Tutorial-4_OverridingStateInCoC

June 13, 2024

1 Tutorial #4 - Overriding a state in the cascade of care

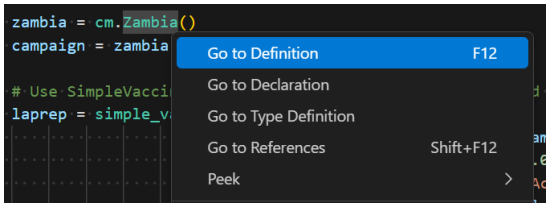
Sometimes the default cascade of care doesn't do exactly what you need. For instance, you might need to add a decision point to determine if someone gets Antiretroviral Therapy (ART) or whether they wait. Maybe you want to break ART into effective ART and non-suppressive ART. This tutorial provides you an example on how to customize the cascade of care to meet your needs.

1.1 To override state, subclass country model

We use the concept of a "country model" to combine the collection of data and code needed to model a particular country. As you may have noticed from the previous tutorials, the country model has its own build functions that we use for creating our `EMODTask`. The base class `Country` has a default `build_campaign()` method that defines a default set of methods to call for constructing the cascade of care. You can override this method or any of the methods that are part of the construction algorithm.

`<WARNING-ObjectOrientedDesign-Lesson>` This approach is similar to the Template Method design pattern. The main idea is that there is a main algorithm consisting of a collection of methods that implement the algorithm. One can subclass the class implementing the template and then override any portion of the algorithm that they'd like.`</WARNING-ObjectOrientedDesign-Leason>`

NOTE: This is one of those places where using an IDE that supports Python can really come in handy. If you were looking at the `build_campaign` function in the notebook for Tutorial #3 in Visual Studio Code, you could right-click on `cm.Zambia()`, select "Go to Definition", and it would take you to the code. Once you are looking at the code for Zambia, you can find the method you want to override, copy the code from the Zambia model, paste into your country model, and start modifying it.



The baseline `HCTUptakeAtDebut` state listens for the `STIDebut` built-in event. This event is broadcast when the person decides to start seeking sexual relationships. When this state hears this event, it will use a sigmoid distribution to determine the probability that the person will enter the testing loop or wait and consider it again later.

In this example, we override this method by making the baseline implementation only last the first 35 years (i.e., 1990 to 2025). In 2025, when it expires, we start a new listener for `STIDebut` that will distribute a very long-lasting PrEP and then send the person through the same sigmoid logic.

NOTE: In the last `add_triggered_event`, notice that it is distributing the same `uptake_choice` intervention as the first one. This is one of those benefits of using Python. If this were done in JSON, the configuration of `uptake_choise` would have to be copy and pasted. Not having this data duplcated multiple times should lead to fewer bugs.

[1]:
```
import emod_api
from emod_api.interventions.common import BroadcastEvent
from emodpy_hiv.country_model import Zambia
from emodpy_hiv.campaign import coc
from emodpy_hiv.interventions import sigmoiddiag
from emodpy_hiv.interventions.cascade_helpers import add_triggered_event
import emodpy_hiv.interventions.prep as prep

class MyZambia(Zambia):
    def __init__(self):
        super().__init__('MyZambia')

    def add_state_HCTUptakeAtDebut(self, campaign: emod_api.campaign, start_day:
 ↪ int):
        disqualifying_properties = [coc.CascadeState.LOST_FOREVER,
                                    coc.CascadeState.ON_ART,
                                    coc.CascadeState.LINKING_TO_ART,
                                    coc.CascadeState.ON_PRE_ART,
                                    coc.CascadeState.LINKING_TO_PRE_ART,
                                    coc.CascadeState.ART_STAGING]

        # ------------------------------------------------------------------
        # --- Do the same thing as baseline but only for 35 years / unitl 2025
        # --- i.e. stop listing for the STIDebut event so new code can
        # ------------------------------------------------------------------
        duration = 35 * 365 # this should cause the current logic to end in 2025
        uptake_choice = sigmoiddiag.new_diagnostic( campaign,
                                                    Positive_Event=coc.
 ↪HCT_TESTING_LOOP_TRIGGER,
                                                    Negative_Event=coc.
 ↪HCT_UPTAKE_POST_DEBUT_TRIGGER_1,
                                                    ramp_min=-0.005,
                                                    ramp_max=0.05,
                                                    ramp_midyear=2005,
                                                    ramp_rate=1,
                                                    female_multiplier=1.0,
                                                    ␣
 ↪disqualifying_properties=disqualifying_properties,
```

```
                                                    new_property_value=coc.
↪CascadeState.HCT_UPTAKE_AT_DEBUT)
      add_triggered_event( campaign,
                           event_name='HCTUptakeAtDebut: state 0 (decision,␣
↪sigmoid by year and sex)',
                           in_trigger=coc.CustomEvent.STI_DEBUT,
                           out_iv=[uptake_choice],
                           property_restrictions=['Accessibility:Yes'],
                           node_ids=None,
                           start_day=start_day,
                           duration=duration ) # !!! This is difference from␣
↪baseline code

      # ------------------------------------------------------------------
      # --- Insert the handing out of a very long lasting PrEP before making
      # --- the same sigmoid decision
      # ------------------------------------------------------------------
      laprep_start_day = start_day + duration
      sigmoid_event = "Enter_Health_Care_System"

      laprep = prep.new_intervention( campaign,
                                      efficacy_times=   [  0, 1825, 1900,␣
↪2000, 2100, 2200 ],
                                      efficacy_values=  [0.8, 0.8,   0.5,  0.
↪3,  0.1,  0.0 ],
                                      intervention_name="LA-PrEP",
                                     ␣
↪disqualifying_properties=disqualifying_properties,
                                      new_property_value=coc.CascadeState.
↪HCT_UPTAKE_AT_DEBUT )

      broadcast_sigmoid = BroadcastEvent( campaign, sigmoid_event )

      add_triggered_event( campaign,
                           event_name='Tutorial 4 - LA-PrEP on STIDebut',
                           in_trigger=coc.CustomEvent.STI_DEBUT,
                           out_iv=[laprep,broadcast_sigmoid],
                           property_restrictions=['Accessibility:Yes'],
                           node_ids=None,
                           start_day=laprep_start_day )

      # Distribute the uptake_choice but do it based on the sigmoid_event
      add_triggered_event( campaign,
                           event_name='HCTUptakeAtDebut: state 0 (decision,␣
↪sigmoid by year and sex)-Tutorial 4 LA-PrEP',
```

```
                                in_trigger=sigmoid_event, # event broadcast when␣
 ↪the person gets PrEP
                                out_iv=[uptake_choice],
                                property_restrictions=['Accessibility:Yes'],
                                node_ids=None,
                                start_day=laprep_start_day )

        return
```

1.2 Code from Tutorial #3

This is the same code that we had in Tutorial #3, except everywhere we used the the Zambia country model, we will now use the MyZambia model.

[2]:
```
def build_config( config ):

    # ------------------
    # --- Using MyZambia
    # ------------------
    zambia = MyZambia()
    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


def build_campaign( ):
    import manifest
    import emodpy_hiv.interventions.cascade_helpers as helpers
    import emodpy_hiv.interventions.prep as prep

    # ------------------
    # --- Using MyZambia
    # ------------------
    zambia = MyZambia()
    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

def build_demographics():

    # ------------------
    # --- Using MyZambia
    # ------------------
    zambia = MyZambia()
    demographics = zambia.build_demographics()

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

    return demographics


# 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_4_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_4" )

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=b61f850f-002a-ef11-aa14-b88303911bc1
Simulations are still being created


Creating Simulations on Comps:
100%|                          | 3/3 [00:00<00:00,
6.06simulation/s]
Waiting on Experiment Tutorial_4 to Finish running:
100%|             | 3/3 [04:06<00:00, 82.21s/simulation]
Experiment b61f850f-002a-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,  1.93s/it]
Running Analyzer Reduces:
100%|                                | 1/1
[00:00<00:00, 142.81it/s]
 | Analysis complete. Took 5.80 seconds (~ 1.933 per item)
```

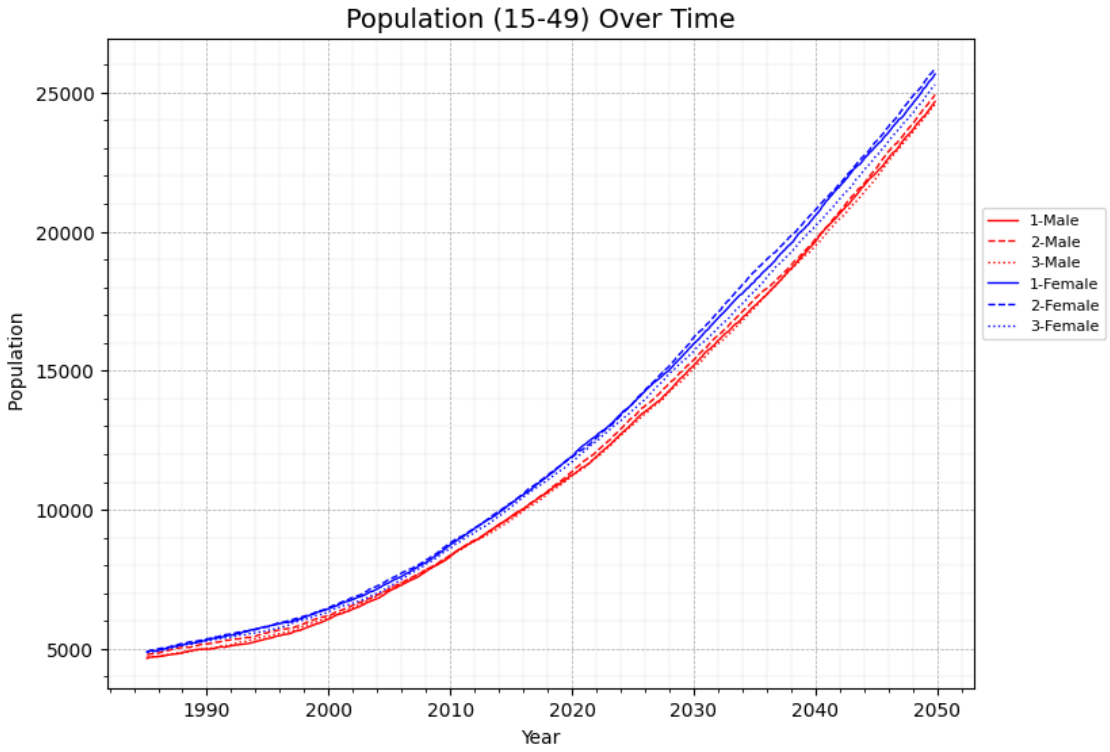
```
Downloaded resuts for experiment b61f850f-002a-ef11-aa14-b88303911bc1.
```

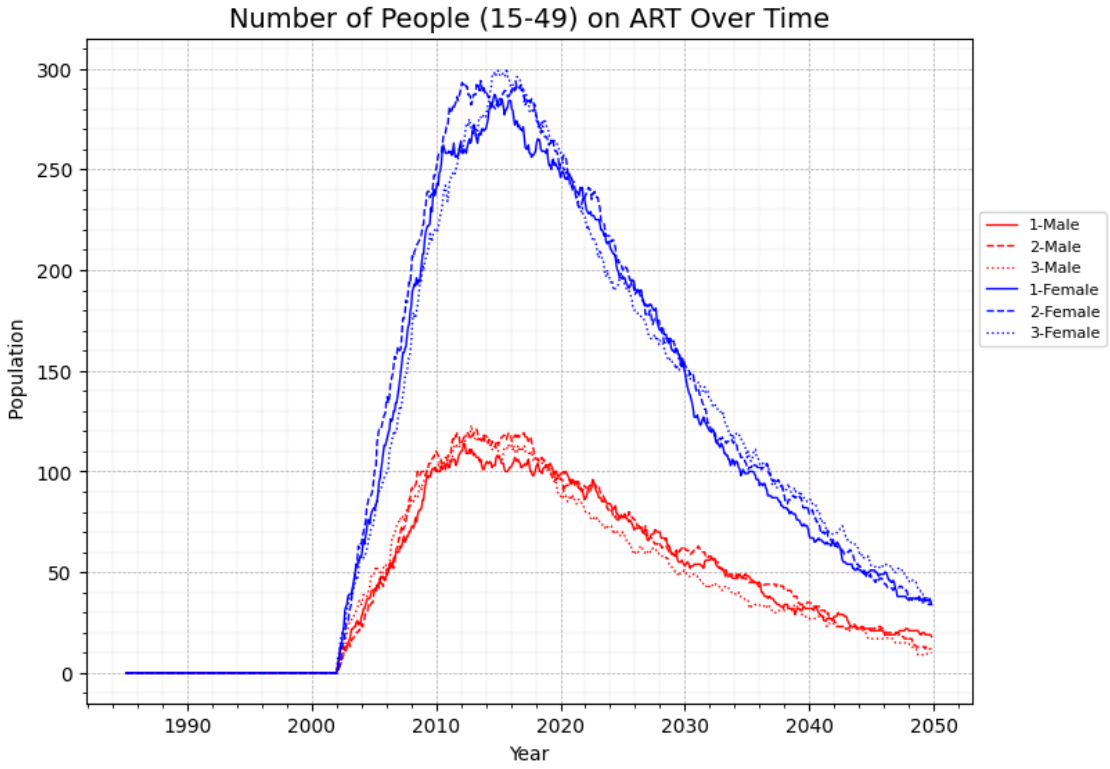
1.3 Plot the results

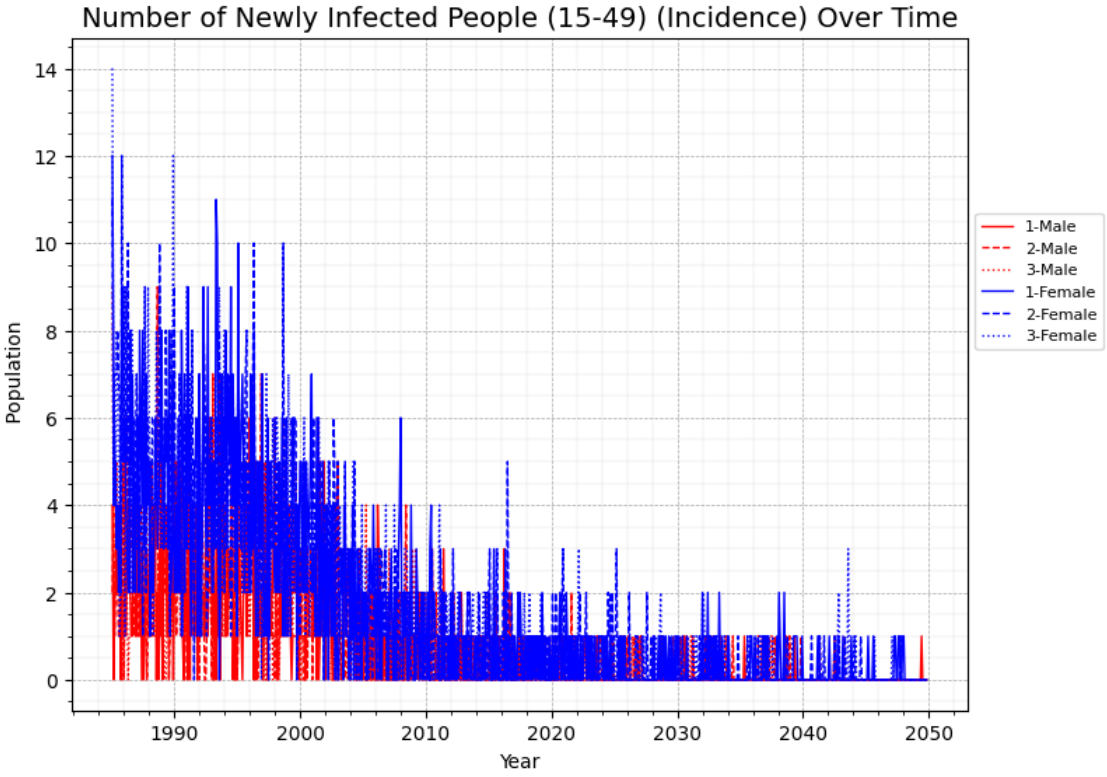
Did distributing LA-PrEP when the people debut change anything?

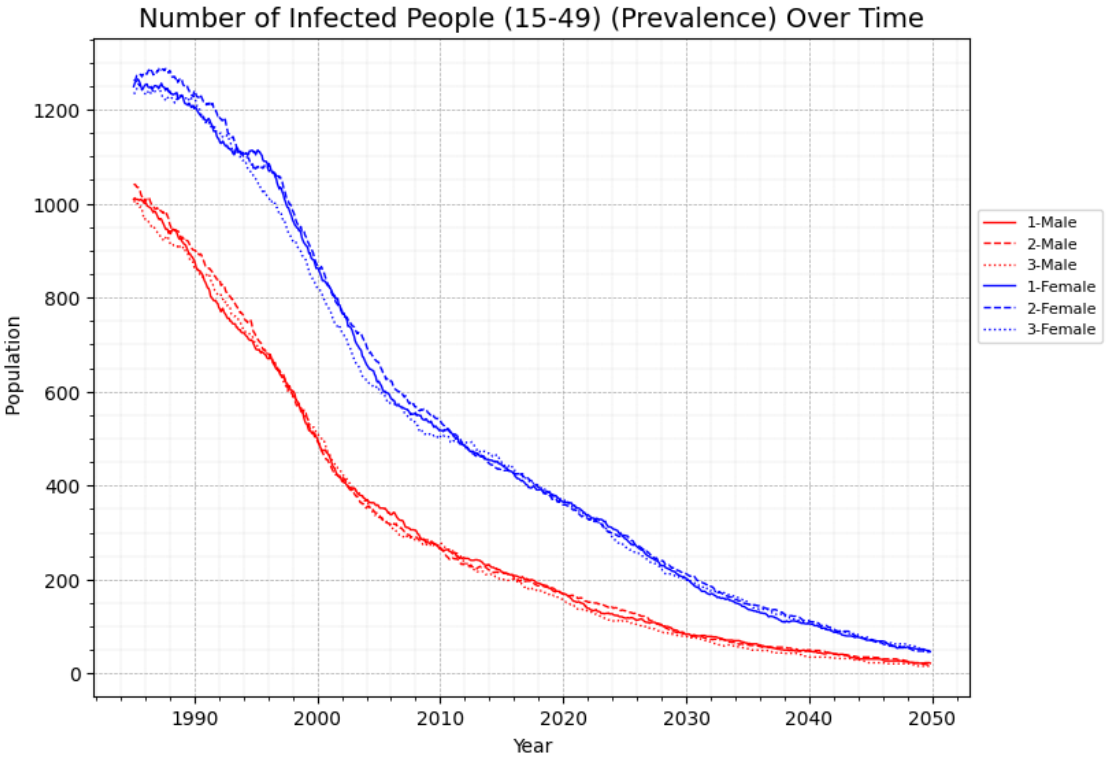
[3]:
```
import plot_report_hiv_by_age_and_gender as my_plt

report_filenames = my_plt.get_report_filenames( "tutorial_4_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_4_results\b71f850f-002a-ef11-aa14-b88303911bc1\ReportHIVByAgeAn
dGender.csv
Reading tutorial_4_results\b81f850f-002a-ef11-aa14-b88303911bc1\ReportHIVByAgeAn
dGender.csv
Reading tutorial_4_results\b91f850f-002a-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
```


Population (15-49) Over Time
Population
Year
25000
20000
15000
10000
5000
1990
2000
2010
2020
2030
2040
2050
1-Male
2-Male
3-Male
1-Female
2-Female
3-Female


Number of People (15-49) on ART Over Time
Population
Year
300
250
200
150
100
50
0
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


Number of Infected People (15-49) (Prevalence) Over Time
Population
Year
0
200
400
600
800
1000
1200
1990
2000
2010
2020
2030
2040
2050
1-Male
2-Male
3-Male
1-Female
2-Female
3-Female

[ ]: