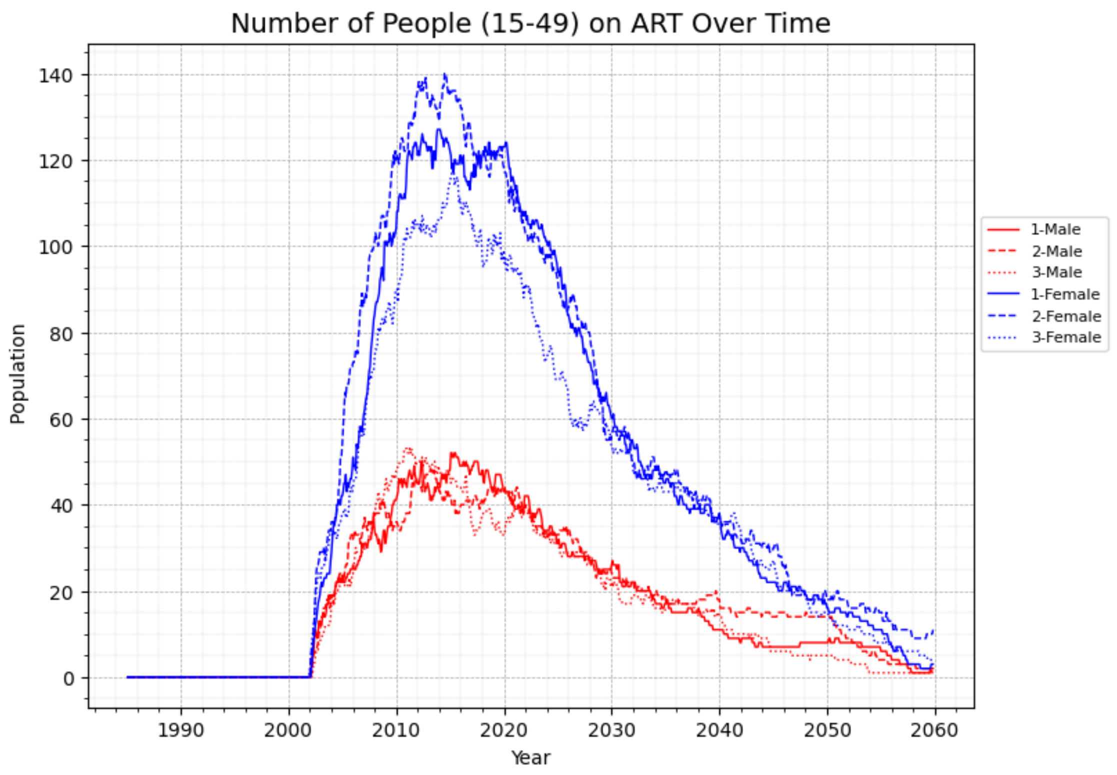
1.5 Plotting the results

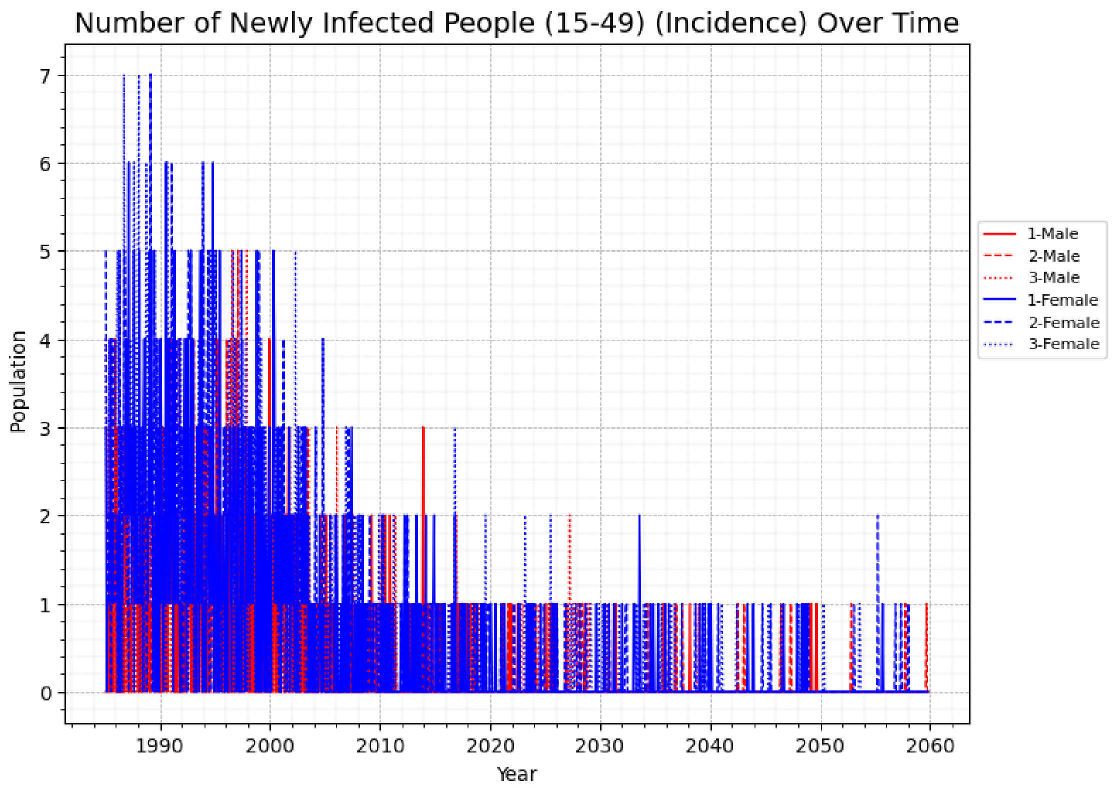
The following code uses some tutorial-helper functions to plot the results of these simulations.

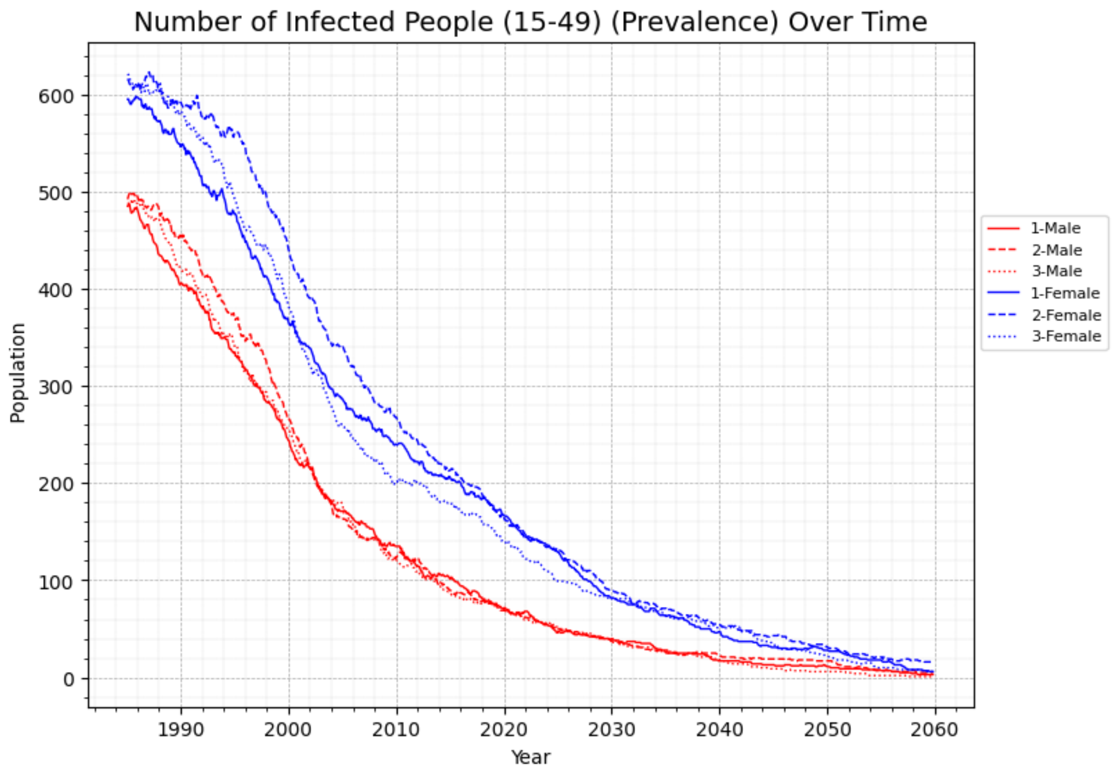
[4]:
```python
import plot_report_hiv_by_age_and_gender as my_plt

report_filenames = my_plt.get_report_filenames( "tutorial_2_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_2_results\21b17de6-fc29-ef11-aa14-b88303911bc1\ReportHIVByAgeAn
dGender.csv
Reading tutorial_2_results\22b17de6-fc29-ef11-aa14-b88303911bc1\ReportHIVByAgeAn
dGender.csv
Reading tutorial_2_results\23b17de6-fc29-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
1-Male
2-Male
3-Male
1-Female
2-Female
3-Female


Number of Newly Infected People (15-49) (Incidence) Over Time
Population
Year
1-Male
2-Male
3-Male
1-Female
2-Female
3-Female



1.6 Moving on to Tutorial #3

Tutorial #1 taught us the basics of running an experiment, and Tutorial #2 showed us how to get results from our experiement. Now, we need to learn more about how to make changes to our baseline country model for our specific project. Tutorial #3 will introduce us to the builder methods that we can use to make those modifications.