emodpy-malaria

Institute for Disease Modeling

Jun 29, 2022

CONTENTS

emodpy-malaria is a collection of Python scripts and utilities created to streamline user interactions with EMOD and idmtools for modeling malaria. Much of the functionality is inherited from the emod_api package and emodpy package.

Additional information about how to use idmtools can be found in Welcome to idmtools. Additional information about EMOD malaria parameters can be found in EMOD parameter reference.

See Welcome to idmtools for a diagram showing how idmtools and each of the related packages are used in an end-to-end workflow using EMOD as the disease transmission model.

CHAPTER

ONE

INSTALLATION PREREQUISITES

First, ensure the following prerequisites are met before following the install instructions in *emodpy-malaria installation*.

- Windows 10 Pro or Enterprise, or Linux
- Python 3.9 64-bit (https://www.python.org/downloads/release)
- Git client, such as Git Bash or the Git GUI
- pip.ini (Windows) or pip.conf (Linux), containing the following:

```
[global]
index-url = https://packages.idmod.org/api/pypi/pypi-production/simple
```

1.1 emodpy-malaria installation

Follow the steps below to install emodpy-malaria.

Note: Currently, an IDM VPN connection is required to run the example.

1. Open a command prompt and create a virtual environment in any directory you choose. The command below names the environment "v-emodpy-malaria", but you may use any desired name:

```
python -m venv v-emodpy-malaria
```

2. Activate the virtual environment:

   - Windows
   - Linux

   Enter the following:

```
v-emodpy-malaria\Scripts\activate
```

   Enter the following:

```
source v-emodpy-malaria/bin/activate
```

3. Install emodpy-malaria packages:

```
pip install emodpy_malaria
```

If you are on Linux, also run:

```
pip install keyrings.alt
```

4. Open a command prompt and clone the emodpy-malaria GitHub repository to a local directory using the following command:

```
git clone https://github.com/InstituteforDiseaseModeling/emodpy-malaria.git
```

5. Verify installation by running the included Python example, `example.py`, located in /examples/start_here:

```
python example.py
```

Upon completion you can view the results in COMPS.

6. When you are finished, deactivate the virtual environment by entering the following at a command prompt:

```
deactivate
```

CHAPTER

TWO

API REFERENCE

2.1 emodpy_malaria package

2.1.1 Subpackages

emodpy_malaria.demographics package

Submodules

emodpy_malaria.demographics.MalariaDemographics module

This module contains the classes and functions for creating demographics files for malaria simulations. For more information on EMOD demographics files, see Demographics file.

class emodpy_malaria.demographics.MalariaDemographics.**MalariaDemographics**(*nodes*, *idref='Gridded world grump2.5arcmin'*, *base_file=None*, *init_prev=0.0*, *include_biting_heterogeneity=True*)

Bases: emod_api.demographics.Demographics.Demographics

This class is derived from emod_api.demographics.Demographics.Demographics and sets certain defaults for malaria in construction.

Parameters

- **nodes** – The number of nodes to create.
- **idref** – Method describing how the latitude and longitude values are created for each of the nodes in a simulation. "Gridded world" values use a grid overlaid across the globe at some arcsec resolution. You may also generate the grid using another tool or coordinate system. For more information, see Metadata.
- **base_file** – A basic demographics file used as a starting point for creating more complicated demographics files. For example, using a single node file to create a multi-node file for spatial simulations.
- **init_prev** – The initial malaria prevalence of the population. Defaults to 0%.
- **include_biting_heterogeneity** – variable biting rates. Defaults to on.

Returns None

set_risk_lowmedium()
Set initial risk for low-medium transmission settings per: https://wiki.idmod.org/display/MAL/Heterogeneous+biting+risk+in+simulations+vs+data.

set_risk_high()
Set initial risk for high transmission settings per: https://wiki.idmod.org/display/MAL/Heterogeneous+biting+risk+in+simulations+vs+data.

add_larval_habitat_multiplier(*schema*, *hab_type*, *multiplier*, *species='ALL_SPECIES'*, *node_id=0*)
Add LarvalHabitatMultiplier to node(s).

Parameters

- **schema** – Path to schema.json.
- **hab_type** – Habitat type.
- **multiplier** – Multiplier or Factor.
- **species** – Specific species (defaults to ALL).
- **node_id** – Nodes for this LHM. Defaults to all.

Returns Nothing.

add_initial_vectors_per_species(*init_vector_species*, *node_ids=None*)
Add an InitialVectorsForSpecies configuration for all nodes or just a set of nodes.

Parameters

- **init_vector_species** – Dictionary of vector species (strings) to initial populations. There is no checking for coherence of species named in other input settings.
- **node_ids** – Array of node ids. Defaults to None for all nodes.

Returns N/A.

add_initial_vectors_per_species_from_csv(*csv_path*)

Add initial vector species population to 'demographics' nodes from a csv file.

Parameters **csv_path** – Path to CSV file with the initial vector species populations for each node.

Returns N/A.

emodpy_malaria.demographics.MalariaDemographics.**from_template_node**(*lat=0*, *lon=0*, *pop=1000000.0*, *name=1*, *forced_id=1*, *init_prev=0.2*, *include_biting_heterogeneity=True*)
Create a single-node *MalariaDemographics* instance from the parameters you supply.

Parameters

- **lat** – Latitude of the centroid of the node to create.
- **lon** – Longitude of the centroid of the node to create.
- **pop** – Human population of the node.
- **name** – The name of the node. This may be a characteristic of the node, such as "rural" or "urban", or an identifying integer.
- **forced_id** – The node ID for the single node.

- **init_prev** – The initial malaria prevalence of the node.

Returns A *MalariaDemographics* instance.

emodpy_malaria.demographics.MalariaDemographics.**from_pop_csv**(*pop_filename_in*, *pop_filename_out='spatial_gridded_pop_dir'*, *site='No_Site'*)

Create a multi-node *MalariaDemographics* instance from a CSV file describing a population.

Parameters

- **pop_filename_in** – The path to the demographics file to ingest.
- **pop_filename_out** – The path to the file to output.
- **site** – A string to identify the country, village, or trial site.

Returns A *MalariaDemographics* instance

emodpy_malaria.demographics.MalariaDemographics.**from_csv**(*input_file*, *res=0.008333333333333333*, *id_ref='from_csv'*, *init_prev=0.0*, *include_biting_heterogeneity=True*)

Create a multi-node *MalariaDemographics* instance from a CSV file describing a population.

Parameters

- **input_file** – The path to the csv file to ingest.
- **res** – Resolution.
- **id_ref** – A string to identify the file, needs to match other input files.
- **init_prev** – The initial malaria prevalence of the population. Defaults to 0%.
- **include_biting_heterogeneity** – variable biting rates. Defaults to on.

Returns A *MalariaDemographics* instance

emodpy_malaria.demographics.MalariaDemographics.**from_params**(*tot_pop=1000000.0*, *num_nodes=100*, *frac_rural=0.3*, *id_ref='from_params'*)

Create a multi-node *MalariaDemographics* instance as a synthetic population based on a few parameters.

Parameters

- **tot_pop** – The total human population in the node.
- **num_nodes** – The number of nodes to create.
- **frac_rural** – The fraction of the population that is rural.
- **id_ref** – Method describing how the latitude and longitude values are created for each of the nodes in a simulation. "Gridded world" values use a grid overlaid across the globe at some arcsec resolution. You may also generate the grid using another tool or coordinate system. For more information, see Metadata.

Returns A *MalariaDemographics* instance.

emodpy_malaria.interventions package

Submodules

emodpy_malaria.interventions.adherentdrug module

emodpy_malaria.interventions.adherentdrug.**adherent_drug**(*campaign*, *cost: int = 1*, *doses: Optional[list] = None*, *dose_interval: int = 1*, *adherence_values: Optional[list] = None*, *non_adherence_options: Optional[list] = None*, *non_adherence_distribution: Optional[list] = None*, *max_dose_consideration_duration: int = 40*, *took_dose_event: str = 'Took_Dose'*, *intervention_name: Optional[str] = None*)

Configures adherent drug dictionary using the **AdherentDrug** class, an individual-level intervention which extends the **AntimalarialDrug** class.

Parameters

- **campaign** – campaign object to which the intervention will be added, and schema_path container
- **cost** – Unit cost per drug.
- **doses** – Lists of drugs for each dose. For example, `[["DrugA", DrugB"], ["DrugB"], [], ["DrugB"]]`. The empty list, `[]`, indicates no drugs for that dose.
- **dose_interval** – Interval between doses of drugs, in days. Default is 1.
- **adherence_values** – A list defining WaningEffectMapCount waning effect's "Values", to be used to set the probability for a particular dose. Where the "Times" is the dose number inferred from 'doses' parameter and "Values" is the probably of that dose being successfully taken.
- **non_adherence_options** – List of enums to define what happens when the user is not adherent. If not defined then NEXT_UPDATE is used. Enum values are: ["STOP", "NEXT_UPDATE", "NEXT_DOSAGE_TIME", "LOST_TAKE_NEXT"].
- **non_adherence_distribution** – Non adherence probability value(s) assigned to the corresponding options in non_adherence_options. There must be one value in this list for each value in non_adherence_options. The sum of these values must equal 1.0.
- **max_dose_consideration_duration** – Maximum number of days that an individual will consider taking the doses of the drug.
- **took_dose_event** – Event that gets sent out every time a dose is taken.
- **intervention_name** – The optional name used to refer to this intervention as a means to differentiate it from others that use the same class. Default is AdeherentDrug_drug1_drug2 in alphabetical order.

Returns: Configured **AdherentDrug** class dictionary

emodpy_malaria.interventions.bednet module

This module contains functionality for bednet distribution.

emodpy_malaria.interventions.bednet.**new_intervention_as_file**(*campaign*, *start_day*, *filename=None*)

Write a campaign file to disk with a single bednet event, using defaults. Useful for testing and learning.

Parameters

- **campaign** – The emod_api.campaign object to which the intervention will be added.
- **start_day** – The day of the simulation on which the bednets are distributed. We recommend aligning this with the start of the simulation.
- **filename** – The campaign filename; can be omitted and default will be used and returned to user.

Returns The campaign filename written to disk.

emodpy_malaria.interventions.bednet.**add_itn_scheduled**(*campaign*, *start_day: int = 0*, *coverage_by_ages: Optional[list] = None*, *demographic_coverage: float = 1.0*, *target_num_individuals: Optional[int] = None*, *node_ids: Optional[list] = None*, *repetitions: int = 1*, *timesteps_between_repetitions: int = 365*, *ind_property_restrictions: Optional[list] = None*, *receiving_itn_broadcast_event: Optional[str] = None*, *blocking_initial_effect: float = 0.9*, *blocking_box_duration: float = 0*, *blocking_decay_time_constant: float = 7300*, *killing_initial_effect: float = 0.6*, *killing_box_duration: int = 0*, *killing_decay_time_constant: float = 7300*, *repelling_initial_effect: float = 0*, *repelling_box_duration: float = 0*, *repelling_decay_time_constant: float = 0*, *usage_initial_effect: float = 1*, *usage_box_duration: float = 0*, *usage_decay_time_constant: float = 0*, *insecticide: str = ''*, *cost: float = 0*, *intervention_name: str = 'SimpleBednet'*)

Add a scheduled SimpleBednet intervention.

Parameters

- **campaign** – object for building, modifying, and writing campaign configuration files.
- **start_day** – Start day of intervention.
- **coverage_by_ages** – A list of dictionaries defining the coverage per age group. For example, `[{"coverage":1,"min": 1, "max": 10}, {"coverage":1,"min": 11, "max": 50}]`.
- **start_day** – The day the intervention is given out.

- **demographic_coverage** – This value is the probability that each individual in the target population will receive the intervention. It does not guarantee that the exact fraction of the target population set by Demographic_Coverage receives the intervention.
- **target_num_individuals** – The exact number of people to select out of the targeted group. If this value is set, demographic_coverage parameter is ignored
- **node_ids** – List of nodes to which to distribute the intervention. [] or None, indicates all nodes will get the intervention
- **repetitions** – The number of times an intervention is given, used with timesteps_between_repetitions. -1 means the intervention repeats forever. Sets **Number_Repetitions**
- **timesteps_between_repetitions** – The interval, in timesteps, between repetitions. Ignored if repetitions = 1. Sets **Timesteps_Between_Repetitions**.
- **ind_property_restrictions** – A list of dictionaries of IndividualProperties, which are needed for the individual to receive the intervention. Sets the **Property_Restrictions_Within_Node**. In the format `[{ "BitingRisk":"High"}, {"IsCool":"Yes}]`
- **receiving_itn_broadcast_event** – Optional. BroadcastEvent that's sent out when bednet is received. Default is to send out 'Received_ITN' event. To not send out event set to None.
- **killing_initial_effect** – Initial strength of the Killing effect. The effect may decay over time.
- **killing_box_duration** – Box duration of effect in days before the decay of Killing Initial_Effect.
- **killing_decay_time_constant** – The exponential decay length, in days of the Killing Initial_Effect.
- **blocking_initial_effect** – Initial strength of the Blocking effect. The effect may decay over time.
- **blocking_box_duration** – Box duration of effect in days before the decay of Blocking Initial_Effect.
- **blocking_decay_time_constant** – The exponential decay length, in days of the Blocking Initial_Effect.
- **repelling_initial_effect** – Initial strength of the Repelling effect. The effect may decay over time.
- **repelling_box_duration** – Box duration of effect in days before the decay of Repelling Initial_Effect.
- **repelling_decay_time_constant** – The exponential decay length, in days of the Repelling Initial_Effect.
- **usage_initial_effect** – Determines when and if an individual is using a bed net.
- **usage_box_duration** – ?
- **usage_decay_time_constant** – ?
- **insecticide** – The name of the insecticide defined in config.Insecticides for this intervention. If insecticides are being used, then this must be defined as one of those values. If they are not being used, then this does not needed to be specified or can be empty string. It cannot have a value if config.Insecticides does not define anything.

- **cost** – Unit cost per bednet
- **intervention_name** – The optional name used to refer to this intervention as a means to differentiate it from others that use the same class. It's possible to have multiple SimpleBednet interventions attached to a person if they have different Intervention_Name values.

Returns Nothing

emodpy_malaria.interventions.bednet.**add_itn_triggered**(*campaign*, *start_day: int = 0*, *demographic_coverage: float = 1.0*, *trigger_condition_list: Optional[list] = None*, *listening_duration: int = - 1*, *delay_period_constant: float = 0*, *node_ids: Optional[list] = None*, *repetitions: int = 1*, *timesteps_between_repetitions: int = 365*, *ind_property_restrictions: Optional[list] = None*, *receiving_itn_broadcast_event: Optional[str] = None*, *blocking_initial_effect: float = 0.9*, *blocking_box_duration: float = 0*, *blocking_decay_time_constant: float = 7300*, *killing_initial_effect: float = 0.6*, *killing_box_duration: int = 0*, *killing_decay_time_constant: float = 7300*, *repelling_initial_effect: float = 0*, *repelling_box_duration: float = 0*, *repelling_decay_time_constant: float = 0*, *usage_initial_effect: float = 1*, *usage_box_duration: float = 0*, *usage_decay_time_constant: float = 0*, *insecticide: str = ''*, *cost: float = 0*, *intervention_name: str = 'SimpleBednet'*)

Adds a triggered SimpleBednet intervention

Parameters

- **campaign** – object for building, modifying, and writing campaign configuration files.
- **start_day** – The day the intervention is given out.
- **demographic_coverage** – This value is the probability that each individual in the target population will receive the intervention. It does not guarantee that the exact fraction of the target population set by Demographic_Coverage receives the intervention.
- **trigger_condition_list** – A list of the events that will trigger intervention distribution.
- **listening_duration** – The number of time steps that the distributed event will monitor for triggers. Default is -1, which is indefinitely.
- **delay_period_constant** – Optional. Delay, in days, before the intervention is given out after a trigger is received.
- **node_ids** – List of nodes to which to distribute the intervention. [] or None, indicates all nodes will get the intervention
- **repetitions** – The number of times an intervention is given, used with timesteps_between_repetitions. -1 means the intervention repeats forever. Sets **Number_Repetitions**

- **timesteps_between_repetitions** – The interval, in timesteps, between repetitions. Ignored if repetitions = 1. Sets **Timesteps_Between_Repetitions**.
- **ind_property_restrictions** – A list of dictionaries of IndividualProperties, which are needed for the individual to receive the intervention. Sets the **Property_Restrictions_Within_Node**. In the format `[{ "BitingRisk":"High"}, {"IsCool":"Yes}]`
- **receiving_itn_broadcast_event** – Optional. BroadcastEvent that's sent out when bednet is received. Default is to send out 'Received_ITN' event. To not send out event set to None.
- **killing_initial_effect** – Initial strength of the Killing effect. The effect may decay over time.
- **killing_box_duration** – Box duration of effect in days before the decay of Killing Initial_Effect.
- **killing_decay_time_constant** – The exponential decay length, in days of the Killing Initial_Effect.
- **blocking_initial_effect** – Initial strength of the Blocking effect. The effect may decay over time.
- **blocking_box_duration** – Box duration of effect in days before the decay of Blocking Initial_Effect.
- **blocking_decay_time_constant** – The exponential decay length, in days of the Blocking Initial_Effect.
- **repelling_initial_effect** – Initial strength of the Repelling effect. The effect may decay over time.
- **repelling_box_duration** – Box duration of effect in days before the decay of Repelling Initial_Effect.
- **repelling_decay_time_constant** – The exponential decay length, in days of the Repelling Initial_Effect.
- **usage_initial_effect** – Determines when and if an individual is using a bed net.
- **usage_box_duration** – ?
- **usage_decay_time_constant** – ?
- **insecticide** – The name of the insecticide defined in config.Insecticides for this intervention. If insecticides are being used, then this must be defined as one of those values. If they are not being used, then this does not needed to be specified or can be empty string. It cannot have a value if config.Insecticides does not define anything.
- **cost** – Unit cost per bednet
- **intervention_name** – The optional name used to refer to this intervention as a means to differentiate it from others that use the same class. It's possible to have multiple SimpleBednet interventions attached to a person if they have different Intervention_Name values.

Returns Nothing

emodpy_malaria.interventions.common module

emodpy_malaria.interventions.common.**add_triggered_campaign_delay_event**(*campaign*, *start_day: int = 1*, *trigger_condition_list: Optional[list] = None*, *listening_duration: int = - 1*, *delay_period_constant: float = 0*, *demographic_coverage: float = 1.0*, *node_ids: Optional[list] = None*, *repetitions: int = 1*, *timesteps_between_repetitions: int = 365*, *ind_property_restrictions: Optional[list] = None*, *disqualifying_properties: Optional[list] = None*, *target_age_min: float = 0*, *target_age_max: float = 125*, *target_gender: str = 'All'*, *blackout_event_trigger: Optional[str] = None*, *blackout_period: float = 0*, *blackout_on_first_occurrence: bool = 0*, *individual_intervention: Optional[any] = None*)

Create and add campaign event that responds to a trigger after an optional delay with an intervention.

Parameters

- **campaign** – campaign object to which the intervention will be added, and schema_path container
- **start_day** – The day the intervention is given out.
- **trigger_condition_list** – A list of the events that will trigger intervention distribution.
- **listening_duration** – The number of time steps that the distributed event will monitor for triggers. Default is -1, which is indefinitely.
- **delay_period_constant** – Optional. Delay, in days, before the intervention is given out after a trigger is received.
- **demographic_coverage** – This value is the probability that each individual in the target population will receive the intervention. It does not guarantee that the exact fraction of the target population set by Demographic_Coverage receives the intervention.

- **node_ids** – List of nodes to which to distribute the intervention. [] or None, indicates all nodes will get the intervention
- **repetitions** – The number of times an intervention is given, used with timesteps_between_repetitions. -1 means the intervention repeats forever. Sets **Number_Repetitions**
- **timesteps_between_repetitions** – The interval, in timesteps, between repetitions. Ignored if repetitions = 1. Sets **Timesteps_Between_Repetitions**
- **ind_property_restrictions** – A list of dictionaries of IndividualProperties, which are needed for the individual to receive the intervention. Sets the **Property_Restrictions_Within_Node**
- **target_age_min** – The lower end of ages targeted for an intervention, in years. Sets **Target_Age_Min**
- **target_age_max** – The upper end of ages targeted for an intervention, in years. Sets **Target_Age_Max**
- **target_gender** – The gender targeted for an intervention: All, Male, or Female.
- **blackout_event_trigger** – The event to broadcast if an intervention cannot be distributed due to the **Blackout_Period**.
- **blackout_period** – After the initial intervention distribution, the time, in days, to wait before distributing the intervention again. If it cannot distribute due to the blackout period, it will broadcast the user-defined **Blackout_Event_Trigger**.
- **blackout_on_first_occurrence** – If set to true (1), individuals will enter the blackout period after the first occurrence of an event in the **Trigger_Condition_List***.
- **individual_intervention** – Individual intervention or a list of individual interventions to be distributed by this event

Returns Nothing

emodpy_malaria.interventions.common.**add_campaign_event**(*campaign*, *start_day: int = 1*, *demographic_coverage: float = 1.0*, *target_num_individuals: Optional[int] = None*, *node_ids: Optional[list] = None*, *repetitions: int = 1*, *timesteps_between_repetitions: int = 365*, *ind_property_restrictions: Optional[list] = None*, *target_age_min: float = 0*, *target_age_max: float = 125*, *target_gender: str = 'All'*, *individual_intervention: Optional[any] = None*, *node_intervention: Optional[any] = None*)

Adds a campaign event to the campaign with a passed in intervention.

Parameters

- **campaign** – campaign object to which the intervention will be added, and schema_path container
- **start_day** – The day the intervention is given out.

- **demographic_coverage** – This value is the probability that each individual in the target population will receive the intervention. It does not guarantee that the exact fraction of the target population set by Demographic_Coverage receives the intervention.
- **target_num_individuals** – The exact number of people to select out of the targeted group. If this value is set, demographic_coverage parameter is ignored
- **node_ids** – List of nodes to which to distribute the intervention. [] or None, indicates all nodes will get the intervention
- **repetitions** – The number of times an intervention is given, used with timesteps_between_repetitions. -1 means the intervention repeats forever. Sets **Number_Repetitions**
- **timesteps_between_repetitions** – The interval, in timesteps, between repetitions. Ignored if repetitions = 1. Sets **Timesteps_Between_Repetitions**
- **ind_property_restrictions** – A list of dictionaries of IndividualProperties, which are needed for the individual to receive the intervention. Sets the **Property_Restrictions_Within_Node**
- **target_age_min** – The lower end of ages targeted for an intervention, in years. Sets **Target_Age_Min**
- **target_age_max** – The upper end of ages targeted for an intervention, in years. Sets **Target_Age_Max**
- **target_gender** – The gender targeted for an intervention: All, Male, or Female.
- **individual_intervention** – Individual intervention or a list of individual interventions to be distributed by this event
- **node_intervention** – Node intervention or a list of node interventions to be distributed by this event

Returns Nothing

emodpy_malaria.interventions.community_health_worker module

emodpy_malaria.interventions.community_health_worker.**add_community_health_worker**(*campaign*, *start_day: int = 1*, *trigger_condition_list: Optional[list] = None*, *demographic_coverage: float = 1.0*, *node_ids: Optional[list] = None*, *ind_property_restrictions: Optional[list] = None*, *target_age_min: int = 0*, *target_age_max: int = 125*, *target_gender: str = 'All'*, *initial_amount: int = 6*, *amount_in_shipment: int = 2147480000*, *days_between_shipments: float = 7*, *duration: float = 3.40282e+38*, *intervention_config: Optional[any] = None*, *max_distributed_per_day: int = 2147480000*, *max_stock: int = 2147480000*, *waiting_period: int = 0*)

Sets up a CommunityHealthWorkerEventCoordinator with the passed in intervention

Parameters

- **campaign** – campaign object to which the intervention will be added, and schema_path container
- **start_day** – The day the intervention is given out.
- **trigger_condition_list** – The list of individual events that are of interest to the community health worker (CHW). If one of these events occurs, the individual or node is put into a queue to receive the CHW's intervention. The CHW processes the queue when the event coordinator is updated.
- **demographic_coverage** – This value is the probability that each individual in the target population will receive the intervention. It does not guarantee that the exact fraction of the target population set by Demographic_Coverage receives the intervention.
- **node_ids** – List of nodes to which to distribute the intervention. [] or None, indicates all nodes will get the intervention
- **ind_property_restrictions** – A list of dictionaries of IndividualProperties, which are needed for the individual to receive the intervention. Sets the **Property_Restrictions_Within_Node**
- **target_age_min** – The lower end of ages targeted for an intervention, in years. Sets **Target_Age_Min**
- **target_age_max** – The upper end of ages targeted for an intervention, in years. Sets **Target_Age_Max**
- **target_gender** – The gender targeted for an intervention: All, Male, or Female.
- **initial_amount** – Each instance will receive this constant/fixed value. Uses **Initial_Amount_Constant**
- **amount_in_shipment** – The number of interventions (such as vaccine doses) that a health worker or clinic receives in a shipment.
- **days_between_shipments** – The number of days to wait before a clinic or health worker receives a new shipment of interventions (such as vaccine doses)
- **duration** – The number of days for an event coordinator to be active before it expires. -1 means it never expires.
- **intervention_config** – A configured intervention to be distributed by the coordinator
- **max_distributed_per_day** – The maximum number of interventions (such as vaccine doses) that can be distributed by health workers or clinics in a given day
- **max_stock** – The maximum number of interventions (such as vaccine doses) that can be stored by a health worker or clinic
- **waiting_period** – The number of days a person or node can be in the queue waiting to get the intervention from the community health worker (CHW)

Returns Nothing

emodpy_malaria.interventions.diag_survey module

This module contains functionality for diagnostic survey interventions.

emodpy_malaria.interventions.diag_survey.**add_diagnostic_survey**(*campaign*, *coverage: float = 1*, *repetitions: int = 1*, *tsteps_btwn_repetitions: int = 365*, *target: object = 'Everyone'*, *start_day: int = 1*, *diagnostic_type: str = 'BLOOD_SMEAR_PARASITES'*, *diagnostic_threshold: float = 40*, *measurement_sensitivity: float = 0.1*, *event_name: str = 'Diagnostic Survey'*, *node_ids: Optional[list] = None*, *positive_diagnosis_configs: Optional[list] = None*, *negative_diagnosis_configs: Optional[list] = None*, *received_test_event: str = 'Received_Test'*, *ind_property_restrictions: Optional[list] = None*, *disqualifying_properties: Optional[list] = None*, *trigger_condition_list: Optional[list] = None*, *listening_duration: int = - 1*, *triggered_campaign_delay: int = 0*, *check_eligibility_at_trigger: bool = False*, *expire_recent_drugs: Optional[any] = None*)

Add an intervention to create either a scheduled or a triggered event to the campaign using the `MalariaDiagnostic` class, an individual-level class, to test individuals. Upon positive or negative diagnosis, the list of events to occur (as defined in **positive_diagnosis_configs** or **negative_diagnosis_configs**) is then executed. These events can trigger other listening interventions.

Parameters

- **camp** – The `emod_api.campaign` object for building, modifying, and writing campaign configuration files.
- **coverage** – The probability an individual receives the diagnostic.
- **repetitions** – Number of repetitions of the survey intervention.
- **tsteps_btwn_repetitions** – Timesteps between repetitions.
- **target** – A dictionary targeting an age range and gender of individuals for treatment. In the format `{"agemin": x, "agemax": y, "gender": z}`. Default is Everyone.
- **start_day** – The day of the simulation on which the intervention is created. If triggered, runs on trigger, not on **start_day**.
- **diagnostic_type** – Type of malaria diagnostic used. Default is **BLOOD_SMEAR_PARASITES**. Available options are:

  - BLOOD_SMEAR_PARASITES
  - BLOOD_SMEAR_GAMETOCYTES
  - PCR_PARASITES
  - PCR_GAMETOCYTES
  - PF_HRP2
  - TRUE_INFECTION_STATUS
  - TRUE_PARASITE_DENSITY
  - FEVER

- **diagnostic_threshold** – The diagnostic detection threshold based on **diagnostic_type**:

  TRUE_INFECTION_STATUS

  BLOOD_SMEAR_PARASITES In parasites/microliter, use measurement = float(1.0 / measurementSensitivity * Poisson(measurementSensitivity * true_parasite_density)).

  BLOOD_SMEAR_GAMETOCYTES In gametocytes/microliter, use measurement = float(1.0 / measurementSensitivity * Poisson(measurementSensitivity * true_gametocyte_density)).

  PCR_PARASITES and PCR_GAMETOCYTES Use the true values and an algorithm based on the paper Improving statistical inference on pathogen densities estimated by quantitative molecular methods : malaria gametocytaemia as a case study.

  PF_HRP2 Add a new method to get the PfHRP2 value and check against the threshold.

  TRUE_PARASITE_DENSITY Check the true parasite density against the threshold.

  FEVER Check the person's fever against the threshold.

- **measurement_sensitivity** – Setting for **Measurement_Sensitivity** in `MalariaDiagnostic`.
- **event_name** – Description of the event.
- **node_ids** – The list of nodes to apply this intervention to (**Node_List** parameter). If not provided, set value of NodeSetAll.
- **positive_diagnosis_configs** – List of events to happen to an individual who receives a positive result from test.
- **negative_diagnosis_configs** – List of events to happen to individual who receives a negative result from test.
- **received_test_event** – String for individuals to broadcast upon receiving diagnostic.
- **ind_property_restrictions** – List of IndividualProperty key:value pairs that individuals must have to receive the diagnostic intervention. For example, `[{"IndividualProperty1":"PropertyValue1"}, {"IndividualProperty2":"PropertyValue2"}]`. Default is no restrictions.
- **disqualifying_properties** – List of IndividualProperty key:value pairs that cause an intervention to be aborted. For example, `[{"IndividualProperty1":"PropertyValue1"}, {"IndividualProperty2":"PropertyValue2"}]`.
- **trigger_condition_list** – List of events that will trigger a diagnostic survey.

- **listening_duration** – Duration after start day to stop listening for events, as specified in **trigger_condition_list**. Default is -1, non-stop listening.
- **triggered_campaign_delay** – Delay of running the intervention after receiving a trigger from the **trigger_condition_list**.
- **check_eligibility_at_trigger** – If triggered event is delayed, you have an option to check individual/node's eligibility at the initial trigger or when the event is actually distributed after delay.
- **expire_recent_drugs** – Adds `[{"DrugStatus:None"}]` to **Property_Restrictions_Within_Node** for positive test config, so only those with that property receive drugs.

Returns None

emodpy_malaria.interventions.drug module

emodpy_malaria.interventions.drug.**add_scheduled_antimalarial_drug**(*campaign*, *start_day: int = 1*, *demographic_coverage: float = 1.0*, *target_num_individuals: Optional[int] = None*, *node_ids: Optional[list] = None*, *repetitions: int = 1*, *timesteps_between_repetitions: int = 365*, *ind_property_restrictions: Optional[list] = None*, *target_age_min: int = 0*, *target_age_max: int = 125*, *target_gender: str = 'All'*, *drug_type: Optional[str] = None*, *cost_to_consumer: float = 0*, *intervention_name: Optional[str] = None*)

Add an antimalarial drug intervention to your campaign. This is equivalent to AntimalarialDrug.

Parameters

- **campaign** – The `emod_api.campaign` object to which the intervention will be added.
- **start_day** – The day the intervention is given out.
- **demographic_coverage** – This value is the probability that each individual in the target population will receive the intervention. It does not guarantee that the exact fraction of the target population set by Demographic_Coverage receives the intervention.
- **target_num_individuals** – The exact number of people to select out of the targeted group. If this value is set, demographic_coverage parameter is ignored
- **node_ids** – List of nodes to which to distribute the intervention. [] or None, indicates all nodes will get the intervention
- **repetitions** – The number of times an intervention is given, used with timesteps_between_repetitions. -1 means the intervention repeats forever. Sets **Number_Repetitions**

- **timesteps_between_repetitions** – The interval, in timesteps, between repetitions. Ignored if repetitions = 1. Sets **Timesteps_Between_Repetitions**
- **ind_property_restrictions** – A list of dictionaries of IndividualProperties, which are needed for the individual to receive the intervention. Sets the **Property_Restrictions_Within_Node**
- **target_age_min** – The lower end of ages targeted for an intervention, in years. Sets **Target_Age_Min**
- **target_age_max** – The upper end of ages targeted for an intervention, in years. Sets **Target_Age_Max**
- **target_gender** – The gender targeted for an intervention: All, Male, or Female.
- **drug_type** – The name of the drug to distribute in a drug intervention. Possible values are contained in **Malaria_Drug_Params** in Drugs and treatments. Use `set_team_drug_params()` to set those values
- **cost_to_consumer** – Per-unit cost when drug is distributed
- **intervention_name** – The optional name used to refer to this intervention as a means to differentiate it from others that use the same class. Default is AntimalarialDrug_<drug_type>.

Returns The intervention event.

emodpy_malaria.interventions.drug.**new_intervention_as_file**(*campaign*, *start_day*, *drug_type='Chloroquine'*, *filename='AntimalarialDrug.json'*)

Take an AntimalarialDrug intervention from a JSON file and add it to your campaign.

Parameters

- **campaign** – The `emod_api.campaign` object to which the intervention will be added.
- **start_day** – The day of the simulation on which the drug is distributed. We recommend aligning this with the start of the simulation.
- **drug_type** – The name of the drug to distribute in a drug intervention. Possible values are contained in **Malaria_Drug_Params** in Drugs and treatments. Use `set_team_drug_params()` to set those values
- **filename** – The JSON file that contains the intervention.

Returns The filename.

emodpy_malaria.interventions.drug_campaign module

This module contains functionality for malaria intervention distribution via a cascade of care that may contain diagnostics and drug treatments.

emodpy_malaria.interventions.drug_campaign.**drug_configs_from_code**(*campaign*, *drug_code: Optional[str] = None*)

Add a single or multiple drug regimen to the configuration file based on its code and add the corresponding AntimalarialDrug intervention to the return dictionary. For example, passing the `ALP` drug code will add the drug configuration for artemether, lumefantrine, and primaquine to the configuration file and will return a dictionary containing a full treatment course for those three drugs. For more information, see **Malaria_Drug_Params** in Drugs and treatments.

Parameters

- **campaign** – The `emod_api.campaign` object to which the intervention will be added.

- **drug_code** – The code of the drug regimen. This must be listed in the drug_cfg dictionary.

Returns A dictionary containing the intervention for the given drug regimen.

emodpy_malaria.interventions.drug_campaign.**add_drug_campaign**(*campaign*, *campaign_type: str = 'MDA'*, *drug_code: Optional[str] = None*, *start_days: Optional[list] = None*, *coverage: float = 1.0*, *repetitions: int = 1*, *tsteps_btwn_repetitions: int = 60*, *diagnostic_type: str = 'BLOOD_SMEAR_PARASITES'*, *diagnostic_threshold: float = 40*, *measurement_sensitivity: float = 0.1*, *fmda_radius: int = 0*, *node_selection_type: str = 'DISTANCE_ONLY'*, *trigger_coverage: float = 1.0*, *snowballs: int = 0*, *treatment_delay: int = 0*, *triggered_campaign_delay: int = 0*, *node_ids: Optional[list] = None*, *target_group: any = 'Everyone'*, *drug_ineligibility_duration: int = 0*, *ind_property_restrictions: Optional[list] = None*, *disqualifying_properties: Optional[list] = None*, *trigger_condition_list: Optional[list] = None*, *listening_duration: int = -1*, *adherent_drug_configs: Optional[list] = None*, *target_residents_only: int = 1*, *check_eligibility_at_trigger: bool = False*, *receiving_drugs_event_name='Received_Campaign_Drugs'*)

Add a drug intervention campaign from a list of malaria campaign types. This intervention uses the MalariaDiagnostic class to create either a scheduled or a triggered event to the campaign and the AntimalarialDrug class to configure drug interventions. You can also specify a delay period for a triggered event that broadcasts afterwards. If the campaign is repeated or triggered, separate NodeLevelHealthTriggeredIV interventions are created with a delay that sends an event to distribute drugs.

Parameters

- **campaign** – The emod_api.campaign object to which the intervention will be added.
- **campaign_type** – The type of drug campaign. Available options are:

  MDA Add a mass drug administration intervention.

  MSAT Add a mass screening and treatment intervention.

  SMC Add a seasonal malaria chemoprevention intervention.

  fMDA Add a focal mass drug administration intervention based on results from a diagnostic survey, which is either scheduled or triggered (when **trigger_condition_list** is present).

  MTAT Add a mass testing and treatment intervention.

rfMSAT Add a reactive focal mass screening and treatment intervention. Detecting malaria triggers diagnostic surveys to run on neighboring nodes and so on, up to the number of triggered interventions defined in the **snowballs** parameter.

rfMDA Add a reactive focal mass drug administration intervention. This triggers BroadcastEventToOtherNodes to broadcast a "Give_Drugs_rfMDA" event, which triggers MultiInterventionDistributor to distribute drugs and a "ReceivedTreatment" event followed by a delayed "Give_Drugs_rfMDA" event to neighboring nodes, which will trigger another drug distribution.

- **drug_code** – The code of the drug regimen to distribute. This must be listed in the `drug_cfg` dictionary.
- **start_days** – List of start days (integers) when the drug regimen will be distributed. Due to diagnostic/treatment configuration, the earliest start day is 1. When **trigger_condition_list** is used, the first entry of **start_days** is the day to start listening for the trigger(s).
- **coverage** – The demographic coverage of the distribution (the fraction of people at home during the campaign).
- **repetitions** – The number of repetitions.
- **tsteps_btwn_repetitions** – The timesteps between the repetitions.
- **diagnostic_type** – The setting for **Diagnostic_Type** in MalariaDiagnostic. In addition to the accepted values listed there, you may also set TRUE_INFECTION_STATUS, which calls StandardDiagnostic instead.
- **diagnostic_threshold** – The setting for **Diagnostic_Threshold** in MalariaDiagnostic.
- **measurement_sensitivity** – The setting for **Measurement_Sensitivity** in MalariaDiagnostic.
- **detection_threshold** – The setting for **Detection_Threshold** in MalariaDiagnostic.
- **fmda_radius** – Radius (in km) of focal response upon finding infection. Used in simulations with many small nodes to simulate community health workers distributing drugs to surrounding houses. Used when **campaign_type** is set to fMDA.
- **node_selection_type** – The setting for **Node_Selection_Type** in BroadcastEventToOtherNodes.
- **trigger_coverage** – The fraction of trigger events that will trigger reactive case detection (RCD). Used when **campaign_type** is set to rfMSAT or rfMDA. To set the fraction of individuals reached during RCD response, use **coverage**.
- **snowballs** – The number of times each triggered intervention will be distributed to surrounding nodes. For example, one snowball gives drugs to nodes neighboring the first node and two snowballs gives drugs to the nodes neighboring those nodes. Used when **campaign_type** is set to rfMSAT.
- **treatment_delay** – For **campaign_type** set to MSAT or fMDA, the length of time between administering a diagnostic and giving drugs; for values of rfMSAT or rfMDA, the length of time between treating the index case and triggering an RCD response.
- **triggered_campaign_delay** – When using **trigger_condition_list**, this indicates the delay period between receiving the trigger event and running the triggered campaign intervention.
- **node_ids** – The setting for **Node_List** in Nodeset_Config classes.
- **target_group** – A dictionary of `{'agemin': x, 'agemax': y}` to target MDA, SMC, MSAT, fMDA to individuals between x and y years of age. Default is Everyone.

- **drug_ineligibility_duration** – The number of days to set the **DrugStatus** individual property to **RecentDrug**, after which the property value is reverted. This property value prevents people from receiving drugs too frequently, but they can still receive diagnostics during this period. For more information, see Targeting interventions to nodes or individuals.
- **ind_property_restrictions** – The setting for **Property_Restrictions_Within_Node** in TriggeredEventCoordinator that individuals must have to receive the diagnostic intervention.
- **disqualifying_properties** – The setting for **Disqualifying_Properties** in AntimalarialDrug or in MalariaDiagnostic.
- **trigger_condition_list** – The setting for **Start_Trigger_Condition_List** in TriggeredEventCoordinator.
- **listening_duration** – The setting for **Duration** in TriggeredEventCoordinator.
- **adherent_drug_configs** – List of adherent drug configurations, which are dictionaries from configure_adherent_drug.
- **target_residents_only** – The setting for **Target_Residents_Only** in TriggeredEventCoordinator.
- **check_eligibility_at_trigger** – Set to True to check the individual or node's eligibility at the initial trigger; set to False to check eligibility when the event is actually distributed after a delay.
- **receiving_drugs_event_name** – The event to broadcast when a person receives drugs.

Returns A dictionary with drug campaign parameters.

emodpy_malaria.interventions.drug_campaign.**add_MDA**(*campaign*, *start_days: Optional[list] = None*, *coverage: float = 1.0*, *drug_configs: Optional[list] = None*, *receiving_drugs_event: Optional[emod_api.interventions.common.BroadcastEvent] = None*, *repetitions: int = 1*, *tsteps_btwn_repetitions: int = 60*, *node_ids: Optional[list] = None*, *expire_recent_drugs: Optional[emod_api.interventions.common.PropertyValueChanger] = None*, *ind_property_restrictions: Optional[list] = None*, *disqualifying_properties: Optional[list] = None*, *target_group: any = 'Everyone'*, *trigger_condition_list: Optional[list] = None*, *listening_duration: int = - 1*, *triggered_campaign_delay: int = 0*, *target_residents_only: int = 1*, *check_eligibility_at_trigger: bool = False*)

Add an MDA (mass drug administration) drug intervention to your campaign. See `add_drug_campaign()` for more information about each argument.

Returns None

emodpy_malaria.interventions.drug_campaign.**add_MSAT**(*campaign*, *start_days: Optional[list] = None*, *coverage: float = 1.0*, *drug_configs: Optional[list] = None*, *receiving_drugs_event: Optional[emod_api.interventions.common.BroadcastEvent] = None*, *repetitions: int = 1*, *tsteps_btwn_repetitions: int = 60*, *treatment_delay: int = 0*, *diagnostic_type: str = 'BLOOD_SMEAR_PARASITES'*, *diagnostic_threshold: float = 40*, *measurement_sensitivity: float = 0.1*, *node_ids: Optional[list] = None*, *expire_recent_drugs: Optional[emod_api.interventions.common.PropertyValueChanger] = None*, *ind_property_restrictions: Optional[list] = None*, *disqualifying_properties: Optional[list] = None*, *target_group: any = 'Everyone'*, *trigger_condition_list: Optional[list] = None*, *triggered_campaign_delay: int = 0*, *listening_duration: int = - 1*, *check_eligibility_at_trigger: bool = False*)

Add an MSAT (mass screening and treatment) drug intervention to your campaign. See `add_drug_campaign()` for more information about each argument.

Returns None

emodpy_malaria.interventions.drug_campaign.**add_fMDA**(*campaign*, *start_days: Optional[list] = None*, *trigger_coverage: float = 1*, *coverage: float = 1*, *drug_configs: Optional[list] = None*, *receiving_drugs_event: Optional[emod_api.interventions.common.BroadcastEvent] = None*, *repetitions: int = 1*, *tsteps_btwn_repetitions: int = 365*, *treatment_delay: int = 0*, *diagnostic_type: str = 'BLOOD_SMEAR_PARASITES'*, *diagnostic_threshold: float = 40*, *measurement_sensitivity: float = 0.1*, *fmda_radius: int = 0*, *node_selection_type: str = 'DISTANCE_ONLY'*, *node_ids: Optional[list] = None*, *expire_recent_drugs: Optional[emod_api.interventions.common.PropertyValueChanger] = None*, *ind_property_restrictions: Optional[list] = None*, *disqualifying_properties: Optional[list] = None*, *target_group: any = 'Everyone'*, *trigger_condition_list: Optional[list] = None*, *listening_duration: int = - 1*, *triggered_campaign_delay: int = 0*, *check_eligibility_at_trigger: bool = False*)

Add an fMDA (focal mass drug administration) drug intervention to your campaign. See `add_drug_campaign()` for more information about each argument.

Returns None

emodpy_malaria.interventions.drug_campaign.**add_rfMSAT**(*campaign*, *start_day: int = 0*, *coverage: float = 1*, *drug_configs: Optional[list] = None*, *receiving_drugs_event: Optional[emod_api.interventions.common.BroadcastEvent] = None*, *listening_duration: int = - 1*, *treatment_delay: int = 0*, *trigger_coverage: float = 1*, *diagnostic_type: str = 'BLOOD_SMEAR_PARASITES'*, *diagnostic_threshold: float = 40*, *measurement_sensitivity: float = 0.1*, *fmda_radius: int = 0*, *node_selection_type: str = 'DISTANCE_ONLY'*, *snowballs: int = 0*, *node_ids: Optional[list] = None*, *expire_recent_drugs: Optional[emod_api.interventions.common.PropertyValueChanger] = None*, *ind_property_restrictions: Optional[list] = None*, *disqualifying_properties: Optional[list] = None*)

Add a rfMSAT (reactive focal mass screening and treatment) drug intervention to your campaign. See *add_drug_campaign()* for more information about each argument.

Returns None

emodpy_malaria.interventions.drug_campaign.**add_rfMDA**(*campaign*, *start_day: int = 0*, *coverage: float = 1*, *drug_configs: Optional[list] = None*, *receiving_drugs_event: Optional[emod_api.interventions.common.BroadcastEvent] = None*, *listening_duration: int = - 1*, *treatment_delay: int = 0*, *trigger_coverage: float = 1*, *fmda_radius: int = 0*, *node_selection_type: str = 'DISTANCE_ONLY'*, *node_ids: Optional[list] = None*, *expire_recent_drugs: Optional[emod_api.interventions.common.PropertyValueChanger] = None*, *ind_property_restrictions: Optional[list] = None*, *disqualifying_properties: Optional[list] = None*)

Add an rfMDA (reactive focal mass drug administration) drug intervention to your campaign. See *add_drug_campaign()* for more information about each argument.

Returns None

emodpy_malaria.interventions.drug_campaign.**fmda_cfg**(*campaign*, *fmda_type: any = 0*, *node_selection_type: str = 'DISTANCE_ONLY'*, *event_trigger: str = 'Give_Drugs'*)

Create an fMDA (focal mass drug administration) configuration.

Parameters

- **fmda_type** – The setting for **Max_Distance_To_Other_Nodes_Km** in BroadcastEventToOtherNodes.
- **node_selection_type** – The setting for **Node_Selection_Type** in BroadcastEventToOtherNodes.
- **event_trigger** – The setting for **Event_Trigger** in BroadcastEventToOtherNodes.

Returns Configured BroadcastEventToOtherNodes intervention.

emodpy_malaria.interventions.inputeir module

emodpy_malaria.interventions.inputeir.**add_scheduled_input_eir**(*campaign*, *start_day: int = 1*, *node_ids: Optional[list] = None*, *monthly_eir: Optional[list] = None*, *daily_eir: Optional[list] = None*, *age_dependence: str = 'OFF'*, *scaling_factor: float = 1.0*, *intervention_name: str = 'InputEIR'*)

Create a full CampaignEvent that distributes InputEIR to a population.

Parameters

- **campaign** – Passed in campaign (from emod_api.campaign)
- **start_day** – The day on which the monthly_eir cycle starts
- **node_ids** – List of nodes to which to distribute the intervention. [] or None, indicates all nodes will get the intervention
- **monthly_eir** – An array of 12 elements that contain an entomological inoculation rate (EIR) for each month; Each value should be between 0 and 1000
- **daily_eir** – An array of 365 values where each value is the mean number of infectious bites experienced by an individual for that day of the year
- **start_day** – The day on which the monthly_eir cycle starts
- **age_dependence** – Determines how InputEIR depends on the age of the target. Options are "OFF", "LINEAR", "SURFACE_AREA_DEPENDENT"
- **scaling_factor** – A modifier that is multiplied by the EIR determined for the current day
- **intervention_name** – The optional name used to refer to this intervention as a means to differentiate it from others that use the same class. It's possible to have multiple InputEIR interventions attached to a node if they have different Intervention_Name values.

Returns Nothing

emodpy_malaria.interventions.inputeir.**new_intervention_as_file**(*campaign*, *start_day: int = 0*, *monthly_eir: Optional[list] = None*, *daily_eir: Optional[list] = None*, *filename: str = 'InputEIR.json'*)

Create an InputEIR intervention as its own file.

Parameters

- **campaign** – Passed in campaign (from emod_api.campaign)
- **start_day** – The day on which the monthly_eir cycle starts
- **monthly_eir** – An array of 12 elements that contain an entomological inoculation rate (EIR) for each month; Each value should be between 0 and 1000
- **daily_eir** – An array of 365 values where each value is the mean number of infectious bites experienced by an individual for that day of the year
- **filename** – filename used for the file created

Returns The filename of the file created

emodpy_malaria.interventions.irs module

emodpy_malaria.interventions.irs.**add_scheduled_irs_housing_modification**(*campaign*, *start_day: int = 1*, *demographic_coverage: float = 1*, *node_ids: Optional[list] = None*, *killing_initial_effect: float = 1*, *killing_box_duration: int = 0*, *killing_decay_time_constant: int = 90*, *repelling_initial_effect: float = 0*, *repelling_box_duration: int = 0*, *repelling_decay_time_constant: int = 90*, *insecticide: str = ''*, *intervention_name: str = 'IRSHousingModification'*)

Adds scheduled IRSHousingModification intervention to the campaign. The IRSHousingModification intervention class includes Indoor Residual Spraying (IRS) in the simulation. IRS is another key vector control tool in which insecticide is sprayed on the interior walls of a house so that mosquitoes resting on the walls after consuming a blood meal will die. IRS can also have a repellent effect. Because this class is distributed to individuals, it can target subgroups of the population. To target all individuals in a node, use IndoorSpaceSpraying. Do not use IRSHousingModification and IndoorSpaceSpraying together.

Parameters

- **campaign** – A campaign builder that also contains schema_path parameters
- **start_day** – The day on which the intervention is distributed
- **demographic_coverage** – The fraction of individuals in the target demographic that will receive this intervention
- **node_ids** – A list of node ids to which this intervention will be distributed. None or [] distributes intervention to all nodes
- **killing_initial_effect** – Initial strength of the Killing effect. The effect may decay over time.
- **killing_box_duration** – Box duration of effect in days before the decay of Killing Initial_Effect.
- **killing_decay_time_constant** – The exponential decay length, in days of the Killing Initial_Effect.
- **repelling_initial_effect** – Initial strength of the Killing effect. The effect decays over time.

- **repelling_box_duration** – Box duration of effect in days before the decay of Repelling Initial Effect.
- **repelling_decay_time_constant** – The exponential decay length, in days of the Repelling Initial Effect.
- **insecticide** – The name of the insecticide defined in config.Insecticides for this intervention. If insecticides are being used, then this must be defined as one of those values. If they are not being used, then this does not needed to be specified or can be empty string. It cannot have a value if config.Insecticides does not define anything.
- **intervention_name** – The optional name used to refer to this intervention as a means to differentiate it from others that use the same class. It's possible to have multiple IRSHousingModification interventions attached to a person if they have different Intervention_Name values.

Returns Nothing

emodpy_malaria.interventions.irs.**add_triggered_irs_housing_modification**(*campaign*, *start_day: int = 1*, *demographic_coverage: float = 1*, *node_ids: Optional[list] = None*, *trigger_condition_list: Optional[list] = None*, *listening_duration: int = - 1*, *delay_period_constant: float = 0*, *killing_initial_effect: float = 1*, *killing_box_duration: int = 0*, *killing_decay_time_constant: int = 90*, *repelling_initial_effect: float = 0*, *repelling_box_duration: int = 0*, *repelling_decay_time_constant: int = 90*, *insecticide: str = ''*, *intervention_name: str = 'IRSHousingModification'*)

Adds triggered IRSHousingModification intervention to the campaign. The IRSHousingModification intervention class includes Indoor Residual Spraying (IRS) in the simulation. IRS is another key vector control tool in which insecticide is sprayed on the interior walls of a house so that mosquitoes resting on the walls after consuming a blood meal will die. IRS can also have a repellent effect. Because this class is distributed to individuals, it can target subgroups of the population. To target all individuals in a node, use IndoorSpaceSpraying. Do not use IRSHousingModification and IndoorSpaceSpraying together.

Parameters

- **campaign** – A campaign builder that also contains schema_path parameters
- **start_day** – The day on which the intervention is distributed
- **demographic_coverage** – The fraction of individuals in the target demographic that will receive this intervention
- **node_ids** – A list of node ids to which this intervention will be distributed. None or [] distributes intervention to all nodes
- **trigger_condition_list** – A list of the events that will trigger intervention distribution.
- **listening_duration** – The number of time steps that the distributed event will monitor for triggers. Default is -1, which is indefinitely.
- **delay_period_constant** – Optional. Delay, in days, before the intervention is given out after a trigger is received.
- **killing_initial_effect** – Initial strength of the Killing effect. The effect may decay over time.
- **killing_box_duration** – Box duration of effect in days before the decay of Killing Initial_Effect.
- **killing_decay_time_constant** – The exponential decay length, in days of the Killing Initial_Effect.
- **repelling_initial_effect** – Initial strength of the Killing effect. The effect decays over time.
- **repelling_box_duration** – Box duration of effect in days before the decay of Repelling Initial Effect.
- **repelling_decay_time_constant** – The exponential decay length, in days of the Repelling Initial Effect.
- **insecticide** – The name of the insecticide defined in config.Insecticides for this intervention. If insecticides are being used, then this must be defined as one of those values. If they are not being used, then this does not needed to be specified or can be empty string. It cannot have a value if config.Insecticides does not define anything.
- **intervention_name** – The optional name used to refer to this intervention as a means to differentiate it from others that use the same class. It's possible to have multiple IRSHousingModification interventions attached to a person if they have different Intervention_Name values.

Returns Nothing

emodpy_malaria.interventions.irs.**irs_configuration**(*campaign*, *killing_initial_effect: float = 1*, *killing_box_duration: int = 0*, *killing_decay_time_constant: int = 90*, *repelling_initial_effect: float = 0*, *repelling_box_duration: int = 0*, *repelling_decay_time_constant: int = 90*, *insecticide: str = ''*, *intervention_name: str = 'IRSHousingModification'*)

Configures and returns IRSHousingModification intervention. The IRSHousingModification intervention class includes Indoor Residual Spraying (IRS) in the simulation. IRS is another key vector control tool in which insecticide is sprayed on the interior walls of a house so that mosquitoes resting on the walls after consuming a blood meal will die. IRS can also have a repellent effect. Because this

class is distributed to individuals, it can target subgroups of the population. To target all individuals in a node, use IndoorSpaceSpraying. Do not use IRSHousingModification and IndoorSpaceSpraying together.

Parameters

- **campaign** – A campaign builder that also contains schema_path parameters
- **killing_initial_effect** – Initial strength of the Killing effect. The effect may decay over time.
- **killing_box_duration** – Box duration of effect in days before the decay of Killing Initial_Effect.
- **killing_decay_time_constant** – The exponential decay length, in days of the Killing Initial_Effect.
- **repelling_initial_effect** – Initial strength of the Killing effect. The effect decays over time.
- **repelling_box_duration** – Box duration of effect in days before the decay of Repelling Initial Effect.
- **repelling_decay_time_constant** – The exponential decay length, in days of the Repelling Initial Effect.
- **insecticide** – The name of the insecticide defined in config.Insecticides for this intervention. If insecticides are being used, then this must be defined as one of those values. If they are not being used, then this does not needed to be specified or can be empty string. It cannot have a value if config.Insecticides does not define anything.
- **intervention_name** – The optional name used to refer to this intervention as a means to differentiate it from others that use the same class. It's possible to have multiple IRSHousingModification interventions attached to a person if they have different Intervention_Name values.

Returns Configured IRSHousingModification intervention

emodpy_malaria.interventions.irs.**new_intervention_as_file**(*campaign*, *start_day*, *filename=None*)

emodpy_malaria.interventions.ivermectin module

emodpy_malaria.interventions.ivermectin.**add_scheduled_ivermectin**(*campaign*, *start_day: int = 1*, *demographic_coverage: float = 1.0*, *target_num_individuals: Optional[int] = None*, *node_ids: Optional[list] = None*, *repetitions: int = 1*, *timesteps_between_repetitions: int = 365*, *ind_property_restrictions: Optional[list] = None*, *killing_initial_effect: float = 1*, *killing_box_duration: float = 0*, *killing_decay_time_constant: float = 0*, *insecticide: str = ''*, *cost: float = 1*, *intervention_name: str = 'Ivermectin'*, *broadcast_event: str = 'Received_Ivermectin'*)

Adds a scheduled Ivermectin CampaignEvent to the campaign, which can be repeated any number of times. It's possible to have multiple Ivermectin interventions attached to a person if they have different Intervention_Name values.

Note: for WaningEffect, box_duration = 0 + decay_time_constant > 0 => WaningEffectExponential box_duration > 0 + decay_time_constant = 0 => WaningEffectBox/Constant (depending on duration) box_duration > 0 + decay_time_constant > 0 => WaningEffectBoxExponential

Parameters

- **campaign** – A campaign builder that also contains schema_path parameters
- **start_day** – The day on which the intervention is distributed
- **demographic_coverage** – probability of choosing an individual, is ignored if "target_num_individuals" is set
- **target_num_individuals** – number of individuals to receive ivermectin, demographic_coverage will be ignored if this is set
- **node_ids** – The list of nodes to apply this intervention to (**Node_List** parameter). If not provided, intervention is distributed to all nodes.
- **repetitions** – The number of times an intervention is given, used with timesteps_between_repetitions. -1 means the intervention repeats forever. Sets **Number_Repetitions**
- **timesteps_between_repetitions** – The interval, in timesteps, between repetitions. Ignored if repetitions = 1. Sets **Timesteps_Between_Repetitions**
- **ind_property_restrictions** – A list of dictionaries of IndividualProperties, which are needed for the individual to receive the intervention. Sets the **Property_Restrictions_Within_Node**
- **killing_initial_effect** – Initial strength of the Killing effect. The effect may decay over time.

- **killing_box_duration** – Box duration of effect in days before the decay of Killing Initial_Effect.
- **killing_decay_time_constant** – The exponential decay length, in days of the Killing Initial_Effect.
- **insecticide** – The name of the insecticide defined in config.Insecticides for this intervention. If insecticides are being used, then this must be defined as one of those values. If they are not being used, then this does not needed to be specified or can be empty string. It cannot have a value if config.Insecticides does not define anything.
- **cost** – Unit cost per Ivermectin dosing (unamortized)
- **intervention_name** – The optional name used to refer to this intervention as a means to differentiate it from others that use the same class. It's possible to have multiple Ivermectin interventions attached to a person if they have different Intervention_Name values.
- **broadcast_event** – An event to be broadcast when a person receives Ivermectin intervention. Default: "Received_Ivermectin", you can turn this off by passing in an empty string or None

Returns Nothing

emodpy_malaria.interventions.ivermectin.**add_triggered_ivermectin**(*campaign*, *start_day: int = 1*, *trigger_condition_list: Optional[list] = None*, *listening_duration: int = -1*, *delay_period_constant: float = 0*, *demographic_coverage: float = 1.0*, *node_ids: Optional[list] = None*, *ind_property_restrictions: Optional[list] = None*, *killing_initial_effect: float = 1*, *killing_box_duration: float = 0*, *killing_decay_time_constant: float = 0*, *insecticide: str = ''*, *cost: float = 1*, *intervention_name: str = 'Ivermectin'*, *broadcast_event: str = 'Received_Ivermectin'*)

Adds a triggered Ivermectin CampaignEvent to the campaign, that responds to a trigger after an optional delay. The intervention is distributed on start_day and responds to triggers for a listening_duration of days.

It's possible to have multiple Ivermectin interventions attached to a person if they have different Intervention_Name values.

Note: for WaningEffect, box_duration = 0 + decay_time_constant > 0 => WaningEffectExponential box_duration > 0 + decay_time_constant = 0 => WaningEffectBox/Constant (depending on duration) box_duration > 0 + decay_time_constant > 0 => WaningEffectBoxExponential

Parameters

- **campaign** – campaign object to which the intervention will be added, and schema_path container
- **start_day** – The day the intervention is given out.

- **trigger_condition_list** – A list of the events that will trigger intervention distribution.
- **listening_duration** – The number of time steps that the distributed event will monitor for triggers. Default is -1, which is indefinitely.
- **delay_period_constant** – Optional. Delay, in days, before the intervention is given out after a trigger is received.
- **demographic_coverage** – This value is the probability that each individual in the target population will receive the intervention. It does not guarantee that the exact fraction of the target population set by Demographic_Coverage receives the intervention.
- **node_ids** – List of nodes to which to distribute the intervention. [] or None, indicates all nodes will get the intervention
- **ind_property_restrictions** – A list of dictionaries of IndividualProperties, which are needed for the individual to receive the intervention. Sets the **Property_Restrictions_Within_Node**
- **killing_initial_effect** – Initial strength of the Killing effect. The effect may decay over time.
- **killing_box_duration** – Box duration of effect in days before the decay of Killing Initial_Effect.
- **killing_decay_time_constant** – The exponential decay length, in days of the Killing Initial_Effect.
- **insecticide** – The name of the insecticide defined in config.Insecticides for this intervention. If insecticides are being used, then this must be defined as one of those values. If they are not being used, then this does not needed to be specified or can be empty string. It cannot have a value if config.Insecticides does not define anything.
- **cost** – Unit cost per Ivermectin dosing (unamortized)
- **intervention_name** – The optional name used to refer to this intervention as a means to differentiate it from others that use the same class. It's possible to have multiple Ivermectin interventions attached to a person if they have different Intervention_Name values.
- **broadcast_event** – An event to be broadcast when a person receives Ivermectin intervention. Default: "Received_Ivermectin", you can turn this off by passing in an empty string or None

Returns Nothing

emodpy_malaria.interventions.larvicide module

emodpy_malaria.interventions.larvicide.**add_larvicide**(*campaign*, *start_day: int = 1*, *num_repetitions: int = - 1*, *timesteps_between_reps: int = 365*, *spray_coverage: float = 1.0*, *killing_effect: float = 1*, *habitat_target: str = 'ALL_HABITATS'*, *insecticide: Optional[str] = None*, *box_duration: int = 100*, *decay_time_constant: float = 0.0*, *node_ids: Optional[list] = None*)

Create a new Larvicides scheduled campaign intervention & add to campaign. box_duration = 0 + decay_time_constant > 0 => WaningEffectExponential box_duration > 0 + decay_time_constant = 0 => WaningEffectBox/Constant (depending on duration) box_duration > 0 + decay_time_constant > 0 => WaningEffectBoxExponential

Inspired by: https://github.com/InstituteforDiseaseModeling/dtk-tools/blob/master/dtk/interventions/novel_vector_control.py#L279.

Parameters

- **campaign** –
- **start_day** – the day to distribute the Larvicides intervention.
- **num_repetitions** – Optional number of repetitions.
- **timesteps_between_reps** – Gap between repetitions, if num_reptitions specified.
- **spray_coverage** – how much of each node to cover (total portion killed = killing effect * coverage).
- **killing_effect** – portion of vectors killed by the intervention (Initial_Effect in WaningEffect).
- **habitat_target** – Possible values are: "TEMPORARY_RAINFALL", "WATER_VEGETATION", "HUMAN_POPULATION", "CONSTANT", "BRACKISH_SWAMP", "LINEAR_SPLINE", "ALL_HABITATS". The latter is the default.
- **insecticide** – insecticide name. Must be a value in the config but consistency is not checked at this time.
- **box_duration** – Box_Duration of the WaningEffect.
- **decay_time_constant** – decay_time_constant of the WaningEffect.
- **node_ids** – list of node ids to which distribute the intervention.

Returns N/A.

emodpy_malaria.interventions.larvicide.**new_intervention_as_file**(*campaign*, *start_day: int = 1*, *filename: Optional[str] = None*)

Creates a file with SpaceSpray intervention :param campaign: :param start_day: the day to distribute the Larvicides intervention :param filename: name of the filename created

Returns: filename of the file created

emodpy_malaria.interventions.mosquitorelease module

emodpy_malaria.interventions.mosquitorelease.**add_scheduled_mosquito_release**(*campaign*, *start_day: int = 0*, *node_ids: Optional[list] = None*, *repetitions: int = 1*, *timesteps_between_repetitions: int = 365*, *intervention_name: str = 'MosquitoRelease'*, *released_number: Optional[int] = None*, *released_fraction: Optional[float] = None*, *released_infectious: float = 0*, *released_species: str = 'arabiensis'*, *released_genome: Optional[list] = None*)

Adds to the campaign a node-level MosquitoRelease intervention

Parameters

- **campaign** – A campaign builder that also contains schema_path parameters
- **start_day** – The day to release the vectors.
- **node_ids** – List of nodes to which to distribute the intervention. [] or None, indicates all nodes will get the intervention
- **repetitions** – The number of times an intervention is given, used with timesteps_between_repetitions. -1 means the intervention repeats forever. Sets **Number_Repetitions**
- **timesteps_between_repetitions** – The interval, in timesteps, between repetitions. Ignored if repetitions = 1. Sets **Timesteps_Between_Repetitions**
- **intervention_name** – The optional name used to refer to this intervention as a means to differentiate it from others that use the same class. It's possible to have multiple MosquitoRelease interventions if they have different Intervention_Name values.
- **released_number** – The number of vectors to release, sets Released_Type = "FIXED_NUMBER"
- **released_fraction** – The fraction of the current population of mosquitoes to release. The 'population' will depend on the gender of the mosquitoes being released and it will be the

population from the previous time step. Sets Released_Type = "FRACTION"

- **released_infectious** – The fraction of vectors released that are to be infectious. One can only use this feature when 'Malaria_Model'!='MALARIA_MECHANISTIC_MODEL_WITH_PARASITE_GENETICS'
- **released_species** – The case sensitive name of the species of vectors to be released.
- **released_genome** – This defines the alleles of the genome of the vectors to be released. It must define all of the alleles including the gender 'gene'. '*' is not allowed.

Returns Formatted intervention

emodpy_malaria.interventions.mosquitorelease.**new_intervention_as_file**(*campaign*, *start_day: int = 1*, *filename: str = 'MosquitoRelease.json'*)

Creates a campaign file with a MosquitoRelease intervention

Parameters

- **campaign** – A campaign builder that also contains schema_path parameters
- **start_day** – The day to release the vectors.
- **filename** – name of campaign filename to be created

Returns returns filename

emodpy_malaria.interventions.outbreak module

emodpy_malaria.interventions.outbreak.**add_outbreak_individual**(*campaign*, *start_day: int = 1*, *demographic_coverage: float = 1.0*, *target_num_individuals: Optional[int] = None*, *node_ids: Optional[list] = None*, *repetitions: int = 1*, *timesteps_between_repetitions: int = 365*, *ind_property_restrictions: Optional[list] = None*, *target_age_min: int = 0*, *target_age_max: int = 125*, *target_gender: str = 'All'*, *ignore_immunity: bool = True*, *incubation_period_override: int = -1*, *antigen: int = 0*, *genome: int = 0*, *broadcast_event: Optional[str] = None*)

Adds a scheduled OutbreakIndividual intervention. This is set up to be used with Malaria-Ongoing branch.

Parameters

- **campaign** – campaign object to which the intervention will be added, and schema_path container
- **start_day** – The day the intervention is given out.

- **demographic_coverage** – This value is the probability that each individual in the target population will receive the intervention. It does not guarantee that the exact fraction of the target population set by Demographic_Coverage receives the intervention.
- **target_num_individuals** – The exact number of people to select out of the targeted group. If this value is set, demographic_coverage parameter is ignored
- **node_ids** – List of nodes to which to distribute the intervention. [] or None, indicates all nodes will get the intervention
- **repetitions** – The number of times an intervention is given, used with timesteps_between_repetitions. -1 means the intervention repeats forever. Sets **Number_Repetitions**
- **timesteps_between_repetitions** – The interval, in timesteps, between repetitions. Ignored if repetitions = 1. Sets **Timesteps_Between_Repetitions**
- **ind_property_restrictions** – A list of dictionaries of IndividualProperties, which are needed for the individual to receive the intervention. Sets the **Property_Restrictions_Within_Node**
- **target_age_min** – The lower end of ages targeted for an intervention, in years. Sets **Target_Age_Min**
- **target_age_max** – The upper end of ages targeted for an intervention, in years. Sets **Target_Age_Max**
- **target_gender** – The gender targeted for an intervention: All, Male, or Female.
- **ignore_immunity** – Individuals will be force-infected (with a specific strain) regardless of actual immunity level when set to True (1). Default is True (1). The person will or will not get an infection based on their immunity level if this is set to False.
- **incubation_period_override** – The incubation period, in days, that infected individuals will go through before becoming infectious. This value overrides the incubation period set in the configuration file. Set to -1 to honor the configuration parameter settings
- **antigen** – The antigenic base strain ID of the outbreak infection
- **genome** – The genetic substrain ID of the outbreak infection
- **broadcast_event** – Optional event that will be sent out at the same time as outbreak is distributed

Returns Nothing

emodpy_malaria.interventions.outbreak.**add_outbreak_malaria_genetics**(*campaign*, *start_day: int = 1*, *demographic_coverage: float = 1.0*, *target_num_individuals: Optional[int] = None*, *node_ids: Optional[list] = None*, *repetitions: int = 1*, *timesteps_between_repetitions: int = 365*, *ind_property_restrictions: Optional[list] = None*, *target_age_min: int = 0*, *target_age_max: int = 125*, *target_gender: str = 'All'*, *ignore_immunity: bool = True*, *incubation_period_override: int = -1*, *create_nucleotide_sequence_from: str = 'BARCODE_STRING'*, *barcode_string: Optional[str] = None*, *drug_resistant_string: Optional[str] = None*, *msp_variant_value: Optional[int] = None*, *pfemp1_variants_values: Optional[list] = None*, *barcode_allele_frequencies_per_genome_locatio Optional[list] = None*, *drug_resistant_allele_frequencies_per_genom Optional[list] = None*)

Creates a scheduled OutbreakIndividualMalariaGenetics CampaignEvent which can then be added to a campaign.

Parameters

- **campaign** – campaign object to which the intervention will be added, and schema_path container
- **start_day** – The day the intervention is given out.
- **demographic_coverage** – This value is the probability that each individual in the target population will receive the intervention. It does not guarantee that the exact fraction of the target population set by Demographic_Coverage receives the intervention.
- **target_num_individuals** – The exact number of people to select out of the targeted group. If this value is set, demographic_coverage parameter is ignored.
- **node_ids** – List of nodes to which to distribute the intervention. [] or None, indicates all nodes will get the intervention
- **repetitions** – The number of times an intervention is given, used with timesteps_between_repetitions. -1 means the intervention repeats forever. Sets **Number_Repetitions**

- **timesteps_between_repetitions** – The interval, in timesteps, between repetitions. Ignored if repetitions = 1. Sets **Timesteps_Between_Repetitions**
- **ind_property_restrictions** – A list of dictionaries of IndividualProperties, which are needed for the individual to receive the intervention. Sets the **Property_Restrictions_Within_Node**
- **target_age_min** – The lower end of ages targeted for an intervention, in years. Sets **Target_Age_Min**
- **target_age_max** – The upper end of ages targeted for an intervention, in years. Sets **Target_Age_Max**
- **target_gender** – The gender targeted for an intervention: All, Male, or Female.
- **ignore_immunity** – Individuals will be force-infected (with a specific strain) regardless of actual immunity level when set to True (1). Default is True (1). The person will or will not get an infection based on their immunity level if this is set to False.
- **incubation_period_override** – The incubation period, in days, that infected individuals will go through before becoming infectious. This value overrides the incubation period set in the configuration file. Set to -1 to honor the configuration parameter settings
- **create_nucleotide_sequence_from** – A string that indicates how the genomes are created. Possible values are: BARCODE_STRING, ALLELE_FREQUENCIES, NUCLEOTIDE_SEQUENCE.
- **barcode_string** – A series of nucleotide base letters (A, C, G, T) that represent the values at locations in the genome. The length of the string depends on the number of locations defined in config.Parasite_Genetics.Barcode_Genome_Locations. Each character of the string corresponds to one of the locations. The locations are assumed to be in ascending order. Also depends on create_nucleotide_sequence_from when it is equal to NUCLEOTIDE_SEQUENCE or BARCODE_STRING.
- **drug_resistant_string** – A series of nucleotide base letters (A, C, G, T) that represent the values at locations in the genome. The length of the string depends on the number of locations defined in config.Parasite_Genetics.Drug_Resistant_Genome_Locations. Each character of the string corresponds to one of the locations. The locations are assumed to be in ascending order. Also depends on create_nucleotide_sequence_from when it is equal to NUCLEOTIDE_SEQUENCE or BARCODE_STRING.
- **msp_variant_value** – The Merozoite Surface Protein value used to determine how the antibodies recognizes the merzoites. This value depends on config.Falciparum_MSP_Variants and must be less than or equal to it. It also depends on create_nucleotide_sequence_from when it is equal to NUCLEOTIDE_SEQUENCE.
- **pfemp1_variants_values** – The PfEMP1 Variant values / major epitopes used to define how the antibodies recognize the infected red blood cells. The values of the array depend on config.Falciparum_PfEMP1_Variants and must be less than or equal to it. There must be exactly 50 values – one for each epitope. It also depends on create_nucleotide_sequence_from when it is equal to NUCLEOTIDE_SEQUENCE.
- **barcode_allele_frequencies_per_genome_location** – The fractions of allele occurrences for each location in the barcode. This 2D array should have one array for each location/character in the barcode. For each location, there should be four values between 0 and 1 indicating the probability that specific character appears. The possible letters are: A=0, C=1, G=2, T=3. It also depends on create_nucleotide_sequence_from when it is equal to ALLELE_FREQUENCIES. The frequencies should sum up to 1.

- **drug_resistant_allele_frequencies_per_genome_location** – The fractions of allele occurrences for each location in the drug resistant markers. This 2D array should have one array for each drug resistant location. For each location, there should be four values between 0 and 1 indicating the probability that specific character will appear. The possible letters are'A'=0, 'C'=1, 'G'=2, 'T'=3. It also depends on create_nucleotide_sequence_from when it is equal to ALLELE_FREQUENCIES. The frequencies should sum up to 1.

Returns CampaignEvent which then can be added to the campaign file

emodpy_malaria.interventions.outbreak.**add_outbreak_malaria_var_genes**(*campaign*, *start_day: int = 1*, *demographic_coverage: float = 1.0*, *target_num_individuals: Optional[int] = None*, *node_ids: Optional[list] = None*, *repetitions: int = 1*, *timesteps_between_repetitions: int = 365*, *ind_property_restrictions: Optional[list] = None*, *target_age_min: int = 0*, *target_age_max: int = 125*, *target_gender: str = 'All'*, *ignore_immunity: bool = True*, *incubation_period_override: int = - 1*, *irbc_type: Optional[list] = None*, *minor_epitope_type: Optional[list] = None*, *msp_type: Optional[int] = None*)

Creates a scheduled OutbreakIndividualMalariaGenetics CampaignEvent which can then be added to a campaign.

Parameters

- **campaign** – campaign object to which the intervention will be added, and schema_path container
- **start_day** – The day the intervention is given out.
- **demographic_coverage** – This value is the probability that each individual in the target population will receive the intervention. It does not guarantee that the exact fraction of the target population set by Demographic_Coverage receives the intervention.
- **target_num_individuals** – The exact number of people to select out of the targeted group. If this value is set, demographic_coverage parameter is ignored
- **node_ids** – List of nodes to which to distribute the intervention. [] or None, indicates all nodes will get the intervention
- **repetitions** – The number of times an intervention is given, used with timesteps_between_repetitions. -1 means the intervention repeats forever. Sets **Number_Repetitions**

- **timesteps_between_repetitions** – The interval, in timesteps, between repetitions. Ignored if repetitions = 1. Sets **Timesteps_Between_Repetitions**
- **ind_property_restrictions** – A list of dictionaries of IndividualProperties, which are needed for the individual to receive the intervention. Sets the **Property_Restrictions_Within_Node**
- **target_age_min** – The lower end of ages targeted for an intervention, in years. Sets **Target_Age_Min**
- **target_age_max** – The upper end of ages targeted for an intervention, in years. Sets **Target_Age_Max**
- **target_gender** – The gender targeted for an intervention: All, Male, or Female.
- **ignore_immunity** – Individuals will be force-infected (with a specific strain) regardless of actual immunity level when set to True (1). Default is True (1). The person will or will not get an infection based on their immunity level if this is set to False.
- **incubation_period_override** – The incubation period, in days, that infected individuals will go through before becoming infectious. This value overrides the incubation period set in the configuration file. Set to -1 to honor the configuration parameter settings
- **irbc_type** – The array PfEMP1 Major epitope variant values. There must be exactly 50 values. Min value = 0, MAX value = config.Falciparum_PfEMP1_Variants.
- **minor_epitope_type** – The array PfEMP1 Minor epitope variant values. There must be exactly 50 values. Min value = 0, MAX value = config.Falciparum_Nonspecific_Types * MINOR_EPITOPE_VARS_PER_SET(=5) .
- **msp_type** – The Merozoite Surface Protein variant value of this infection. Min value = 0, MAX value = config.Falciparum_MSP_Variants.

Returns CampaignEvent which then can be added to the campaign file

emodpy_malaria.interventions.outbreak.**add_campaign_event**(*campaign*, *start_day: int = 1*, *demographic_coverage: float = 1.0*, *target_num_individuals: Optional[int] = None*, *node_ids: Optional[list] = None*, *repetitions: int = 1*, *timesteps_between_repetitions: int = 365*, *ind_property_restrictions: Optional[list] = None*, *target_age_min: int = 0*, *target_age_max: int = 125*, *target_gender: str = 'All'*, *intervention: Optional[any] = None*)

Adds a campaign event to the campaign with a passed in intervention.

Parameters

- **campaign** – campaign object to which the intervention will be added, and schema_path container
- **start_day** – The day the intervention is given out.
- **demographic_coverage** – This value is the probability that each individual in the target population will receive the intervention. It does not guarantee that the exact fraction of the target population set by Demographic_Coverage receives the intervention.

- **target_num_individuals** – The exact number of people to select out of the targeted group. If this value is set, demographic_coverage parameter is ignored
- **node_ids** – List of nodes to which to distribute the intervention. [] or None, indicates all nodes will get the intervention
- **repetitions** – The number of times an intervention is given, used with timesteps_between_repetitions. -1 means the intervention repeats forever. Sets **Number_Repetitions**
- **timesteps_between_repetitions** – The interval, in timesteps, between repetitions. Ignored if repetitions = 1. Sets **Timesteps_Between_Repetitions**
- **ind_property_restrictions** – A list of dictionaries of IndividualProperties, which are needed for the individual to receive the intervention. Sets the **Property_Restrictions_Within_Node**
- **target_age_min** – The lower end of ages targeted for an intervention, in years. Sets **Target_Age_Min**
- **target_age_max** – The upper end of ages targeted for an intervention, in years. Sets **Target_Age_Max**
- **target_gender** – The gender targeted for an intervention: All, Male, or Female.
- **intervention** – Intervention or a list of interventions to be distributed by this event

Returns:

emodpy_malaria.interventions.outdoorrestkill module

emodpy_malaria.interventions.outdoorrestkill.**add_outdoorrestkill**(*campaign*, *start_day: int = 1*, *node_ids: Optional[list] = None*, *insecticide: Optional[str] = None*, *killing_initial_effect: float = 1*, *killing_box_duration: int = - 1*, *killing_decay_time_constant: float = 0*)

Adds a node-targeted OutdoorRestKill intervention to the campaign

Parameters

- **campaign** – campaign object to which the intervention will be added, and schema_path container
- **start_day** – the day on which to distribute the intervention
- **insecticide** – The name of the insecticide defined in config.Insecticides for this intervention. If insecticides are being used, then this must be defined as one of those values. If they are not being used, then this does not needed to be specified or can be empty string. It cannot have a value if config.Insecticides does not define anything.
- **killing_initial_effect** – **Initial_Effect** in the *Killing_Config**
- **killing_box_duration** – Length in days before the **Initial_Effect** starts to decay, -1 indicates forever.
- **killing_decay_time_constant** – The rate of decay of the *Initial_Effect**

- **node_ids** – List of nodes to which to distribute the intervention. None or empty list implies "all nodes".

Returns configured campaign object

emodpy_malaria.interventions.scale_larval_habitats module

emodpy_malaria.interventions.scale_larval_habitats.**add_scale_larval_habitats**(*campaign*, *df=None*, *start_day: int = 0*, *repetitions: int = 1*, *timesteps_between_repetitions: int = 365*)

Reduce available larval habitat in a node-specific way.

Parameters

- **campaign** – campaign object to which the intervention will be added, and schema_path container
- **df** – The dataframe containing habitat scale factors. **Examples**:

```
Scale TEMPORARY_RAINFALL by 3-fold for all nodes, all species:
df = pd.DataFrame({ 'TEMPORARY_RAINFALL': [3]})

Scale TEMPORARY_RAINFALL by 3-fold for all nodes, arabiensis only:
df = pd.DataFrame({ 'TEMPORARY_RAINFALL.arabiensis': [3]})

Scale differently by node ID:
df = pd.DataFrame({ 'NodeID' : [0, 1, 2, 3, 4],
                    'CONSTANT': [1, 0, 1, 1, 1],
                    'TEMPORARY_RAINFALL': [1, 1, 0, 1, 0]})

Scale differently by both node ID and species:
df = pd.DataFrame({ 'NodeID' : [0, 1, 2, 3, 4],
                    'CONSTANT.arabiensis': [1, 0, 1, 1, 1],
                    'TEMPORARY_RAINFALL.arabiensis': [1, 1, 0, 1, 0],
                    'CONSTANT.funestus': [1, 0, 1, 1, 1]})

Scale some habitats by species and others same for all species:
df = pd.DataFrame({ 'NodeID' : [0, 1, 2, 3, 4],
                    'CONSTANT.arabiensis': [1, 0, 1, 1, 1],
                    'TEMPORARY_RAINFALL.arabiensis': [1, 1, 0, 1, 0],
                    'CONSTANT.funestus': [1, 0, 1, 1, 1],
                    'LINEAR_SPLINE': [1, 1, 0, 1, 0]})

Scale nodes at different dates:
df = pd.DataFrame({  'NodeID' : [0, 1, 2, 3, 4],
                     'CONSTANT': [1, 0, 1, 1, 1],
                     'TEMPORARY_RAINFALL': [1, 1, 0, 1, 0],
                     'Start_Day': [0, 30, 60, 65, 65]
                     })
```

- **start_day** – The date that habitats are scaled for all scaling actions specified in **df**. Used only if there is no Start_Day column in **df**.
- **repetitions** – The number of times to repeat the intervention.
- **timesteps_between_repetitions** – The number of time steps between repetitions.

Returns None

emodpy_malaria.interventions.scale_larval_habitats.**add_habitat_reduction_event**(*campaign*, *start_day: int*, *node_ids: list*, *habitat_scales: list*, *repetitions: int*, *timesteps_between_repetitions: int*)

Add a campaign event to reduce vector's larval habitat(s).

Parameters

- **campaign** – campaign object to which the intervention will be added, and schema_path container
- **start_day** – The day the intervention is given out.
- **node_ids** – List of nodes to which to distribute the intervention. [] or None, indicates all nodes will get the intervention
- **habitat_scales** – List of dictionaries for scaling larval habitats. Examples:

```
[{"Habitat": "ALL_HABITATS", "Species": "ALL_SPECIES", "Factor": 0.5}
↪,
{"Habitat": "CONSTANT", "Species": "arabiensis", "Factor": 2}]
```

- **repetitions** – The number of times an intervention is given, used with timesteps_between_repetitions. -1 means the intervention repeats forever. Sets **Number_Repetitions**
- **timesteps_between_repetitions** – The interval, in timesteps, between repetitions. Ignored if repetitions = 1. Sets **Timesteps_Between_Repetitions**

Returns Nothing

emodpy_malaria.interventions.spacespraying module

emodpy_malaria.interventions.spacespraying.**add_scheduled_space_spraying**(*campaign*, *start_day: int = 1*, *node_ids: Optional[list] = None*, *repetitions: int = 1*, *timesteps_between_repetitions: int = 365*, *spray_coverage: float = 1.0*, *insecticide: str = ''*, *killing_initial_effect: float = 1*, *killing_box_duration: float = - 1*, *killing_decay_time_constant: float = 0*, *intervention_name: str = 'SpaceSpraying'*, *cost_to_consumer: float = 0*)

Parameters

- **campaign** – campaign object to which the intervention will be added, and schema_path container
- **start_day** – The day the intervention is given out.
- **node_ids** – List of nodes to which to distribute the intervention. [] or None, indicates all nodes will get the intervention
- **repetitions** – The number of times an intervention is given, used with timesteps_between_repetitions. -1 means the intervention repeats forever. Sets **Number_Repetitions**
- **timesteps_between_repetitions** – The interval, in timesteps, between repetitions. Ignored if repetitions = 1. Sets **Timesteps_Between_Repetitions**
- **spray_coverage** – The portion of the node that has been sprayed. This value is multiplied by the current efficacy of the WaningEffect
- **insecticide** – The name of the insecticide defined in <config.Insecticides> for this intervention. If insecticides are being used, then this must be defined as one of those values. If they are not being used, then this does not needed to be specified or can be empty string. It cannot have a value if <config.Insecticides> does not define anything.
- **intervention_name** – The optional name used to refer to this intervention as a means to differentiate it from others that use the same class. It's possible to have multiple UsageDependentBednets interventions attached to a person if they have different Intervention_Name values.
- **killing_initial_effect** – Initial strength of the Killing effect. The effect may decay over time.
- **killing_box_duration** – Box duration of effect in days before the decay of Killing Initial_Effect. -1 indicates effect is indefinite (WaningEffectConstant)

- **killing_decay_time_constant** – The exponential decay length, in days of the Killing Initial_Effect.
- **cost_to_consumer** – Per unit cost when distributed

Returns:

emodpy_malaria.interventions.spacespraying.**new_intervention_as_file**(*campaign*, *start_day: int = 0*, *filename: str = 'SpaceSpraying.json'*)

Creates a file with SpaceSpray intervention

Parameters

- **campaign** – campaign object to which the intervention will be added, and schema_path container
- **start_day** – the day to distribute the SpaceSpraying intervention
- **filename** – name of the filename created

Returns filename of the file created

emodpy_malaria.interventions.sugartrap module

emodpy_malaria.interventions.sugartrap.**add_scheduled_sugar_trap**(*campaign*, *start_day: int = 0*, *node_ids: Optional[list] = None*, *repetitions: int = 1*, *timesteps_between_repetitions: int = 365*, *cost_to_consumer: float = 0*, *expiration_config: Optional[dict] = None*, *expiration_constant: float = 30*, *insecticide: str = ''*, *intervention_name: str = 'SugarTrap'*, *killing_initial_effect: float = 1*, *killing_box_duration: float = - 1*, *killing_decay_time_constant: float = 0*)

Creates and adds a scheduled intervention that distributes a SugarTrap (ATSB) to the campaign.

Note: for WaningEffect, box_duration = 0 + decay_time_constant > 0 => WaningEffectExponential box_duration > 0 or -1 + decay_time_constant = 0 => WaningEffectBox or Constant if box_duration is -1 box_duration > 0 + decay_time_constant > 0 => WaningEffectBoxExponential

Parameters

- **campaign** – campaign object to which the intervention will be added, and schema_path container
- **start_day** – The day the intervention is given out.
- **node_ids** – List of nodes to which to distribute the intervention. [] or None, indicates all nodes will get the intervention

- **repetitions** – The number of times an intervention is given, used with timesteps_between_repetitions. -1 means the intervention repeats forever. Sets **Number_Repetitions**
- **timesteps_between_repetitions** – The interval, in timesteps, between repetitions. Ignored if repetitions = 1. Sets **Timesteps_Between_Repetitions**
- **cost_to_consumer** – Per unit cost when distributed
- **expiration_config** – (Optional) A dictionary of parameters that define a distribution from which a duration will be selected for when the trap expires. If the trap is distributed on day 1 and has a duration of 10, it will expire on day 10 - 10 days of efficacy including the day of distribution. If the duration is zero, the trap is still distributed but is not applied and expires that day. If this is not defined, 'expiration_constant' parameter is used, a CONSTANT_DISTRIBUTION.

  Examples:

```
Please note this is not "Expiration_Period_*", but just "Expiration_*
↪"
for Gaussian: {"Expiration_Distribution": "GAUSSIAN_DISTRIBUTION",
    "Expiration_Gaussian_Mean": 20, "Expiration_Gaussian_Std_Dev":10}
for Exponential {"Expiration_Distribution": "EXPONENTIAL_DISTRIBUTION
↪",
    "Expiration_Exponential":150}
```

- **expiration_constant** – Each SugarTrap intervention will expire after this exact time. This is overwritten by whatever distribution is defined in 'expiration_config' parameter, if defined. Default is SugarTrap will expire after 30 days.
- **insecticide** – The name of the insecticide defined in <config.Insecticides> for this intervention. If insecticides are being used, then this must be defined as one of those values. If they are not being used, then this does not needed to be specified or can be empty string. It cannot have a value if <config.Insecticides> does not define anything.
- **intervention_name** – The optional name used to refer to this intervention as a means to differentiate it from others that use the same class. It's possible to have multiple UsageDependentBednets interventions attached to a person if they have different Intervention_Name values.
- **killing_initial_effect** – Initial strength of the Killing effect. The effect may decay over time.
- **killing_box_duration** – Box duration of effect in days before the decay of Killing Initial_Effect. -1 indicates effect is indefinite (WaningEffectConstant)
- **killing_decay_time_constant** – The exponential decay length, in days of the Killing Initial_Effect.

Returns Nothing

emodpy_malaria.interventions.sugartrap.**new_intervention_as_file**(*campaign*, *start_day: int = 0*, *filename: str = 'SugarTrap.json'*)

Create new campaign file with a single event which distributes a SugarTrap intervention mostly with defaults. Useful for sanity testing and first time users. :param campaign: campaign builder. :param start_day: the day to distribute the SpaceSpraying intervention :param filename: name of the filename created

Returns Filename of the file created.

emodpy_malaria.interventions.treatment_seeking module

emodpy_malaria.interventions.treatment_seeking.**add_treatment_seeking**(*campaign*, *start_day: int = 1*, *targets: Optional[list] = None*, *drug: Optional[list] = None*, *node_ids: Optional[list] = None*, *ind_property_restrictions: Optional[list] = None*, *drug_ineligibility_duration: float = 0*, *duration: int = -1*, *broadcast_event_name: str = 'ReceivedTreatment'*)

Add an event-triggered drug-seeking behavior intervention to the campaign using the **NodeLevelHealthTriggeredIV**. The intervention will distribute drugs to targeted individuals within the node.

Parameters

- **campaign** – object for building, modifying, and writing campaign configuration files.
- **start_day** – Start day of intervention.
- **targets** – List of dictionaries defining the trigger event and coverage for and
- **is** (*properties of individuals to target with the intervention. Default*) – `[{ "trigger":"NewClinicalCase","coverage":0.8,"agemin":15, "agemax":70, "seek":0.4,"rate":0.3},{"trigger":"NewSevereCase", "coverage":0.8,"seek":0.6, "rate":0.5}]`.
- **drug** – List of drug(s) to administer. Default is `["Artemether","Lumefantrine"]`.
- **node_ids** – The list of nodes to apply this intervention to (**Node_List**
- **NodeSetAll.** (*parameter). If not provided, set value of*) –
- **ind_property_restrictions** – List of IndividualProperty key:value pairs that
- **example,** (*individuals must have to receive the intervention. For*) – `["IndividualProperty1:PropertyValue1", "IndividualProperty2:PropertyValue2"]`.
- **drug_ineligibility_duration** – number of days for which an individual will be ineligible for more drugs
- **duration** – duration from start_day until people will no longer seek drugs when sick. Default is -1, where this never happens.
- **broadcast_event_name** – Event to broadcast when successful health seeking behavior.
- **ReceivedTreatment.** (*Default is*) –

Returns None

emodpy_malaria.interventions.usage_dependent_bednet module

emodpy_malaria.interventions.usage_dependent_bednet.**add_scheduled_usage_dependent_bednet**(*campaign*, *start_day: int = 1*, *demographic_coverage: float = 1*, *target_num_individu Optional[int] = None*, *node_ids: Optional[list] = None*, *ind_property_restr Optional[list] = None*, *intervention_name: str = 'UsageDependentBednet'*, *discard_config: Optional[dict] = None*, *insecticide: str = ''*, *repelling_initial_effe float = 0*, *repelling_box_durat*

ageDependentBednet class.

Note: for WaningEffect, box_duration = 0 + decay_time_constant > 0 => WaningEffectExponential box_duration > 0 + decay_time_constant = 0 => WaningEffectBox/Constant (depending on duration) box_duration > 0 + decay_time_constant > 0 => WaningEffectBoxExponential

Parameters

- **campaign** – campaign object to which the intervention will be added, and schema_path container
- **start_day** – The day on which to start distributing the bednets (**Start_Day** parameter).
- **demographic_coverage** – This value is the probability that each individual in the target population will receive the intervention. It does not guarantee that the exact fraction of the target population set by Demographic_Coverage receives the intervention.
- **target_num_individuals** – The exact number of people to select out of the targeted group. If this value is set, demographic_coverage parameter is ignored
- **node_ids** – The list of nodes to apply this intervention to (**Node_List** parameter). If not provided, intervention is distributed to all nodes.
- **ind_property_restrictions** – A list of dictionaries of IndividualProperties, which are needed for the individual to receive the intervention. Sets the **Property_Restrictions_Within_Node**
- **ind_property_restrictions** – The IndividualProperty key:value pairs that individuals must have to receive the intervention (**Property_Restrictions_Within_Node** parameter). In the format `[{ "BitingRisk":"High"}, {"IsCool":"Yes}]`.
- **intervention_name** – The optional name used to refer to this intervention as a means to differentiate it from others that use the same class. It's possible to have multiple UsageDependentBednets interventions attached to a person if they have different Intervention_Name values.
- **discard_config** – A dictionary of parameters needed to define expiration distribution. No need to definite the distribution with all its parameters Default is bednet being discarded with EXPONENTIAL_DISTRIBUTION with Expiration_Period_Exponential of 10 years

  Examples:

```
for Gaussian: {"Expiration_Period_Distribution": "GAUSSIAN_
↪DISTRIBUTION",
               "Expiration_Period_Gaussian_Mean": 20,
               "Expiration_Period_Gaussian_Std_Dev":10}
for Exponential: {"Expiration_Period_Distribution": "EXPONENTIAL_
↪DISTRIBUTION",
                  "Expiration_Period_Exponential":150}
```

- **insecticide** – The name of the insecticide defined in <config.Insecticides> for this intervention. If insecticides are being used, then this must be defined as one of those values. If they are not being used, then this does not needed to be specified or can be empty string. It cannot have a value if <config.Insecticides> does not define anything.
- **repelling_initial_effect** – Initial strength of the Repelling effect. The effect may decay over time.
- **repelling_box_duration** – Box duration of effect in days before the decay of Repelling Initial_Effect.

- **repelling_decay_time_constant** – The exponential decay length, in days of the Repelling Initial_Effect.
- **blocking_initial_effect** – Initial strength of the Blocking effect. The effect may decay over time.
- **blocking_box_duration** – Box duration of effect in days before the decay of Blocking Initial_Effect.
- **blocking_decay_time_constant** – The exponential decay length, in days of the Blocking Initial_Effect.
- **killing_initial_effect** – Initial strength of the Killing effect. The effect may decay over time.
- **killing_box_duration** – Box duration of effect in days before the decay of Killing Initial_Effect.
- **killing_decay_time_constant** – The exponential decay length, in days of the Killing Initial_Effect.
- **age_dependence** – A dictionary defining the age dependence of net use. Must contain a list of ages in years and list of usage rate. Default is uniform across all ages. Times are in years of age Examples:

```
{"Times":[], "Values":[]} or {"youth_cov":0.7, "youth_min_age":3,
↪"youth_max_age":13}
```

- **seasonal_dependence** – A dictionary defining the seasonal dependence of net use. Time since start will reset to zero once it reaches 365. This allows you to simulate seasonal effects. Times are given in days of the year; values greater than 365 are ignored. Dictionaries can be (times, values) for linear spline or (minimum coverage, day of maximum coverage) for sinusoidal dynamics. Default is constant use during the year. Examples:

```
{"Times":[], "Values":[]} or {"min_cov":0.45, "max_day":300}
```

Returns

None .. note:

```
Previous way of setting discard config is no longer available, you can␣
↪translate it to the current way by:
discard_config the old way {'halflife1': 260, 'halflife2': 2106,
↪'fraction1': float(table_dict['fast_fraction'])
discard_config translated = {"Expiration_Period_Distribution": "DUAL_
↪EXPONENTIAL_DISTRIBUTION",
                             "Expiration_Period_Mean_1": discard_
↪halflife, or halflife1
                             "Expiration_Period_Mean_2": 365 * 40, or␣
↪halflife2
                             "Expiration_Period_Proportion_1": 1 or
↪'fraction1'}
```

Example:

```
discard_config = {"Expiration_Period_Exponential": 10 * 365}
age_dependence = {"Times": [0, 4, 10, 60],
           "Values": [1, 0.9, 0.8, 0.5]}
```

(continues on next page)

(continued from previous page)

```
add_usage_dependent_bednet(campaign, start_day=12, demographic_
↪coverage=0.25,
            age_dependence=age_dependence):
```

emodpy_malaria.interventions.usage_dependent_bednet.**add_triggered_usage_dependent_bednet**(*campaign*, *start_day: int = 1*, *demographic_coverage: float = 1*, *node_ids: Optional[list] = None*, *ind_property_restr Optional[list] = None*, *trigger_condition_list: Optional[list] = None*, *triggered_campaign_d Optional[float] = None*, *listening_duration: int = -1*, *intervention_name: str = 'UsageDependentBednet'*, *discard_config: Optional[dict] = None*, *insec-*

ageDependentBednet class.

Note: for WaningEffect, box_duration = 0 + decay_time_constant > 0 => WaningEffectExponential box_duration > 0 + decay_time_constant = 0 => WaningEffectBox/Constant (depending on duration) box_duration > 0 + decay_time_constant > 0 => WaningEffectBoxExponential

Parameters

- **campaign** – campaign object to which the intervention will be added, and schema_path container
- **start_day** – The day on which to start distributing the bednets (**Start_Day** parameter).
- **demographic_coverage** – This value is the probability that each individual in the target population will receive the intervention. It does not guarantee that the exact fraction of the target population set by Demographic_Coverage receives the intervention.
- **node_ids** – The list of nodes to apply this intervention to (**Node_List** parameter). If not provided, intervention is distributed to all nodes.
- **ind_property_restrictions** – A list of dictionaries of IndividualProperties, which are needed for the individual to receive the intervention. Sets the **Property_Restrictions_Within_Node**
- **trigger_condition_list** – (Optional) A list of the events that will trigger the ITN intervention. If included, **start** is the day when monitoring for triggers begins.
- **triggered_campaign_delay** – (Optional) Delay in days before the intervention is given out after being triggered.
- **listening_duration** – If run as a birth-triggered event or a trigger_condition_list, specifies the duration for the distribution to continue. Default is to continue until the end of the simulation.
- **ind_property_restrictions** – The IndividualProperty key:value pairs that individuals must have to receive the intervention (**Property_Restrictions_Within_Node** parameter). In the format `[{ "BitingRisk":"High"}, {"IsCool":"Yes}]`.
- **intervention_name** – The optional name used to refer to this intervention as a means to differentiate it from others that use the same class. It's possible to have multiple UsageDependentBednets interventions attached to a person if they have different Intervention_Name values.
- **discard_config** – A dictionary of parameters needed to define expiration distribution. No need to definite the distribution with all its parameters Default is bednet being discarded with EXPONENTIAL_DISTRIBUTION with Expiration_Period_Exponential of 10 years

  Examples:

```
for Gaussian: {"Expiration_Period_Distribution": "GAUSSIAN_
↪DISTRIBUTION",
               "Expiration_Period_Gaussian_Mean": 20,
               "Expiration_Period_Gaussian_Std_Dev":10}
for Exponential: {"Expiration_Period_Distribution": "EXPONENTIAL_
↪DISTRIBUTION",
                  "Expiration_Period_Exponential":150}
```

- **insecticide** – The name of the insecticide defined in <config.Insecticides> for this intervention. If insecticides are being used, then this must be defined as one of those values. If they are not being used, then this does not needed to be specified or can be empty string. It cannot have a value if <config.Insecticides> does not define anything.

- **repelling_initial_effect** – Initial strength of the Repelling effect. The effect may decay over time.
- **repelling_box_duration** – Box duration of effect in days before the decay of Repelling Initial_Effect.
- **repelling_decay_time_constant** – The exponential decay length, in days of the Repelling Initial_Effect.
- **blocking_initial_effect** – Initial strength of the Blocking effect. The effect may decay over time.
- **blocking_box_duration** – Box duration of effect in days before the decay of Blocking Initial_Effect.
- **blocking_decay_time_constant** – The exponential decay length, in days of the Blocking Initial_Effect.
- **killing_initial_effect** – Initial strength of the Killing effect. The effect may decay over time.
- **killing_box_duration** – Box duration of effect in days before the decay of Killing Initial_Effect.
- **killing_decay_time_constant** – The exponential decay length, in days of the Killing Initial_Effect.
- **age_dependence** – A dictionary defining the age dependence of net use. Must contain a list of ages in years and list of usage rate. Default is uniform across all ages. Times are in years of age Examples:

```
{"Times":[], "Values":[]} or {"youth_cov":0.7, "youth_min_age":3,
↪"youth_max_age":13}
```

- **seasonal_dependence** – A dictionary defining the seasonal dependence of net use. Time since start will reset to zero once it reaches 365. This allows you to simulate seasonal effects. Times are given in days of the year; values greater than 365 are ignored. Dictionaries can be (times, values) for linear spline or (minimum coverage, day of maximum coverage) for sinusoidal dynamics. Default is constant use during the year. Examples:

```
{"Times":[], "Values":[]} or {"min_cov":0.45, "max_day":300}
```

Returns None

Note: Previous way of setting discard config is no longer available, you can translate it to the current way by: discard_config the old way {'halflife1': 260, 'halflife2': 2106, 'fraction1': float(table_dict['fast_fraction']) discard_config translated = {"Expiration_Period_Distribution": "DUAL_EXPONENTIAL_DISTRIBUTION", "Expiration_Period_Mean_1": discard_halflife, or halflife1 "Expiration_Period_Mean_2": 365 * 40, or halflife2 "Expiration_Period_Proportion_1": 1 or 'fraction1'}

Example:

```
discard_config = {"Expiration_Period_Exponential": 10 * 365}
age_dependence = {"Times": [0, 4, 10, 60],
           "Values": [1, 0.9, 0.8, 0.5]}
add_usage_dependent_bednet(campaign, start=12, coverage=0.25,
            age_dependence=age_dependence):
```

emodpy_malaria.interventions.usage_dependent_bednet.**new_intervention_as_file**(*camp*, *start_day*, *filename='UsageDependentBednet.jso*

emodpy_malaria.interventions.vaccine module

This module contains functionality for vaccine distribution.

emodpy_malaria.interventions.vaccine.**add_scheduled_vaccine**(*campaign*, *start_day: int = 1*, *demographic_coverage: float = 1.0*, *target_num_individuals: Optional[int] = None*, *node_ids: Optional[list] = None*, *repetitions: int = 1*, *timesteps_between_repetitions: int = 365*, *ind_property_restrictions: Optional[list] = None*, *target_age_min: int = 0*, *target_age_max: int = 125*, *target_gender: str = 'All'*, *broadcast_event: Optional[str] = None*, *vaccine_type: str = 'AcquisitionBlocking'*, *vaccine_take: float = 1*, *vaccine_initial_effect: float = 1*, *vaccine_box_duration: int = 365*, *vaccine_decay_time_constant: float = 100*, *efficacy_is_multiplicative: bool = True*)

Adds a scheduled SimpleVaccine event, with an optional BroadcastEvent, broadcast when vaccine is received.

Parameters

- **campaign** – campaign object to which the intervention will be added, and schema_path container
- **start_day** – The day the intervention is given out.
- **demographic_coverage** – This value is the probability that each individual in the target population will receive the intervention. It does not guarantee that the exact fraction of the target population set by Demographic_Coverage receives the intervention.
- **target_num_individuals** – The exact number of people to select out of the targeted group. If this value is set, demographic_coverage parameter is ignored
- **node_ids** – List of nodes to which to distribute the intervention. [] or None, indicates all nodes will get the intervention
- **repetitions** – The number of times an intervention is given, used with timesteps_between_repetitions. -1 means the intervention repeats forever. Sets **Number_Repetitions**
- **timesteps_between_repetitions** – The interval, in timesteps, between repetitions. Ignored if repetitions = 1. Sets **Timesteps_Between_Repetitions**
- **ind_property_restrictions** – A list of dictionaries of IndividualProperties, which are needed for the individual to receive the intervention. Sets the **Property_Restrictions_Within_Node**

- **target_age_min** – The lower end of ages targeted for an intervention, in years. Sets **Target_Age_Min**
- **target_age_max** – The upper end of ages targeted for an intervention, in years. Sets **Target_Age_Max**
- **target_gender** – The gender targeted for an intervention: All, Male, or Female.
- **broadcast_event** – "The name of the event to be broadcast. This event must be set in the **Custom_Coordinator_Events** configuration parameter. When None or Empty, nothing is broadcast.
- **vaccine_type** – The type of vaccine to distribute in a vaccine intervention. Options are: "Generic", "TransmissionBlocking", "AcquisitionBlocking", "MortalityBlocking"
- **vaccine_take** – The rate at which delivered vaccines will successfully stimulate an immune response and achieve the desired efficacy.
- **vaccine_initial_effect** – Initial efficacy of the vaccine, before decay.
- **vaccine_box_duration** – Duration in days of initial efficacy of vaccine before it starts to decay.
- **vaccine_decay_time_constant** – Time over which vaccine efficacy wanes and the vaccine_box_duration.
- **efficacy_is_multiplicative** – The overall vaccine efficacy when individuals receive more than one vaccine. When set to true (1), the vaccine efficacies are multiplied together; when set to false (0), the efficacies are additive.

Returns Nothing

emodpy_malaria.interventions.vaccine.**add_triggered_vaccine**(*campaign*, *start_day: int = 1*, *trigger_condition_list: Optional[list] = None*, *listening_duration: int = - 1*, *delay_period_constant: float = 0*, *demographic_coverage: float = 1.0*, *node_ids: Optional[list] = None*, *repetitions: int = 1*, *timesteps_between_repetitions: int = 365*, *ind_property_restrictions: Optional[list] = None*, *target_age_min: int = 0*, *target_age_max: int = 125*, *target_gender: str = 'All'*, *broadcast_event: Optional[str] = None*, *vaccine_type: str = 'AcquisitionBlocking'*, *vaccine_take: float = 1*, *vaccine_initial_effect: float = 1*, *vaccine_box_duration: int = 365*, *vaccine_decay_time_constant: float = 100*, *efficacy_is_multiplicative: bool = True*)

Adds an event-triggered SimpleVaccine event, with an optional BroadcastEvent, broadcast when vaccine is received.

Parameters

- **campaign** – campaign object to which the intervention will be added, and schema_path container
- **start_day** – The day the intervention is given out.
- **trigger_condition_list** – A list of the events that will trigger intervention distribution.
- **listening_duration** – The number of time steps that the distributed event will monitor for triggers. Default is -1, which is indefinitely.
- **delay_period_constant** – Optional. Delay, in days, before the intervention is given out after a trigger is received.
- **demographic_coverage** – This value is the probability that each individual in the target population will receive the intervention. It does not guarantee that the exact fraction of the target population set by Demographic_Coverage receives the intervention.
- **node_ids** – List of nodes to which to distribute the intervention. [] or None, indicates all nodes will get the intervention
- **repetitions** – The number of times an intervention is given, used with timesteps_between_repetitions. -1 means the intervention repeats forever. Sets **Number_Repetitions**
- **timesteps_between_repetitions** – The interval, in timesteps, between repetitions. Ignored if repetitions = 1. Sets **Timesteps_Between_Repetitions**
- **ind_property_restrictions** – A list of dictionaries of IndividualProperties, which are needed for the individual to receive the intervention. Sets the **Property_Restrictions_Within_Node**
- **target_age_min** – The lower end of ages targeted for an intervention, in years. Sets **Target_Age_Min**
- **target_age_max** – The upper end of ages targeted for an intervention, in years. Sets **Target_Age_Max**
- **target_gender** – The gender targeted for an intervention: All, Male, or Female.
- **broadcast_event** – "The name of the event to be broadcast. This event must be set in the **Custom_Coordinator_Events** configuration parameter. When None or Empty, nothing is broadcast.
- **vaccine_type** – The type of vaccine to distribute in a vaccine intervention. Options are: "Generic", "TransmissionBlocking", "AcquisitionBlocking", "MortalityBlocking"
- **vaccine_take** – The rate at which delivered vaccines will successfully stimulate an immune response and achieve the desired efficacy.
- **vaccine_initial_effect** – Initial efficacy of the vaccine, before decay.
- **vaccine_box_duration** – Duration in days of initial efficacy of vaccine before it starts to decay.
- **vaccine_decay_time_constant** – Time over which vaccine efficacy wanes and the vaccine_box_duration.
- **efficacy_is_multiplicative** – The overall vaccine efficacy when individuals receive more than one vaccine. When set to true (1), the vaccine efficacies are multiplied together; when set to false (0), the efficacies are additive.

Returns Nothing

emodpy_malaria.interventions.vaccine.**new_intervention_as_file**(*campaign*, *start_day: int = 0*, *filename: str = 'SimpleVaccine.json'*)

Write a campaign file to disk with a single bednet event, using defaults. Useful for testing and learning.

Parameters

- **campaign** – The emod_api.campaign object to which the intervention will be added.
- **start_day** – The day of the simulation on which the bednets are distributed. We recommend aligning this with the start of the simulation.
- **filename** – The campaign filename; can be omitted and default will be used and returned to user.

Returns The campaign filename written to disk.

emodpy_malaria.reporters package

Submodules

emodpy_malaria.reporters.builtin module

emodpy_malaria.reporters.builtin.**add_report_vector_genetics**(*task*, *manifest*, *start_day: int = 0*, *end_day: int = 365000*, *node_ids: Optional[list] = None*, *species: Optional[str] = None*, *gender: str = 'VECTOR_FEMALE'*, *include_vector_state: int = 1*, *include_death_state: int = 0*, *stratify_by: str = 'GENOME'*, *combine_similar_genomes: int = 0*, *specific_genome_combinations_for_stratification: Optional[list] = None*, *allele_combinations_for_stratification: Optional[list] = None*, *alleles_for_stratification: Optional[list] = None*, *filename_suffix: str = ''*)

Adds ReportVectorGenetics to the simulation. See class definition for description of the report.

Parameters

- **task** – task to which to add the reporter, if left as None, reporter is returned (used for unittests)
- **manifest** – schema path file
- **start_day** – the day of the simulation to start reporting data
- **end_day** – the day of the simulation to stop reporting data
- **node_ids** – the list of nodes in which to collect data, empty or None means all nodes
- **species** – the species to include information on
- **gender** – gender of species to include information on. Default: “VECTOR_FEMALE”, other options: “VECTOR_MALE”, “VECTOR_BOTH_GENDERS”

- **include_vector_state** – if 1(true), adds the columns for vectors in the different states (i.e Eggs, Larva, etc)
- **include_death_state** – if 1(true), adds columns for the number of vectors that died in this state during this time step as well as the average age. It adds two columns for each of the following states: ADULT, INFECTED, INFECTIOUS, and MALE
- **stratify_by** – the way to stratify data. Default: "GENOME", other options: "SPECIFIC_GENOME", "ALLELE", "ALLELE_FREQ"
- **combine_similar_genomes** – if 1(true), genomes are combined if for each locus (ignoring gender) the set of allele of the two genomes are the same (i.e. 1-0 is similar to 0-1). Depends on: "GENOME", "SPECIFIC_GENOME" specific_genome_combinations_for_stratification: if stratifying by "SPECIFIC_GENOME", then use these genomes to stratify by. Example:

```
[{"Allele_Combination": [[ "a0",  "*" ], [ "b1", "b0" ]]},
{"Allele_Combination": [[ "a1", "a0" ], [ "b0",  "*" ]]}]
```

- **specific_genome_combinations_for_stratification** – ff stratifying by "SPECIFIC_GENOME", then use these genomes to stratify by. '*' = list all entries at that location, '?' = combine all entries at that location
- **allele_combinations_for_stratification** – if stratifying by "ALLELE", then also add these allele name combos to the stratification, Example:

```
[[ "a0", "b0" ], [ "a1", "b1" ]]
```

- **alleles_for_stratification** – For example:

```
[ "a0", "a1", "b0", "b1" ]
```

- **filename_suffix** – augments the filename of the report. If multiple reports are being generated, this allows you to distinguish among the multiple reports

Returns if task is not set, returns the configured reporter, otherwise returns nothing

emodpy_malaria.reporters.builtin.**add_report_vector_stats**(*task*, *manifest*, *species_list: Optional[list] = None*, *stratify_by_species: int = 0*, *include_death_state: int = 0*, *include_wolbachia: int = 0*, *include_gestation: int = 0*)

Adds ReportVectorStats report to the simulation. See class definition for description of the report.

Parameters

- **task** – task to which to add the reporter, if left as None, reporter is returned (used for unittests)
- **manifest** – schema path file
- **species_list** – a list of species to include information on
- **stratify_by_species** – if 1(true), data will break out each the species for each node
- **include_death_state** – if 1(true), adds columns for the number of vectors that died in this state during this time step as well as the average age. It adds two columns for each of the following states: ADULT, INFECTED, INFECTIOUS, and MALE
- **include_wolbachia** – if 1(true), add a column for each type of Wolbachia
- **include_gestation** – if 1(true), adds columns for feeding and gestation

Returns if task is not set, returns the configured reporter, otherwise returns nothing

emodpy_malaria.reporters.builtin.**add_malaria_summary_report**(*task*, *manifest*, *start_day: int = 0*, *end_day: int = 365000*, *node_ids: Optional[list] = None*, *reporting_interval: float = 365*, *must_have_ip_key_value: str = ''*, *must_have_intervention: str = ''*, *age_bins: Optional[list] = None*, *infectiousness_bins: Optional[list] = None*, *max_number_reports: int = 100*, *parasitemia_bins: Optional[list] = None*, *pretty_format: int = 0*, *filename_suffix: str = ''*)

Adds MalariaSummaryReport to the simulation. See class definition for description of the report.

Parameters

- **task** – task to which to add the reporter, if left as None, reporter is returned (used for unittests)
- **manifest** – schema path file
- **start_day** – the day of the simulation to starts collecting data for the report
- **end_day** – the day of the simulation to stop reporting data
- **node_ids** – a list of nodes from which to collect data for the report
- **reporting_interval** – Defines the cadence of the report by specifying how many time steps to collect data before writing to the file
- **must_have_ip_key_value** – a "Key:Value" pair that the individual must have in order to be included. Empty string means don't look at IPs (individual properties)
- **must_have_intervention** – the name of the an intervention that the person must have in order to be included. Empty string means don't look at the interventions
- **age_bins** – The max age in years per bin, listed in ascending order. Use a large value for the last bin, to collect all remaining individuals
- **infectiousness_bins** – infectiousness Bins to aggregate within for the report
- **max_number_reports** – the maximum number of report output files that will be produced for a given simulation
- **parasitemia_bins** – Parasitemia bins on which to aggregate. A value <= 0 in the first bin indicates that uninfected individuals are added to this bin. You must sort your input data from low to high.
- **pretty_format** – if 1(true) sets pretty JSON formatting, which includes carriage returns, line feeds, and spaces for easier readability. The default, 0 (false), saves space where everything is on one line.
- **filename_suffix** – augments the filename of the report. If multiple reports are being generated, this allows you to distinguish among the multiple reports

Returns if task is not set, returns the configured reporter, otherwise returns nothing

emodpy_malaria.reporters.builtin.**add_malaria_patient_json_report**(*task*, *manifest*, *start_day: int = 0*, *end_day: int = 365000*, *node_ids: Optional[list] = None*, *min_age_years: float = 0*, *max_age_years: float = 125*, *must_have_ip_key_value: str = ''*, *must_have_intervention: str = ''*, *filename_suffix: str = ''*)

Adds MalariaPatientJSONReport report to the simulation. See class definition for description of the report.

Parameters

- **task** – task to which to add the reporter, if left as None, reporter is returned (used for unittests)
- **manifest** – schema path file
- **start_day** – the day of the simulation to starts collecting data for the report
- **end_day** – the day of the simulation to stop reporting data
- **node_ids** – a list of nodes from which to collect data for the report
- **min_age_years** – minimum age in years of people to collect data on
- **max_age_years** – maximum age in years of people to collect data on
- **must_have_ip_key_value** – a "Key:Value" pair that the individual must have in order to be included. Empty string means don't look at IPs (individual properties)
- **must_have_intervention** – the name of the an intervention that the person must have in order to be included. Empty string means don't look at the interventions
- **filename_suffix** – augments the filename of the report. If multiple reports are being generated, this allows you to distinguish among the multiple reports

Returns if task is not set, returns the configured reporter, otherwise returns nothing

emodpy_malaria.reporters.builtin.**add_malaria_transmission_report**(*task*, *manifest*, *start_day: int = 0*, *end_day: int = 365000*, *node_ids: Optional[list] = None*, *min_age_years: float = 0*, *max_age_years: float = 125*, *must_have_ip_key_value: str = ''*, *must_have_intervention: str = ''*, *include_human_to_vector: int = 0*, *pretty_format: int = 0*, *filename_suffix: str = ''*)

Adds ReportSimpleMalariaTransmissionJSON report to the simulation. See class definition for description of the report.

Parameters

- **task** – task to which to add the reporter, if left as None, reporter is returned (used for unittests)
- **manifest** – schema path file
- **start_day** – the day to start collecting data for the report.
- **end_day** – the day of the simulation to stop reporting data
- **node_ids** – list of nodes for which to collect data for the report

- **min_age_years** – minimum age in years of people to collect data on
- **max_age_years** – maximum age in years of people to collect data on
- **include_human_to_vector** – ff set to 1, Human-to-Vector transmission events will be included. One can identify these events because the 'acquireIndividualId'=0 and transmitTime=acquireTime. WARNING: This can make the file size quite large
- **must_have_ip_key_value** – a "Key:Value" pair that the individual must have in order to be included. Empty string means don't look at IPs (individual properties)
- **must_have_intervention** – the name of the an intervention that the person must have in order to be included. Empty string means don't look at the interventions
- **pretty_format** – if 1(true) sets pretty JSON formatting, which includes carriage returns, line feeds, and spaces for easier readability. The default, 0 (false), saves space where everything is on one line.
- **filename_suffix** – augments the filename of the report. If multiple reports are being generated, this allows you to distinguish among the multiple reports

Returns if task is not set, returns the configured reporter, otherwise returns nothing

emodpy_malaria.reporters.builtin.**add_report_malaria_filtered**(*task*, *manifest*, *start_day: int = 0*, *end_day: int = 365000*, *node_ids: Optional[list] = None*, *min_age_years: float = 0*, *max_age_years: float = 125*, *must_have_ip_key_value: str = ''*, *must_have_intervention: str = ''*, *has_interventions: Optional[list] = None*, *include_30day_avg_infection_duration: int = 1*, *filename_suffix: str = ''*)

Adds ReportMalariaFiltered report to the simulation. See class definition for description of the report.

Parameters

- **task** – task to which to add the reporter, if left as None, reporter is returned (used for unittests)
- **manifest** – schema path file
- **start_day** – the day of the simulation to start collecting data
- **end_day** – the day of simulation to stop collecting data
- **node_ids** – list of nodes for which to collect the data, None or [] collects all the nodes
- **min_age_years** – Minimum age in years of people to collect data on
- **max_age_years** – Maximum age in years of people to collect data on
- **must_have_ip_key_value** – a "Key:Value" pair that the individual must have in order to be included. Empty string means don't look at IPs (individual properties)
- **must_have_intervention** – the name of the an intervention that the person must have in order to be included. Empty string means don't look at the interventions
- **has_interventions** – a list of intervention names, a channel is added to the report for each InterventionName provided. The channel name will be Has_<InterventionName> and will be the fraction of the population that has that intervention. The **Intervention_Name** in the campaign should be the values in this parameter

- **include_30day_avg_infection_duration** – if (1) true the '30-Day Avg Infection Duration' channel is included in the report
- **filename_suffix** – augments the filename of the report. If multiple reports are being generated, this allows you to distinguish among the multiple reports

Returns if task is not set, returns the configured reporter, otherwise returns nothing

emodpy_malaria.reporters.builtin.**add_report_malaria_filtered_intrahost**(*task*, *manifest*, *start_day: int = 0*, *end_day: int = 365000*, *node_ids: Optional[list] = None*, *min_age_years: float = 0*, *max_age_years: float = 125*, *must_have_ip_key_value: str = ''*, *must_have_intervention: str = ''*, *has_interventions: Optional[list] = None*, *include_30day_avg_infection_duration: int = 1*, *filename_suffix: str = ''*)

Adds ReportMalariaFilteredIntraHost report to the simulation. See class definition for description of the report.

Parameters

- **task** – task to which to add the reporter, if left as None, reporter is returned (used for unittests)
- **manifest** – schema path file
- **start_day** – the day of the simulation to start collecting data
- **end_day** – the day of simulation to stop collecting data
- **node_ids** – list of nodes for which to collect the data, None or [] collects all the nodes
- **min_age_years** – Minimum age in years of people to collect data on
- **max_age_years** – Maximum age in years of people to collect data on
- **must_have_ip_key_value** – a "Key:Value" pair that the individual must have in order to be included. Empty string means don't look at IPs (individual properties)
- **must_have_intervention** – the name of the an intervention that the person must have in order to be included. Empty string means don't look at the interventions
- **has_interventions** – a list of intervention names, a channel is added to the report for each InterventionName provided. The channel name will be Has_<InterventionName> and will be the fraction of the population that has that intervention. The **Intervention_Name** in the campaign should be the values in this parameter
- **include_30day_avg_infection_duration** – if (1) true the '30-Day Avg Infection Duration' channel is included in the report
- **filename_suffix** – augments the filename of the report. If multiple reports are being generated, this allows you to distinguish among the multiple reports

Returns if task is not set, returns the configured reporter, otherwise returns nothing

emodpy_malaria.reporters.builtin.**add_spatial_report_malaria_filtered**(*task*, *manifest*, *start_day: int = 0*, *end_day: int = 365000*, *reporting_interval: int = 1*, *node_ids: Optional[list] = None*, *min_age_years: float = 0*, *max_age_years: float = 125*, *must_have_ip_key_value: str = ''*, *must_have_intervention: str = ''*, *spatial_output_channels: Optional[list] = None*, *filename_suffix: str = ''*)

Adds SpatialReportMalariaFiltered report to the simulation. See class definition for description of the report.

Parameters

- **task** – task to which to add the reporter, if left as None, reporter is returned (used for unittests)
- **manifest** – schema path file
- **start_day** – the day of the simulation to start collecting data
- **end_day** – the day of simulation to stop collecting data
- **reporting_interval** – defines the cadence of the report by specifying how many time steps to collect data before writing to the file.
- **node_ids** – list of nodes for which to collect the data, None or [] collects all the nodes
- **min_age_years** – Minimum age in years of people to collect data on
- **max_age_years** – Maximum age in years of people to collect data on
- **must_have_ip_key_value** – a "Key:Value" pair that the individual must have in order to be included. Empty string means don't look at IPs (individual properties)
- **must_have_intervention** – the name of the an intervention that the person must have in order to be included. Empty string means don't look at the interventions
- **spatial_output_channels** – list of names of channels you want to have output for. Available channels are: "Adult_Vectors", "Air_Temperature", "Births", "Blood_Smear_Gametocyte_Prevalence", "Blood_Smear_Parasite_Prevalence", "Campaign_Cost", "Daily_Bites_Per_Human", "Daily_EIR", "Disease_Deaths", "Fever_Prevalence", "Human_Infectious_Reservoir", "Infected", "Infectious_Vectors", "Land_Temperature", "Mean_Parasitemia", "New_Clinical_Cases", "New_Infections", "New_Reported_Infections", "New_Severe_Cases", "PCR_Gametocyte_Prevalence", "PCR_Parasite_Prevalence", "PfHRP2_Prevalence", "Population", "Prevalence", "Rainfall", "Relative_Humidity", "True_Prevalence" Defaults: ["Blood_Smear_Parasite_Prevalence", "New_Clinical_Cases", "Population"]
- **filename_suffix** – augments the filename of the report. If multiple reports are being generated, this allows you to distinguish among the multiple reports

Returns if task is not set, returns the configured reporter, otherwise returns nothing

emodpy_malaria.reporters.builtin.**add_report_event_counter**(*task*, *manifest*, *start_day: int = 0*, *end_day: int = 365000*, *node_ids: Optional[list] = None*, *event_trigger_list: Optional[list] = None*, *min_age_years: float = 0*, *max_age_years: float = 125*, *must_have_ip_key_value: str = ''*, *must_have_intervention: str = ''*, *filename_suffix: str = ''*)

Adds ReportEventCounter report to the simulation. See class definition for description of the report.

Parameters

- **task** – task to which to add the reporter, if left as None, reporter is returned (used for unittests)
- **manifest** – schema path file
- **start_day** – the day of the simulation to start counting events
- **end_day** – the day of simulation to stop collecting data
- **node_ids** – list of nodes in which to count the events
- **event_trigger_list** – list of events which to count
- **min_age_years** – Minimum age in years of people to collect data on
- **max_age_years** – Maximum age in years of people to collect data on
- **must_have_ip_key_value** – a "Key:Value" pair that the individual must have in order to be included. Empty string means don't look at IPs (individual properties)
- **must_have_intervention** – the name of the an intervention that the person must have in order to be included. Empty string means don't look at the interventions
- **filename_suffix** – augments the filename of the report. If multiple reports are being generated, this allows you to distinguish among the multiple reports

Returns if task is not set, returns the configured reporter, otherwise returns nothing

emodpy_malaria.reporters.builtin.**add_malaria_sql_report**(*task*, *manifest*, *start_day: int = 0*, *end_day: int = 365000*, *include_infection_table: int = 1*, *include_health_table: int = 1*, *include_drug_table: int = 0*)

Adds MalariaSqlReport report to the simulation. See class definition for description of the report.

Parameters

- **task** – task to which to add the reporter, if left as None, reporter is returned (used for unittests)
- **manifest** – schema path file
- **start_day** – the day of the simulation to start collecting data
- **end_day** – the day of the simulation to stop collecting data
- **include_infection_table** – if 1(true), include the table that provides data at each time step for each active infection
- **include_health_table** – if 1(true), include the table that provides data at each time step for a person's health

- **include_drug_table** – if 1(true), include the table that provides data at each time step for each drug used

Returns if task is not set, returns the configured reporter, otherwise returns nothing

emodpy_malaria.reporters.builtin.**add_vector_habitat_report**(*task*, *manifest*)

Adds VectorHabitatReport report to the simulation. See class definition for description of the report. You do not need to configure any data parameters to generate this report.

Parameters

- **task** – task to which to add the reporter, if left as None, reporter is returned (used for unittests)
- **manifest** – schema path file

Returns if task is not set, returns the configured reporter, otherwise returns nothing

emodpy_malaria.reporters.builtin.**add_malaria_immunity_report**(*task*, *manifest*, *start_day: int = 0*, *end_day: int = 365000*, *node_ids: Optional[list] = None*, *reporting_interval: int = 1*, *max_number_reports: int = 365000*, *age_bins: Optional[list] = None*, *must_have_ip_key_value: str = ''*, *must_have_intervention: str = ''*, *pretty_format: int = 0*, *filename_suffix: str = ''*)

Adds MalariaImmunityReport report to the simulation. See class definition for description of the report.

Parameters

- **task** – task to which to add the reporter, if left as None, reporter is returned (used for unittests)
- **manifest** – schema path file
- **start_day** – the day of the simulation to start collecting data
- **end_day** – the day of simulation to stop collecting data
- **node_ids** – list of nodes for which to collect data
- **reporting_interval** – defines the cadence of the report by specifying how many time steps to collect data before writing to the file.
- **max_number_reports** – the maximum number of report output files that will be produced for a given simulation
- **age_bins** – The max age in years per bin, listed in ascending order. Use a large value for the last bin, to collect all remaining individuals
- **must_have_ip_key_value** – a "Key:Value" pair that the individual must have in order to be included. Empty string means don't look at IPs (individual properties)
- **must_have_intervention** – the name of the an intervention that the person must have in order to be included. Empty string means don't look at the interventions
- **pretty_format** – if 1(true) sets pretty JSON formatting, which includes carriage returns, line feeds, and spaces for easier readability. The default, 0 (false), saves space where everything is on one line.
- **filename_suffix** – augments the filename of the report. If multiple reports are being generated, this allows you to distinguish among the multiple reports

Returns if task is not set, returns the configured reporter, otherwise returns nothing

emodpy_malaria.reporters.builtin.**add_malaria_survey_analyzer**(*task*, *manifest*, *start_day: int = 0*, *end_day: int = 365000*, *node_ids: Optional[list] = None*, *event_trigger_list: Optional[list] = None*, *reporting_interval: float = 1*, *max_number_reports: int = 365000*, *ip_key_to_collect: str = ''*, *must_have_ip_key_value: str = ''*, *must_have_intervention: str = ''*, *pretty_format: int = 0*, *filename_suffix: str = ''*)

Adds MalariaSurveyJSONAnalyzer report to the simulation. See class definition for description of the report.

Parameters

- **task** – task to which to add the reporter, if left as None, reporter is returned (used for unittests)
- **manifest** – schema path file
- **start_day** – the day of the simulation to start collecting data
- **end_day** – the day of simulation to stop collecting data
- **reporting_interval** – defines the cadence of the report by specifying how many time steps to collect data before writing to the file
- **event_trigger_list** – list of individual events to include into the report
- **max_number_reports** – the maximum number of report output files that will be produced for a given simulation
- **node_ids** – list of nodes for which to collect data
- **ip_key_to_collect** – name of the Individual Property Key whose value to collect. Empty string means collect values for all Individual Properties
- **must_have_ip_key_value** – a "Key:Value" pair that the individual must have in order to be included. Empty string means don't look at IPs (individual properties)
- **must_have_intervention** – the name of the an intervention that the person must have in order to be included. Empty string means don't look at the interventions
- **pretty_format** – if 1(true) sets pretty JSON formatting, which includes carriage returns, line feeds, and spaces for easier readability. The default, 0 (false), saves space where everything is on one line.
- **filename_suffix** – augments the filename of the report. If multiple reports are being generated, this allows you to distinguish among the multiple reports

Returns if task is not set, returns the configured reporter, otherwise returns nothing

emodpy_malaria.reporters.builtin.**add_drug_status_report**(*task*, *manifest*, *start_day: int = 0*, *end_day: int = 365000*)

Adds ReportDrugStatus report to the simulation. See class definition for description of the report.

Parameters

- **task** – task to which to add the reporter, if left as None, reporter is returned (used for unittests)
- **manifest** – schema path file

- **start_day** – the day of the simulation to start collecting data
- **end_day** – the day of the simulation to stop collecting data

Returns if task is not set, returns the configured reporter, otherwise returns nothing

emodpy_malaria.reporters.builtin.**add_report_infection_stats_malaria**(*task*, *manifest*, *start_day: int = 0*, *end_day: int = 365000*)

Adds ReportInfectionStatsMalaria report to the simulation. See class definition for description of the report.

Parameters

- **task** – task to which to add the reporter, if left as None, reporter is returned (used for unittests)
- **manifest** – schema path file
- **start_day** – the day of the simulation to start collecting data
- **end_day** – the day of the simulation to stop collecting data

Returns if task is not set, returns the configured reporter, otherwise returns nothing

emodpy_malaria.reporters.builtin.**add_human_migration_tracking**(*task*, *manifest*)

Adds ReportHumanMigrationTracking report to the simulation. There are no special parameter that need to be configured to generate the report. However, the simulation must have migration enabled.

Parameters

- **task** – task to which to add the reporter, if left as None, reporter is returned (used for unittests)
- **manifest** – schema path file

Returns if task is not set, returns the configured reporter, otherwise returns nothing

emodpy_malaria.reporters.builtin.**add_report_node_demographics**(*task*, *manifest*, *age_bins: Optional[list] = None*, *ip_key_to_collect: str = ''*, *stratify_by_gender: int = 1*)

Adds ReportNodeDemographics report to the simulation. See class definition for description of the report.

Parameters

- **task** – task to which to add the reporter, if left as None, reporter is returned (used for unittests)
- **manifest** – schema path file
- **age_bins** – the age bins (in years) to aggregate within and report. An empty array does not stratify by age. You must sort your input data from low to high.
- **ip_key_to_collect** – The name of the Individual Properties Key by which to stratify the report. An empty string does not stratify by Individual Properties
- **stratify_by_gender** – if 1(true), to stratify by gender. Set to false (0) to not stratify by gender.

Returns if task is not set, returns the configured reporter, otherwise returns nothing

emodpy_malaria.reporters.builtin.**add_report_node_demographics_malaria**(*task*, *manifest*, *age_bins: Optional[list] = None*, *ip_key_to_collect: str = ''*, *stratify_by_gender: int = 1*, *stratify_by_clinical_symptoms: int = 0*)

Adds ReportNodeDemographicsMalaria report to the simulation. See class definition for description of the report.

Parameters

- **task** – task to which to add the reporter, if left as None, reporter is returned (used for unittests)
- **manifest** – schema path file
- **age_bins** – the age bins (in years) to aggregate within and report. An empty array does not stratify by age. You must sort your input data from low to high.
- **ip_key_to_collect** – The name of theIndividualProperties key by which to stratify the report. An empty string does not stratify by Individual Properties
- **stratify_by_gender** – if 1(true), to stratify by gender. Set to false (0) to not stratify by gender.
- **stratify_by_clinical_symptoms** – if set to 1, the data will have an extra stratification for people who have clinical symptoms and those that do not. Default is 0 or no extra stratification

Returns if task is not set, returns the configured reporter, otherwise returns nothing

emodpy_malaria.reporters.builtin.**add_report_node_demographics_malaria_genetics**(*task*, *manifest*, *barcodes: Optional[list] = None*, *drug_resistant_strings: Optional[list] = None*, *drug_resistant_statistic_type: str = 'NUM_PEOPLE_WITH_RESIS*
age_bins: Optional[list] = None, *ip_key_to_collect: str = ''*, *stratify_by_gender: int = 1*)

Adds ReportNodeDemographicsMalariaGenetics report to the simulation. See class definition for description of the report.

Parameters

- **task** – task to which to add the reporter, if left as None, reporter is returned (used for unittests)
- **manifest** – schema path file
- **barcodes** – a list of barcode strings. The report contains the number of human infections with each barcode. Use '*' for a wild card at a loci to include all values at that loci. For example, "A*T" includes AAT, ACT, AGT, and ATT. The report contains a BarcodeOther column for barcodes that are not defined. Note: There is no validation that the barcode strings are valid barcodes for the scenario.
- **drug_resistant_strings** – a list of strings representing the set of drug resistant markers. A column will be created with the number of humans infetions with that barcode. One can use '*' for a wild card. A 'BarcodeOther' column will be created for barcodes not define
- **drug_resistant_statistic_type** – indicates the statistic in the Drug Resistant columns: NUM_PEOPLE_WITH_RESISTANT_INFECTION = A person is counted if they have one infection with that drug resistant marker; NUM_INFECTIONS = The total number of infections with that marker.
- **age_bins** – the age bins (in years) to aggregate within and report. An empty array does not stratify by age. You must sort your input data from low to high.
- **ip_key_to_collect** – The name of theIndividualProperties key by which to stratify the report. An empty string does not stratify by Individual Properties
- **stratify_by_gender** – if 1(true), to stratify by gender. Set to false (0) to not stratify by gender.

Returns if task is not set, returns the configured reporter, otherwise returns nothing

emodpy_malaria.reporters.builtin.**add_report_vector_migration**(*task*, *manifest*, *start_day: int = 0*, *end_day: int = 365000*)

Adds ReportVectorMigration report to the simulation. See class definition for description of the report.

Parameters

- **task** – task to which to add the reporter, if left as None, reporter is returned (used for unittests)
- **manifest** – schema path file
- **start_day** – the day of the simulation to start collecting data
- **end_day** – the day of the simulation to stop collecting data

Returns if task is not set, returns the configured reporter, otherwise returns nothing

emodpy_malaria.reporters.builtin.**add_report_vector_stats_malaria_genetics**(*task*, *manifest*, *species_list: Optional[list] = None*, *stratify_by_species: int = 0*, *include_death_state: int = 0*, *include_wolbachia: int = 0*, *include_gestation: int = 0*, *barcodes: Optional[list] = None*)

Adds ReportVectorStatsMalariaGenetics report to the simulation. See class definition for description of the report.

Parameters

- **task** – task to which to add the reporter, if left as None, reporter is returned (used for unittests)
- **manifest** – schema path file
- **species_list** – a list of species to include information on
- **stratify_by_species** – if 1(true), data will break out each the species for each node
- **include_death_state** – if 1(true), adds columns for the number of vectors that died in this state during this time step as well as the average age. It adds two columns for each of the following states: ADULT, INFECTED, INFECTIOUS, and MALE
- **include_wolbachia** – if 1(true), add a column for each type of Wolbachia
- **include_gestation** – if 1(true), adds columns for feeding and gestation
- **barcodes** – a list of barcode strings. The report contains the number of human infections with each barcode. Use '*' for a wild card at a loci to include all values at that loci. For example, "A*T" includes AAT, ACT, AGT, and ATT. The report contains a BarcodeOther column for barcodes that are not defined. Note: There is no validation that the barcode strings are valid barcodes for the scenario.

Returns if task is not set, returns the configured reporter, otherwise returns nothing

emodpy_malaria.reporters.builtin.**add_event_recorder**(*task*, *event_list: Optional[list] = None*, *only_include_events_in_list: bool = True*, *ips_to_record: Optional[list] = None*, *start_day: int = 0*, *end_day: int = 365000*, *node_ids: Optional[list] = None*, *min_age_years: float = 0*, *max_age_years: float = 365000*, *must_have_ip_key_value: str = ''*, *must_have_intervention: str = ''*, *property_change_ip_to_record: str = ''*)

Adds ReportEventRecorder report to the simulation. See class definition for description of the report.

Parameters

- **task** – task to which to add the reporter
- **event_list** – a list of events to record or exclude, depending on value of only_include_events_in_list

- **only_include_events_in_list** – if True, only record events listed. if False, record ALL events EXCEPT for the ones listed
- **ips_to_record** – list of individual properties to include in report
- **start_day** – The day of the simulation to start collecting data
- **end_day** – The day of the simulation to stop collecting data.
- **node_ids** – Data will be collected for the nodes in this list, if None - all nodes have data collected.
- **min_age_years** – Minimum age in years of people to collect data on
- **max_age_years** – Maximum age in years of people to collect data on
- **must_have_ip_key_value** – A Key:Value pair that the individual must have in order to be included. Empty string means don't look at IndividualProperties
- **must_have_intervention** – The name of the an intervention that the person must have in order to be included. Empty string means don't look at the interventions
- **property_change_ip_to_record** – If the string is not empty, then the recorder will add the PropertyChange event to the list of events that the report is listening to. However, it will only record the events where the property changed the value of the given key

Returns Nothing

emodpy_malaria.reporters.builtin.**add_report_intervention_pop_avg**(*task*, *manifest*, *start_day: int = 0*, *end_day: int = 36500000*, *node_ids: Optional[list] = None*, *min_age_years: float = 0*, *max_age_years: float = 365000*, *must_have_ip_key_value: str = ''*, *must_have_intervention: str = ''*, *filename_suffix: str = ''*)

Adds ReportInterventionPopAvg reporter. See class definition for description of the report.

Parameters

- **task** – Task to which to add the reporter, if left as None, reporter is returned (used for unittests)
- **manifest** – Schema path file
- **start_day** – the day of the simulation to start collecting data
- **end_day** – the day of the simulation to stop collecting data
- **node_ids** – List of nodes for which to collect data
- **min_age_years** – Minimum age in years of people to collect data on
- **max_age_years** – Maximum age in years of people to collect data on
- **must_have_ip_key_value** – a "Key:Value" pair that the individual must have in order to be included. Empty string means don't look at IPs (individual properties)
- **must_have_intervention** – the name of the an intervention that the person must have in order to be included. Empty string means don't look at the interventions
- **filename_suffix** – augments the filename of the report. If multiple reports are being generated, this allows you to distinguish among the multiple reports

Returns if task is not set, returns the configured reporter, otherwise returns nothing

class emodpy_malaria.reporters.builtin.**ReportVectorGenetics**(*class_name: typing.Optional[str] = None*, *parameters: dict = <factory>*, *Enabled: bool = True*, *Pretty_Format: bool = True*)

Bases: emodpy.reporters.base.BuiltInReporter

The vector genetics report is a CSV-formatted report (ReportVectorGenetics.csv) that collects information on how many vectors of each genome/allele combination exist at each time, node, and vector state. Information can only be collected on one species per report.

config(*config_builder*, *manifest*)

parameters: dict

class emodpy_malaria.reporters.builtin.**ReportInfectionStatsMalaria**(*class_name: typing.Optional[str] = None*, *parameters: dict = <factory>*, *Enabled: bool = True*, *Pretty_Format: bool = True*)

Bases: emodpy.reporters.base.BuiltInReporter

config(*config_builder*, *manifest*)

parameters: dict

class emodpy_malaria.reporters.builtin.**ReportVectorStats**(*class_name: typing.Optional[str] = None*, *parameters: dict = <factory>*, *Enabled: bool = True*, *Pretty_Format: bool = True*)

Bases: emodpy.reporters.base.BuiltInReporter

The vector statistics report is a CSV-formatted report (ReportVectorStats.csv) that provides detailed life-cycle data on the vectors in the simulation. The report is stratified by time, node ID, and (optionally) species.

config(*config_builder*, *manifest*)

parameters: dict

class emodpy_malaria.reporters.builtin.**MalariaSummaryReport**(*class_name: typing.Optional[str] = None*, *parameters: dict = <factory>*, *Enabled: bool = True*, *Pretty_Format: bool = True*)

Bases: emodpy.reporters.base.BuiltInReporter

The population-level malaria summary report is a JSON-formatted report (MalariaSummaryReport.json) that provides a summary of malaria data across the population. The data are grouped into different bins such as age, parasitemia, and infectiousness.

config(*config_builder*, *manifest*)

parameters: dict

class emodpy_malaria.reporters.builtin.**MalariaPatientJSONReport**(*class_name: typing.Optional[str] = None*, *parameters: dict = <factory>*, *Enabled: bool = True*, *Pretty_Format: bool = True*)

Bases: emodpy.reporters.base.BuiltInReporter

The malaria patient data report is a JSON-formatted report (MalariaPatientReport.json) that provides medical data for each individual on each day of the simulation. For example, for a specified number of time steps, each "patient" has information collected on the temperature of their fever, their parasite counts, treatments they received, and other relevant data.

config(*config_builder*, *manifest*)

parameters: dict

class emodpy_malaria.reporters.builtin.**ReportSimpleMalariaTransmissionJSON**(*class_name: typing.Optional[str] = None*, *parameters: dict = <factory>*, *Enabled: bool = True*, *Pretty_Format: bool = True*)

Bases: emodpy.reporters.base.BuiltInReporter

The simple malaria transmission report (ReportSimpleMalariaTransmissionJSON.json) is a JSON-formatted report that provides data on malaria transmission, by tracking who transmitted malaria to whom. The report can only be used when the simulation setup parameter **Malaria_Model** is set to MALARIA_MECHANISTIC_MODEL_WITH_CO_TRANSMISSION. This report is typically used as input to the GenEpi model.

config(*config_builder*, *manifest*)

parameters: dict

class emodpy_malaria.reporters.builtin.**ReportMalariaFiltered**(*class_name: typing.Optional[str] = None*, *parameters: dict = <factory>*, *Enabled: bool = True*, *Pretty_Format: bool = True*)

Bases: emodpy.reporters.base.BuiltInReporter

The malaria filtered report (ReportMalariaFiltered.json) is the same as the default InsetChart report, but provides filtering options to enable the user to select the data to be displayed for each time step or for each node. See InsetChart for more information about InsetChart.json.

config(*config_builder*, *manifest*)

parameters: dict

class emodpy_malaria.reporters.builtin.**SpatialReportMalariaFiltered**(*class_name: typing.Optional[str] = None*, *parameters: dict = <factory>*, *Enabled: bool = True*, *Pretty_Format: bool = True*)

Bases: emodpy.reporters.base.BuiltInReporter

The filtered malaria spatial report (SpatialReportMalariaFiltered.bin) provides spatial information on malaria simulations and allows for filtering the data and collection over different intervals. This report is similar to the Spatial output report but allows for data collection and filtering over different intervals using the Start_Day and a Reporting_Interval parameters

config(*config_builder*, *manifest*)

parameters: dict

class emodpy_malaria.reporters.builtin.**ReportMalariaFilteredIntraHost**(*class_name: typing.Optional[str] = None*, *parameters: dict = <factory>*, *Enabled: bool = True*, *Pretty_Format: bool = True*)

Bases: emodpy.reporters.base.BuiltInReporter

The filtered malaria spatial report (ReportMalariaFilteredIntraHost.bin) provides TBD

config(*config_builder*, *manifest*)

parameters: dict

class emodpy_malaria.reporters.builtin.**ReportEventCounter**(*class_name: typing.Optional[str] = None*, *parameters: dict = <factory>*, *Enabled: bool = True*, *Pretty_Format: bool = True*)

Bases: emodpy.reporters.base.BuiltInReporter

The event counter report is a JSON-formatted file (ReportEventCounter.json) that keeps track of how many of each event types occurs during a time step. The report produced is similar to the InsetChart.json channel report, where there is one channel for each event defined in the configuration file (config.json).

config(*config_builder*, *manifest*)

parameters: dict

class emodpy_malaria.reporters.builtin.**MalariaSqlReport**(*class_name: typing.Optional[str] = None*, *parameters: dict = <factory>*, *Enabled: bool = True*, *Pretty_Format: bool = True*)

Bases: emodpy.reporters.base.BuiltInReporter

The MalariaSQL report outputs epidemiological and transmission data. Because of the quantity and complexity of the data, the report output is a multi-table SQLite relational database (see https://sqlitebrowser.org/). Use the configuration parameters to manage the size of the database.

config(*config_builder*, *manifest*)

parameters: dict

class emodpy_malaria.reporters.builtin.**VectorHabitatReport**(*class_name: typing.Optional[str] = None*, *parameters: dict = <factory>*, *Enabled: bool = True*, *Pretty_Format: bool = True*)

Bases: emodpy.reporters.base.BuiltInReporter

The vector habitat report is a JSON-formatted file (VectorHabitatReport.json) containing habitat data for each vector species included in the simulation. It focuses on statistics relevant to mosquito developmental stages (e.g. eggs and larvae), such as egg capacity and larval crowding.

config(*config_builder*, *manifest*)

parameters: dict

class emodpy_malaria.reporters.builtin.**MalariaImmunityReport**(*class_name: typing.Optional[str] = None*, *parameters: dict = <factory>*, *Enabled: bool = True*, *Pretty_Format: bool = True*)

Bases: emodpy.reporters.base.BuiltInReporter

The malaria immunity report is a JSON-formatted file (MalariaImmunityReport.json) that provides statistics for several antibody types for specified age bins over a specified reporting duration. Specifically, the report tracks the average and standard deviation in the fraction of observed antibodies for merozoite surface protein (MSP), Plasmodium falciparum erythrocyte membrane protein 1 (PfEMP1), and non-specific (and less immunogenic) minor surface epitopes. The total possible is determined by parameters Falciparum_MSP_Variants, Falciparum_PfEMP1_Variants, and Falciparum_Nonspecific_Types. The greater the fraction, the more antibodies the individual has against possible new infections. The smaller the fraction, the more naïve the individual's immune system is to malaria.

config(*config_builder*, *manifest*)

parameters: dict

class emodpy_malaria.reporters.builtin.**MalariaSurveyJSONAnalyzer**(*class_name: str = None*, *parameters: dict = <factory>*, *Enabled: bool = True*, *Pretty_Format: bool = True*)

Bases: emodpy.reporters.base.BuiltInReporter

config(*config_builder*, *manifest*)

parameters: dict

class emodpy_malaria.reporters.builtin.**ReportDrugStatus**(*class_name: typing.Optional[str] = None*, *parameters: dict = <factory>*, *Enabled: bool = True*, *Pretty_Format: bool = True*)

Bases: emodpy.reporters.base.BuiltInReporter

The drug status report provides status information on the drugs that an individual has taken or is waiting to take. Because the report provides information for each drug, for each individual, and for each time step, you may want to use the Start_Day and End_Day parameters to limit the size the output file.

config(*config_builder*, *manifest*)

parameters: dict

class emodpy_malaria.reporters.builtin.**ReportHumanMigrationTracking**(*class_name: typing.Optional[str] = None*, *parameters: dict = <factory>*, *Enabled: bool = True*, *Pretty_Format: bool = True*)

Bases: emodpy.reporters.base.BuiltInReporter

The human migration tracking report is a CSV-formatted report (ReportHumanMigrationTracking.csv) that provides details about human travel during simulations. The report provides one line for each surviving individual who migrates during the simulation.

config(*config_builder*, *manifest*)

parameters: dict

class emodpy_malaria.reporters.builtin.**ReportNodeDemographics**(*class_name: typing.Optional[str] = None*, *parameters: dict = <factory>*, *Enabled: bool = True*, *Pretty_Format: bool = True*)

Bases: emodpy.reporters.base.BuiltInReporter

The node demographics report is a CSV-formatted report (ReportNodeDemographics.csv) that provides population information stratified by node. For each time step, the report collects data on each node and age bin.

config(*config_builder*, *manifest*)

parameters: dict

class emodpy_malaria.reporters.builtin.**ReportNodeDemographicsMalaria**(*class_name: typing.Optional[str] = None*, *parameters: dict = <factory>*, *Enabled: bool = True*, *Pretty_Format: bool = True*)

Bases: emodpy.reporters.base.BuiltInReporter

This report extends the data collected in the ReportNodeDemographics by adding data about the number of infections with specific barcodes. The malaria node demographics genetics report does not include columns for Genome_Markers because this report assumes that the simulation setup parameter Malaria_Model is set to MALARIA_MECHANISTIC_MODEL_WITH_PARASITE_GENETICS.

Note: If you need detailed data on the infections with different barcodes, use the MalariaSqlReport. That report contains data on all barcodes, without specifying what they are.

config(*config_builder*, *manifest*)

parameters: dict

class emodpy_malaria.reporters.builtin.**ReportNodeDemographicsMalariaGenetics**(*class_name: typing.Optional[str] = None*, *parameters: dict = <factory>*, *Enabled: bool = True*, *Pretty_Format: bool = True*)

Bases: emodpy.reporters.base.BuiltInReporter

This report extends the data collected in the ReportNodeDemographics by adding data about the number of infections with specific barcodes. The malaria node demographics genetics report does not include columns for Genome_Markers because this report assumes that the simulation setup parameter Malaria_Model is set to MALARIA_MECHANISTIC_MODEL_WITH_PARASITE_GENETICS.

Note: If you need detailed data on the infections with different barcodes, use the MalariaSqlReport. That report contains data on all barcodes, without specifying what they are.

config(*config_builder*, *manifest*)

parameters: dict

class emodpy_malaria.reporters.builtin.**ReportVectorMigration**(*class_name: typing.Optional[str] = None*, *parameters: dict = <factory>*, *Enabled: bool = True*, *Pretty_Format: bool = True*)

Bases: emodpy.reporters.base.BuiltInReporter

This report provides detailed information on where and when vectors are migrating. Because there can be one line for each migrating vector, you may want to use the Start_Day and End_Day parameters to limit the size the output file.

config(*config_builder*, *manifest*)

parameters: dict

class emodpy_malaria.reporters.builtin.**ReportVectorStatsMalariaGenetics**(*class_name: typing.Optional[str] = None*, *parameters: dict = <factory>*, *Enabled: bool = True*, *Pretty_Format: bool = True*)

Bases: emodpy.reporters.base.BuiltInReporter

This report extends the data collected in the ReportVectorStats by adding data about the number of infections with specific barcodes. The malaria node demographics genetics report does not include columns for Genome_Markers because this report assumes that the simulation setup parameter Malaria_Model is set to MALARIA_MECHANISTIC_MODEL_WITH_PARASITE_GENETICS.

config(*config_builder*, *manifest*)

parameters: dict

class emodpy_malaria.reporters.builtin.**ReportInterventionPopAvg**(*class_name: typing.Optional[str] = None*, *parameters: dict = <factory>*, *Enabled: bool = True*, *Pretty_Format: bool = True*)

Bases: emodpy.reporters.base.BuiltInReporter

ReportInterventionPopAvg is a CSV-formatted report that gives population average data on the usage of interventions. It provides data on the fraction of people or nodes that have an intervention as well as averages on the intervention's efficacy. For each persistent intervention that has been distributed to a node or person, the report provides one line in the CSV for each intervention used in that node. Since node-level intervention (usually vector control) can only have one per node, the data will be for that one intervention. The individual-level interventions will have data for the people in that node.

config(*config_builder*, *manifest*)

parameters: dict

emodpy_malaria.weather package

weather is a module providing features for working with EMOD weather files.

Main Features

Here are main features:

- Generate EMOD weather files locally from a csv file.
- Generate EMOD weather files using COMPS SSMT weather service.
- Convert existing EMOD weather files to csv file or dataframes.
- Programmatic access to EMOD weather files via weather object model.

emodpy_malaria.weather.**generate_weather**(*platform: Union[str, idmtools_platform_comps.comps_platform.COMPSPlatform]*, *site_file: Union[str, pathlib.Path]*, *start_date: int*, *end_date: Optional[int] = None*, *node_column: str = 'nodes'*, *lat_column: str = 'lat'*, *lon_column: str = 'lon'*, *id_reference: Optional[str] = None*, *request_name: str = ''*, *local_dir: Optional[Union[str, pathlib.Path]] = None*, *data_source: Optional[str] = None*, *force: bool = False*) → *emodpy_malaria.weather.weather_request.WeatherRequest*

Generate weather files by submitting a request and downloading generated weather files to a specified dir.

Parameters

- **platform** – Platform name (like "Bayesian") or COMPSPlatform object, where the work item will run.
- **site_file** – CSV (.csv) or demographics (.json) file containing a set of sites (points) defined with lat/lon. CSV file must contain columns for: EMOD node ids (node), latitude (lat) and longitude (lon). Demographics file must match EMOD demographics file schema.
- **start_date** – Start date, in formats: year (2018), year and day-of-year (2018001) or date (20180101)
- **end_date** – (Optional) End date, in formats: year (2018), year and day-of-year (2018365) or date (20181231)
- **node_column** – (Optional) Name of a column containing EMOD node ids. The default is "nodes".
- **lat_column** – (Optional) Name of a column containing site (point) latitude. The default is "lat".
- **lon_column** – (Optional) Name of a column containing site (point) longitude. The default is "lon".
- **id_reference** – (Optional) Value of weather metadata IdReference attribute. The default is "Default".
- **request_name** – (Optional) Name to be used for the weather SSMT work item.
- **local_dir** – (Optional) Local dir where files will be downloaded.
- **data_source** – (Optional) SSMT data source to be used.
- **force** – (Optional) Flag ensuring a new weather request is submitted, even if weather files exist in "local_dir".

  Example:

```
wr: WeatherRequest = generate_weather(platform="Bayesian",
                                      site_file="path/to/sites.csv",
                                      start_date=2015,
                                      end_date=2016,
                                      node_column="id",
                                      local_dir="path/to/weather_dir
↪")
```

Returns WeatherRequest object. Can be used to access asset collection id or a local dir (if not given as) argument or a download report.

emodpy_malaria.weather.**csv_to_weather**(*csv_data: Union[str, pathlib.Path, pandas.core.frame.DataFrame]*, *node_column: str = 'nodes'*, *step_column: str = 'steps'*, *weather_columns: Optional[Dict[*emodpy_malaria.weather.weather_variable.WeatherVariable*, str]] = None*, *attributes: Optional[*emodpy_malaria.weather.weather_metadata.WeatherAttributes*] = None*, *weather_dir: Optional[Union[str, pathlib.Path]] = None*, *weather_file_names: Optional[Dict[*emodpy_malaria.weather.weather_variable.WeatherVariable*, str]] = None*) → *emodpy_malaria.weather.weather_set.WeatherSet*

Convert a dataframe or csv file, containing node, step and weather columns, into a weather set and corresponding weather files, if weather dir is specified.

Parameters

- **csv_data** – Dataframe or a csv file path, containing weather data.
- **node_column** – (Optional) Column containing node ids. The default is "nodes". The default is "nodes".
- **step_column** – (Optional) Column containing node index for weather time series values. The default is "steps".
- **weather_columns** – (Optional) Dictionary of weather variables (keys) and weather column names (values). Defaults are WeatherVariables values are used: "airtemp", "humidity", "rainfall", "landtemp".
- **attributes** – (Optional) Weather attribute object containing metadata for WeatherMetadata object.
- **weather_dir** – (Optional) Directory where weather files are stored. If not specified files are not created.
- **weather_file_names** – (Optional) Dictionary of weather variables (keys) and weather .bin file names (values).

Example:

```
wa = WeatherAttributes(start_year=2001, end_year=2010)
ws = csv_to_weather(csv_data="path/to/data.csv", attributes=wa,␣
↪weather_dir="path/to/weather_dir")
```

Returns WeatherSet object.

emodpy_malaria.weather.**weather_to_csv**(*weather_dir: Union[str, pathlib.Path]*, *weather_file_prefix: str = ''*, *weather_file_names: Optional[Dict[*emodpy_malaria.weather.weather_variable.WeatherVariable*, str]] = None*, *csv_file: Optional[Union[str, pathlib.Path]] = None*, *node_column: str = 'nodes'*, *step_column: str = 'steps'*, *weather_columns: Optional[Dict[*emodpy_malaria.weather.weather_variable.WeatherVariable*, str]] = None*) → Tuple[pandas.core.frame.DataFrame, *emodpy_malaria.weather.weather_metadata.WeatherAttributes*]

Convert weather files into a dataframe and a .csv file, if csv file path is specified.

Parameters

- **weather_dir** – Local dir containing weather files.
- **weather_file_prefix** – (Optional) Weather files prefix, e.g. "dtk_15arcmin_"

- **weather_file_names** – (Optional) Dictionary of weather variables (keys) and weather .bin file names (values).
- **csv_file** – (Optional) The path of a csv file to be generated. If not specified csv file is not created.
- **node_column** – (Optional) Column containing node ids. The default is "nodes".
- **step_column** – (Optional) Column containing node index for weather time series values. The default is "steps".
- **weather_columns** –

  (Optional) Dictionary of weather variables (keys) and weather column names (values). Defaults are WeatherVariables values are used: "airtemp", "humidity", "rainfall", "landtemp".

  Example:

```
df, attributes = weather_to_csv(weather_dir="path/to/weather_dir")
```

Returns Dataframe and weather attributes objects.

Submodules

emodpy_malaria.weather.data_sources module

Data sources metadata, a temporarily snapshot of data sources config, in the future to be removed with a weather service API offers this metadata.

emodpy_malaria.weather.weather_data module

Weather data module implementing functionality for working with binary weather files (.bin.json).

class emodpy_malaria.weather.weather_data.**WeatherData**(*data: numpy.ndarray*, *metadata: Optional[*emodpy_malaria.weather.weather_metadata.WeatherMeta = *None*)

Bases: object

Functionality for working with binary weather files (.bin.json).

validate()
Validate data and metadata relationship.

property metadata: *emodpy_malaria.weather.weather_metadata.WeatherMetadata*
Metadata property, exposing weather metadata object.

property data: **numpy.ndarray**
Raw data, reshaped in one row per node weather time series.

classmethod from_dict(*node_series: Dict[int, Union[np.ndarray[np.float32], List[float]]]*, *same_nodes: Dict[int, List[int]] = None*, *attributes:* WeatherAttributes *= None*) → *WeatherData*
Creates a WeatherData object from a dictionary mapping nodes and node weather time series. The method identifies unique node weather time series and produces a corresponding node-offset dictionary.

Parameters

- **node_series** – Dictionary with node ids as keys and weather time series as values (don't have to be unique).
- **same_nodes** – (Optional) Dictionary, mapping nodes from 'node_series' dictionary to additional nodes which series are the same. Keys are node ids, values are lists of node ids.
- **attributes** – (Optional) Attributes used to initiate weather metadata. If not provided, defaults are used.

Returns WeatherData object.

to_dict(*only_unique_series=False*, *copy_data: bool = True*) → Dict[int, np.ndarray[np.float32]]

Create a node-to-series dictionary from the current object. This method can be used to edit weather data.

Parameters

- **only_unique_series** – (Optional) A flag controlling whether the output dictionary will contain series for all nodes (if set to true) or only unique series.
- **copy_data** – (Optional) Flag indicating whether to copy data numpy array to prevent unintentional changes.

Returns A dictionary with node ids and keys and node weather time series as values.

classmethod from_csv(*file_path: Union[str, pathlib.Path]*, *info: Optional[*emodpy_malaria.weather.weather_data.DataFrameInfo*] = None*, *attributes: Optional[*emodpy_malaria.weather.weather_metadata.WeatherAttributes*] = None*) → *emodpy_malaria.weather.weather_data.WeatherData*

Creates a WeatherData object from a csv file. Used for creating or editing weather files. The method identifies unique node weather time series and produces a corresponding node-offset dictionary.

Parameters

- **file_path** – The csv file path from which weather data is loaded (expected columns: node, step, value).
- **info** – (Optional) Dataframe info object describing dataframe columns and content.
- **attributes** – (Optional) Attributes used to initiate weather metadata. If not provided, defaults are used.

Returns WeatherData object.

to_csv(*file_path: Union[str, pathlib.Path]*, *info: Optional[*emodpy_malaria.weather.weather_data.DataFrameInfo*] = None*) → pandas.core.frame.DataFrame

Creates a csv file and stores node ids, time steps and weather node weather time series as separate columns.

Parameters

- **file_path** – The csv file path into which weather data will be stored.
- **info** – (Optional) Dataframe info object describing dataframe columns and content.

Returns Dataframe created as an intermediate object used to save data to a csv file.

classmethod from_dataframe(*df: pd.DateFrame*, *info:* DataFrameInfo *= None*, *attributes:* WeatherAttributes *= None*) → *WeatherData*

Creates WeatherData object from the Pandas dataframe. The dataframe is expected to contain node ids, time steps and weather node weather time series as separate columns.

Parameters

- **df** – Dataframe containing nodes and weather time series (expected columns: node, step, value).
- **info** – (Optional) Dataframe info object describing dataframe columns and content.
- **attributes** – (Optional) Attributes used to initiate weather metadata. If not provided, defaults are used.

Returns WeatherData object.

to_dataframe(*info: Optional[emodpy_malaria.weather.weather_data.DataFrameInfo] = None*) → pandas.core.frame.DataFrame

Creates a dataframe containing node ids, time steps and weather time series as separate columns.

Parameters **info** – (Optional) Dataframe info object describing dataframe columns and content.

Returns Dataframe containing node ids and weather time series.

classmethod from_file(*file_path: Union[str, pathlib.Path]*) → *emodpy_malaria.weather.weather_data.WeatherData*

Create WeatherData object by reading weather data from binary (.bin) and metadata (.bin.json) files.

Parameters **file_path** – The weather binary (.bin) file path. The metadata file path is constructed by appending ".json".

Returns WeatherData object.

to_file(*file_path: Union[str, pathlib.Path]*) → NoReturn

Create weather binary (.bin) and metadata (.json) files, containing weather data and metadata.

Parameters **file_path** – The weather binary (.bin) file path. The metadata file path is constructed by adding ".json".

Returns None.

class emodpy_malaria.weather.weather_data.**DataFrameInfo**(*node_column: Optional[str] = None*, *step_column: Optional[str] = None*, *value_column: Optional[str] = None*, *only_unique_series: bool = False*)

Bases: object

The object containing info about dataframe columns and content. Used to pass dataframe info between methods working with weather dataframes.

property node_column

property step_column

property value_column

classmethod detect_columns(*df*, *column_candidates: Optional[Dict[str, List[str]]] = None*) → *emodpy_malaria.weather.weather_data.DataFrameInfo*

Auto-detect required column names (nodes, time-steps and weather time series) for the DataFrameInfo object.

Parameters

- **df** – The dataframe containing nodes, time-steps and weather time series.
- **column_candidates** – (Optional) Dictionary of candidate column names to be used instead of defaults.

Returns DataFrameInfo object with detected column names.

emodpy_malaria.weather.weather_metadata module

Weather metadata module, implementing functionality for working with weather metadata files (.bin.json).

class emodpy_malaria.weather.weather_metadata.**WeatherAttributes**(*attributes_dict: Optional[Dict[str, Union[str, int, float]]] = None*, *reference: Optional[str] = None*, *resolution: Optional[str] = None*, *provenance: Optional[str] = None*, *update_freq: Optional[str] = None*, *start_year: Optional[int] = None*, *end_year: Optional[int] = None*, *start_doy: Optional[int] = None*, *lat_min: Optional[float] = None*, *lat_max: Optional[float] = None*, *lon_min: Optional[float] = None*, *lon_max: Optional[float] = None*, *tool: Optional[str] = None*, *author: Optional[str] = None*, *schema_version: Optional[str] = None*)

Bases: object

Weather attributes containing metadata used to construct the "metadata" element and stored in .bin.json file. Used to initiate weather metadata object. If no metadata is specified, defaults are used. If metadata dict is specified, that dictionary is used as is, without setting any defaults.

property attributes_dict: Dict[str, Union[str, int, float]]

property tool: str
Tool name used to prepare weather files.

property date_created: str
Date weather files are created.

property author: str
Author of weather files.

property id_reference: str
Id reference value used to semantically bind weather files.

property update_resolution: str
Data update frequency (e.g. monthly, yearly).

property data_years: str
Data years range describing the temporal coverage.

property provenance: str
Data provenance, describing the data source and how the data was obtained.

property spatial_resolution: str
Weather data Spatial resolution (e.g., 30arcsec or 1km)

classmethod format_create_date(*created: datetime.datetime*) → str

classmethod metadata_defaults_dict() → Dict[str, Union[str, int, float]]
Metadata defaults dictionary.

classmethod required_metadata_defaults_dict(*exclude_keys: Optional[List[str]] = None*) → Dict[str, Union[str, int, float]]
Dictionary of required metadata defaults.

update(*value: Dict[str, Union[int, str]]*)
Update metadata dictionary.

validate() → NoReturn
Validate metadata contains requires argument.

class emodpy_malaria.weather.weather_metadata.**WeatherMetadata**(*node_ids: Union[List[int], Dict[int, int]]*, *series_len: Optional[int] = None*, *attributes: Optional[Union[*emodpy_malaria.weather.weather_metad
emodpy_malaria.weather.weather_metadata.WeatherA
Dict[str, Union[str, int, float]]]] = None)
Bases: *emodpy_malaria.weather.weather_metadata.WeatherAttributes*

Weather metadata containing weather data attributes, counts and node offsets. Used to initiate WeatherData, expose metadata programmatically and create weather metadata files (.bin.json).

validate()
Validate metadata object node-related counts. Relies on inherited validation of metadata attributes.

property attributes: *emodpy_malaria.weather.weather_metadata.WeatherAttributes*
Cast back into WeatherAttributes.

property datavalue_count: int
Number of data values in each weather time series (must be > 0).

property series_len: int
Number of data values in each timeseries, should be > 0.

property series_count: int
The number of weather time series (expected based on metadata), corresponding to the number of offsets.

property series_unique_count
The number of unique weather time series (expected based on metadata), based on offsets.

property total_value_count: int
The total count of all values, in all weather time series (expected based on metadata).

property nodes: List[int]
The list of nodes (node ids) in the node-offset dictionary.

property node_count: int
The number of node in the node-offset dictionary.

property node_offset_str: str
The node offset string, as it will appear in the weather metadata file (.bin.json).

property node_offsets: Dict[int, int]
Node-offset dictionary, mapping node ids (keys) to node offsets (values).

property offset_nodes: Dict[int, List[int]]
The offset-nodes dictionary, grouping nodes (values) by offset (key). Used to find unique series.

to_file(*file_path: Union[str, pathlib.Path]*) → NoReturn
Save weather metadata object as weather metadata file (.bin.json).

Parameters **file_path** – The path of the output weather metadata file.

Returns None

classmethod from_file(*file_path: Union[str, pathlib.Path]*) → *emodpy_malaria.weather.weather_metadata.WeatherMetadata*

Read weather metadata file into a weather metadata object.

emodpy_malaria.weather.weather_request module

Weather service methods for submitting and working with weather data requests.

class emodpy_malaria.weather.weather_request.**WeatherArgs**(*site_file: Union[str, pathlib.Path]*, *start_date: Union[int, str, datetime.datetime]*, *end_date: Optional[Union[int, str, datetime.datetime]] = None*, *node_column: str = 'node'*, *lat_column: str = 'lat'*, *lon_column: str = 'lon'*, *id_reference: str = 'Default'*)

Bases: object

Arguments defining weather request space and time scope.

validate()

Validates: site file (exists, is readable, and it contains specified columns) and dates range.

class emodpy_malaria.weather.weather_request.**RequestReport**

Bases: object

Specifies an object containing weather request operational reports.

download: Dict[str, List[str]] = None

class emodpy_malaria.weather.weather_request.**DataSource**(*name: Optional[str] = None*)

Bases: object

property name: str

Data source name property.

property file_prefix: str

Weather file prefix based on the current data source resolution.

property weather_variables: List[*emodpy_malaria.weather.weather_variable.WeatherVariable*]

List of weather variables supported by the current data source.

class emodpy_malaria.weather.weather_request.**WeatherRequest**(*platform: Union[str, idmtools_platform_comps.comps_platform.COMPSPlatform*, *local_dir: Optional[str] = None*, *data_source: Optional[str] = None*, *is_staging: Optional[bool] = None*)

Bases: object

Functionality for requesting and downloading weather files. Leverages idmtools API for COMPS SSMT.

property data_id: str

Expose asset collection id as interface data id property.

property local_dir: str

Local dir to/from where weather files will be downloaded.

property files: List[str]
List expected weather file paths.

property files_exist: bool
Returns True if all expected weather files exist in the local dir.

property report: *emodpy_malaria.weather.weather_request.RequestReport*
Returns report object.

generate(*weather_args:* emodpy_malaria.weather.weather_request.WeatherArgs, *request_name: Optional[str] = None*, *force: bool = False*) → Optional[*emodpy_malaria.weather.weather_request.WeatherRequest*]
Submits the weather request and when data is ready sets the data_id property.

Parameters

- **weather_args** – Arguments defining space and time scope and weather files' id reference.
- **request_name** – (Optional) Name to be used for the weather SSMT work item.
- **force** – (Optional) Force the download, even if target weather files already exist in the local dir.

Returns Returns this WeatherRequest object (to support method chaining).

download(*data_id: Optional[str] = None*, *local_dir: Optional[Union[str, pathlib.Path]] = None*, *force: bool = False*) → *emodpy_malaria.weather.weather_request.WeatherRequest*
Downloads weather files.

Parameters

- **data_id** – (Optional) Asset collection ID to be downloaded, even if not generated by this request.
- **local_dir** – (Optional) Local dir where files will be downloaded. If not specified a temp dir is created.
- **force** – (Optional) Force the download, even if target weather files already exist in the local dir.

Returns Returns this WeatherRequest object (to support method chaining).

emodpy_malaria.weather.weather_set module

Weather set module is implementing functionality for working with sets of weather files.

class emodpy_malaria.weather.weather_set.**WeatherSet**(*dir_path: Optional[Union[str, pathlib.Path]] = None*, *file_names: Optional[Dict[*emodpy_malaria.weather.weather_variable.WeatherVar *str]] = None*, *weather_columns: Optional[Dict[*emodpy_malaria.weather.weather_variable.WeatherVar *str]] = None*)

Bases: object

Representation of a set of weather files required by EMOD, for all or a subset of weather variables. Automate tasks for working with multiple weather files using WeatherData and WeatherMetadata objects. WeatherSet contains a dictionary of weather variables to WeatherData and WeatherMetadata objects. Supports: 1. Conversion from/to csv, dataframe (from_csv, to_csv, from_dataframe, to_dataframe) 2. Conversion from/to EMOD weather files, .bin and .bin.json (from_file, to_file)

keys()
Returns the list of WeatherVariables.

values() → List[*emodpy_malaria.weather.weather_data.WeatherData*]
Returns the list of WeatherData objects.

items() → None.dict.items
Returns an iterator for weather dictionary items.

property dir_path: *str*
Directory path containing weather files.

property file_names: **Dict[*emodpy_malaria.weather.weather_variable.WeatherVariable*, str]**
Dictionary of weather variables (keys) and weather file names (values).

property attributes: ***emodpy_malaria.weather.weather_metadata.WeatherAttributes***

property weather_variables:
List[*emodpy_malaria.weather.weather_variable.WeatherVariable*]
The list of weather variables the weather set covers.

property weather_columns:
Dict[*emodpy_malaria.weather.weather_variable.WeatherVariable*, str]
The list of weather columns.

classmethod from_dataframe(*df: pd.DateFrame*, *node_column: str = None*, *step_column: str = None*, *weather_columns: Dict[*WeatherVariable*, *str] = None*, *attributes:* WeatherAttributes *= None*) → *WeatherSet*
Initializes WeatherSet object from a dataframe containing weather time series. The dataframe must have node ids, step and weather columns.

Parameters

- **df** – Dataframe containing weather data.
- **node_column** – (Optional) Column containing node ids. The default is "nodes".
- **step_column** – (Optional) Column containing node index for weather time series values. The default is "steps".
- **weather_columns** – (Optional) Dictionary of weather variables (keys) and weather column names (values). Defaults are WeatherVariables values are used: "airtemp", "humidity", "rainfall", "landtemp".
- **attributes** – (Optional) Weather attribute object containing metadata for WeatherMetadata object.

Returns WeatherSet object.

classmethod from_csv(*file_path: Union[str, pathlib.Path]*, *node_column: Optional[str] = None*, *step_column: Optional[str] = None*, *weather_columns: Optional[Dict[*emodpy_malaria.weather.weather_variable.WeatherVariable*, *str]] = None*, *attributes:* *Optional[*emodpy_malaria.weather.weather_metadata.WeatherAttributes*] = None*) → *emodpy_malaria.weather.weather_set.WeatherSet*
Initializes WeatherSet object from a dataframe containing weather time series. The csv file must have node ids, step and weather columns.

Parameters

- **file_path** – The csv file path.

- **node_column** – (Optional) Column containing node ids. The default is "nodes".
- **step_column** – (Optional) Column containing node index for weather time series values. The default is "steps".
- **weather_columns** – (Optional) Dictionary of weather variables (keys) and weather column names (values). Defaults are WeatherVariables values are used: "airtemp", "humidity", "rainfall", "landtemp".
- **attributes** – (Optional) The weather attribute object containing metadata for WeatherMetadata object.

Returns WeatherSet object.

to_dataframe(*node_column: Optional[str] = None*, *step_column: Optional[str] = None*, *weather_columns: Optional[Dict[*emodpy_malaria.weather.weather_variable.WeatherVariable*, str]] = None*) → pandas.core.frame.DataFrame

Creates a dataframe containing node ids, time steps and weather columns.

Parameters

- **node_column** – (Optional) Column containing node ids. The default is "nodes".
- **step_column** – (Optional) Column containing node index for weather time series values. The default is "steps".
- **weather_columns** – (Optional) Dictionary of weather variables (keys) and weather column names (values). Defaults are WeatherVariables values are used: "airtemp", "humidity", "rainfall", "landtemp".

Returns Dataframe containing node ids and weather time series.

to_csv(*file_path: Union[str, pathlib.Path]*, *node_column: Optional[str] = None*, *step_column: Optional[str] = None*, *weather_columns: Optional[Dict[*emodpy_malaria.weather.weather_variable.WeatherVariable*, str]] = None*) → pandas.core.frame.DataFrame

Creates a csv file containing node ids, time steps and weather columns.

Parameters

- **file_path** – The path of a csv file to be generated.
- **node_column** – (Optional) Column containing node ids. The default is "nodes".
- **step_column** – (Optional) Column containing node index for weather time series values. The default is "steps".
- **weather_columns** – (Optional) Dictionary of weather variables (keys) and weather column names (values). Defaults are WeatherVariables values are used: "airtemp", "humidity", "rainfall", "landtemp".

Returns Dataframe containing node ids and weather time series, used to create the csv file.

classmethod from_files(*dir_path: Union[str, pathlib.Path]*, *prefix: str = ''*, *file_names: Optional[Dict[*emodpy_malaria.weather.weather_variable.WeatherVariable*, str]] = None*) → *emodpy_malaria.weather.weather_set.WeatherSet*

Instantiates WeatherSet from to weather files which paths are determined based on given arguments.

Parameters

- **dir_path** – Directory path containing weather files.
- **prefix** – Weather files prefix, e.g. "dtk_15arcmin_"
- **file_names** – Dictionary of weather variables (keys) and weather .bin file names (values).

Returns WeatherSet object.

to_files(*dir_path: Union[str, pathlib.Path]*, *file_names: Optional[Dict[*emodpy_malaria.weather.weather_variable.WeatherVariable*, str]] = None*) → NoReturn

Saves WeatherSet to weather files which paths are determined based on given arguments.

classmethod make_file_paths(*dir_path: Optional[Union[str, pathlib.Path]] = None*, *prefix: str = 'dtk_15arcmin_'*, *suffix: str = '{}_daily.bin'*, *weather_variables: Optional[List[*emodpy_malaria.weather.weather_variable.WeatherVariable*]] = None*, *weather_names: Optional[Dict[*emodpy_malaria.weather.weather_variable.WeatherVariable*, str]] = None*) → Dict[*emodpy_malaria.weather.weather_variable.WeatherVariable*, str]

Construct file paths using the weather directory path, file name prefix/suffix and weather variable names. The logic of this method is the same as of "Path.glob" method, with two adjustments, added to make its use more convenient for working with weather files: - if prefix/suffix are not specified, defaults are used (see method arguments). - if suffix doesn't end with ".bin" or "*", "*.bin" is added (since, otherwise, no matches can be found).

Parameters

- **dir_path** – (Optional) Directory path containing weather files.
- **prefix** – (Optional) Weather file name prefix, usually a fixed string like "dtk_".
- **suffix** – (Optional) Weather file name suffix, usually containing a weather variable name parameter like "*{tag}*.bin").
- **weather_names** – (Optional) Dictionary of weather variables (keys) and custom weather variable names (values).
- **weather_variables** – (Optional) Weather variables to be used in case custom weather names are not specified. In this case lowercase weather variable names are used, for example: AIR_TEMPERATURE -> air_temperature.

Returns

Dictionary of weather variables (keys) and weather file paths. For example, air temperature could be represented as: WeatherVariable.AIR_TEMPERATURE: "dtk_15arcmin_air_temperature_daily.bin"

classmethod select_weather_files(*dir_path: Union[str, pathlib.Path]*, *prefix: str = '*'*, *suffix: str = '*{}*.bin'*, *weather_variables: Optional[List[*emodpy_malaria.weather.weather_variable.WeatherVariable*]] = None*, *weather_names: Optional[Dict[*emodpy_malaria.weather.weather_variable.WeatherVariable*, str]] = None*) → Dict[*emodpy_malaria.weather.weather_variable.WeatherVariable*, str]

Select a set of weather files using the weather directory path, file name prefix/suffix and weather variable names. The logic of this method is the same as of "Path.glob" method, with two adjustments, added to make its use more convenient for working with weather files: - if prefix/suffix are not specified, defaults are used (see method arguments). - if suffix doesn't end with ".bin" or "*", "*.bin" is added (since, otherwise, no matches can be found).

Parameters

- **dir_path** – (Optional) Directory path containing weather files.
- **prefix** – (Optional) Weather file name prefix, usually a fixed string like "dtk_".

- **suffix** – (Optional) Weather file name suffix, usually containing a weather variable name parameter like "*{tag}*.bin").
- **weather_names** – (Optional) Dictionary of weather variables (keys) and custom weather variable names (values).
- **weather_variables** –

  (Optional) Weather variables to be used in case custom weather names are not specified. In this case lowercase weather variable names are used, for example: AIR_TEMPERATURE -> air_temperature.

  Returns: Dictionary of weather variables (keys) and weather file names. For example, WeatherVariable.AIR_TEMPERATURE: "dtk_15arcmin_air_temperature_daily.bin"

validate() → NoReturn
Validate WeatherSet object.

emodpy_malaria.weather.weather_utils module

The module for weather utility functions.

emodpy_malaria.weather.weather_utils.**invert_dict**(*in_dict: Dict*, *sort=False*, *single_value=False*) → Dict
Invert a dictionary by grouping keys by value. In the resulting dictionary keys are unique value from the original dictionary (including individual element of value lists) and values are lists of original dictionary keys. The function is used for grouping nodes by offset or weather time series hash, and determining unique series.

For example,

single_value = False (default) {1: [11, 22], 2: [22], 3: [11]} -> {11: [1, 3], 22: [1, 2]}

single_value = True {1: [11, 22], 2: [22], 3: [11]} -> {11: 1, 22: 1}

Parameters

- **in_dict** – Input dictionary to be inverted.
- **sort** – Sort the resulting dictionary by key and value.
- **single_value** – Only return a single representative "original key" (from in_dict) for each corresponding unique "original value" (from in_dict).

Returns

Inverted dictionary where (single_value = False): keys are "original values" and values are lists of corresponding "original keys" (single_value = True): keys are "original values" and values is a single representative "original key".

emodpy_malaria.weather.weather_utils.**hash_series**(*series: Iterable[numpy.float32]*) → int
Calculate a unique hash for each input series. Used for grouping nodes by weather time series.

Parameters **series** – The list or array of float values.

Returns Series hash value as int.

emodpy_malaria.weather.weather_utils.**save_json**(*content: Dict[str, str]*, *file_path: Union[str, pathlib.Path]*) → NoReturn
Save dictionary to a json file.

Parameters

- **content** – Content in the form of a dictionary.
- **file_path** – The path of the output weather metadata file.
- **Returns** – None

emodpy_malaria.weather.weather_utils.**make_path**(*dir_path: Union[str, pathlib.Path]*) → NoReturn
Make path directories.

emodpy_malaria.weather.weather_utils.**ymd**(*date_arg: datetime.datetime*) → str
Convert datetime into a string of format yyyymmdd.

emodpy_malaria.weather.weather_utils.**parse_date**(*date_arg: Union[int, str]*, *default_month: int*, *default_day: int*) → datetime.datetime

emodpy_malaria.weather.weather_utils.**validate_str_value**(*value: str*) → NoReturn
Assert a string value is not None or empty.

emodpy_malaria.weather.weather_variable module

Weather variable module implement functionality for working with weather variables.

class emodpy_malaria.weather.weather_variable.**WeatherVariable**(*value*)
Bases: enum.Enum

Weather variables required by EMOD.

AIR_TEMPERATURE = 'airtemp'

RELATIVE_HUMIDITY = 'humidity'

RAINFALL = 'rainfall'

LAND_TEMPERATURE = 'landtemp'

classmethod list(*exclude: Optional[Union[*emodpy_malaria.weather.weather_variable.WeatherVariable, *List[*emodpy_malaria.weather.weather_variable.WeatherVariable*]]] = None*) → List[*emodpy_malaria.weather.weather_variable.WeatherVariable*]
List of all weather variables or a subset if 'exclude' argument is used.

Parameters **exclude** – (Optional) List of weather variables to be excluded.

Returns List of all or a subset of weather variables.

classmethod validate_types(*value_dict: Dict[*emodpy_malaria.weather.weather_variable.WeatherVariable, *Any]*, *value_types: Optional[Union[Any, List[Any]]] = None*) → NoReturn
Validate dictionary keys are of type WeatherVariable and values are of specified type.

Parameters

- **value_dict** – Dictionary to be validated.
- **value_types** – Dictionary value types.

Returns None

2.1.2 Submodules

emodpy_malaria.malaria_config module

emodpy_malaria.malaria_config.**get_file_from_http**(*url*)
Get data files from simple http server.

emodpy_malaria.malaria_config.**set_team_defaults**(*config*, *manifest*)
Set configuration defaults using team-wide values, including drugs and vector species.

emodpy_malaria.malaria_config.**set_team_drug_params**(*config*, *manifest*)

emodpy_malaria.malaria_config.**set_parasite_genetics_params**(*config*, *manifest*, *var_gene_randomness_type: str = 'ALL_RANDOM'*)
Sets up the default parameters for parasite genetics simulations Malaria_Model = "MALARIA_MECHANISTIC_MODEL_WITH_PARASITE_GENETICS"

Parameters

- **config** –
- **manifest** – schema path container
- **var_gene_randomness_type** – possible values are "FIXED_NEIGHBORHOOD", "FIXED_MSP", "ALL_RANDOM" (default)

Returns configured config

emodpy_malaria.malaria_config.**get_drug_params**(*cb*, *drug_name*)

emodpy_malaria.malaria_config.**set_drug_param**(*config*, *drug_name: Optional[str] = None*, *parameter: Optional[str] = None*, *value: Optional[any] = None*)
Set a drug parameter, by passing in drug name, parameter and the parameter value. Added to facilitate adding drug Resistances, **Example**:

```
artemether_drug_resistance = [{
   "Drug_Resistant_String": "A",
   "PKPD_C50_Modifier": 2.0,
   "Max_IRBC_Kill_Modifier": 0.9}]
set_drug_param(cb, drug_name='Artemether', parameter="Resistances",␣
↪value=artemether_drug_resistance)
```

Parameters

- **config** –
- **drug_name** – The drug that has a parameter to set
- **parameter** – The parameter to set
- **value** – The new value to set

Returns Nothing or error if drug name is not found

emodpy_malaria.malaria_config.**add_drug_resistance**(*config*, *manifest*, *drugname: Optional[str] = None*, *drug_resistant_string: Optional[str] = None*, *pkpd_c50_modifier: float = 1.0*, *max_irbc_kill_modifier: float = 1.0*)
Adds drug resistances by drug name and parameters

Parameters

- **config** –
- **manifest** –
- **drugname** – name of the drug for which to assign resistances
- **drug_resistant_string** – A series of nucleotide base letters (A, C, G, T) that represent the drug resistant values at locations in the genome
- **pkpd_c50_modifier** – If the parasite has this genome marker, this value will be multiplied times the 'Drug_PKPD_C50' value of the drug. Genomes with multiple markers will be simply multiplied together
- **max_irbc_kill_modifier** – If the parasite has this genome marker, this value will be multiplied times the 'Max_Drug_IRBC_Kill' value of the drug. Genomes with multiple markers will be simply multiplied together

Returns configured config

emodpy_malaria.malaria_config.**set_species_param**(*config*, *species*, *parameter*, *value*, *overwrite=False*)
Pass through for vector version of function.

emodpy_malaria.malaria_config.**add_species**(*config*, *manifest*, *species_to_select*)
Pass through for vector version of function.

emodpy_malaria.malaria_config.**add_insecticide_resistance**(*config*, *manifest*, *insecticide_name: str = ''*, *species: str = ''*, *allele_combo: Optional[list] = None*, *blocking: float = 1.0*, *killing: float = 1.0*, *repelling: float = 1.0*, *larval_killing: float = 1.0*)
Pass through for vector version of function.

emodpy_malaria.malaria_config.**get_species_params**(*config*, *species: Optional[str] = None*)
Pass through for vector version of function.

emodpy_malaria.malaria_config.**set_max_larval_capacity**(*config*, *species_name: str*, *habitat_type: str*, *max_larval_capacity: int*)
Set the Max_Larval_Capacity for a given species and habitat. Effectively doing something like: simulation.task.config.parameters.Vector_Species_Params[i]["Habitats"][j]["Max_Larval_Capacity"] = max_larval_capacity where i is index of species_name and j is index of habitat_type.

Parameters

- **config** –
- **species_name** – string. Species_Name to target.
- **habitat_type** – enum. Habitat_Type to target.
- **max_larval_capacity** – integer. New value of Max_Larval_Capacity.

Returns Nothing.

emodpy_malaria.malaria_vector_species_params module

emodpy_malaria.malaria_vector_species_params.**species_params**(*manifest*, *species*)

Returns configured species parameters based on species name

Parameters

- **manifest** – file that contains path to the schema file
- **species** – species, configuration for which, we will be adding to the simulation.

Returns Configured species parameters

emodpy_malaria.vector_config module

emodpy_malaria.vector_config.**set_team_defaults**(*config*, *manifest*)
Set configuration defaults using team-wide values, including drugs and vector species.

Parameters

- **config** –
- **manifest** –

Returns configured config

emodpy_malaria.vector_config.**get_species_params**(*config*, *species: Optional[str] = None*)
Returns the species parameters dictionary with the matching species **Name**

Parameters

- **config** –
- **species** – Species to look up

Returns Dictionary of species parameters with the matching name

emodpy_malaria.vector_config.**set_species_param**(*config*, *species*, *parameter*, *value*, *overwrite=False*)

Sets a parameter value for a specific species. Raises value error if species not found

Parameters

- **config** –
- **species** – name of species for which to set the parameter
- **parameter** – parameter to set
- **value** – value to set the parameter to
- **overwrite** – if set to True and parameter is a list, overwrites the parameter with value, appends by default

Returns Nothing

emodpy_malaria.vector_config.**add_species**(*config*, *manifest*, *species_to_select*)
Adds species with preset parameters from 'malaria_vector_species_params.py', if species name not found - "gambiae" parameters are added and the new species name assigned.

Parameters

- **config** – schema-backed config smart dict

- **manifest** –
- **species_to_select** – a list of species or a name of a single species you'd like to set from malaria_vector_species_params.py

Returns configured config

emodpy_malaria.vector_config.**add_genes_and_alleles**(*config*, *manifest*, *species: Optional[str] = None*, *alleles: Optional[list] = None*)

Adds alleles to a species

Example:

```
"Genes": [
    {
        "Alleles": [
            {
                "Name": "X1",
                "Initial_Allele_Frequency": 0.5,
                "Is_Y_Chromosome": 0
            },
            {
                "Name": "X2",
                "Initial_Allele_Frequency": 0.25,
                "Is_Y_Chromosome": 0
            },
            {
                "Name": "Y1",
                "Initial_Allele_Frequency": 0.15,
                "Is_Y_Chromosome": 1
            },
            {
                "Name": "Y2",
                "Initial_Allele_Frequency": 0.1,
                "Is_Y_Chromosome": 1
            }
        ],
        "Is_Gender_Gene": 1,
        "Mutations": []
    }
]
```

Parameters

- **config** –
- **manifest** –
- **species** – species to which to assign the alleles
- **alleles** – List of tuples of (**Name**, **Initial_Allele_Frequency**, **Is_Y_Chromosome**) for a set of alleles or (**Name**, **Initial_Allele_Frequency**), 1/0 or True/False can be used for Is_Y_Chromosome, third parameter is assumed False (0). If the third parameter is set to 1 in any of the tuples, we assume, this is a gender gene. **Example**:

```
[("X1", 0.25), ("X2", 0.35), ("Y1", 0.15), ("Y2", 0.25)]
[("X1", 0.25, 0), ("X2", 0.35, 0), ("Y1", 0.15, 1), ("Y2", 0.25, 1)]
```

Returns configured config

emodpy_malaria.vector_config.**add_mutation**(*config*, *manifest*, *species*, *mutate_from*, *mutate_to*, *probability*)

Adds to **Mutations** parameter in a Gene which has the matching **Alleles**

Parameters

- **config** –
- **manifest** –
- **species** – Name of vector species to which we're adding mutations
- **mutate_from** – The allele in the gamete that could mutate
- **mutate_to** – The allele that this locus will change to during gamete generation
- **probability** – The probability that the allele will mutate from one allele to the other during the creation of the gametes

Returns configured config

emodpy_malaria.vector_config.**add_trait**(*config*, *manifest*, *species*, *allele_combo: Optional[list] = None*, *trait_modifiers: Optional[list] = None*)

Use this function to add traits as part of vector genetics configuration, the trait is assigned to the species' **Gene_To_Trait_Modifiers** parameter Should produce something like **Example**:

```
{
    "Allele_Combinations" : [
        [  "X",  "X" ],
        [ "a0", "a1" ]
    ],
    "Trait_Modifiers" : [
        {
            "Trait" : "FECUNDITY",
            "Modifier": 0.7
        }
    ]
}
```

Parameters

- **config** –
- **manifest** –
- **species** – **Name** of species for which to add this **Gene_To_Trait_Modifiers**
- **allele_combo** – List of lists, This defines a possible subset of allele pairs that a vector could have. Each pair are alleles from one gene. If the vector has this subset, then the associated traits will be adjusted. Order does not matter. '*' is allowed when only the occurrence of one allele is important. **Example**:

```
[[  "X",  "X" ], [ "a0", "a1" ]]
```

- **trait_modifiers** – List of tuples of (**Trait**, **Modifier**) for the *Allele_Combinations** **Example**:

```
[("FECUNDITY", 0.5), ("X_SHRED", 0.80)]
```

Returns configured config

emodpy_malaria.vector_config.**add_insecticide_resistance**(*config*, *manifest*, *insecticide_name: str = ''*, *species: str = ''*, *allele_combo: Optional[list] = None*, *blocking: float = 1.0*, *killing: float = 1.0*, *repelling: float = 1.0*, *larval_killing: float = 1.0*)

Use this function to add to the list of **Resistances** parameter for a specific insecticide Add each resistance separately. **Example**:

```
Insecticides = [
{
  "Name": "pyrethroid",
  "Resistances": [
    {
      "Allele_Combinations": [
      [
        "a1",
        "a1"
      ]
     ],
    "Blocking_Modifier": 1.0,
    "Killing_Modifier": 0.85,
    "Repelling_Modifier": 0.72,
    "Larval_Killing_Modifier": 0,
    "Species": "gambiae"
  }
 ]
},
{..}
```

Parameters

- **config** –
- **manifest** –
- **insecticide_name** – The name of the insecticide to which attach the resistance.
- **species** – Name of the species of vectors.
- **allele_combo** – List of combination of alleles that vectors must have in order to be resistant.
- **blocking** – The value used to modify (multiply) the blocking effectivity of an intervention.
- **killing** – The value used to modify (multiply) the killing effectivity of an intervention.
- **repelling** – The value used to modify (multiply) the repelling effectivity of an intervention.
- **larval_killing** – The value used to modify (multiply) the larval killing effectivity of an intervention.

Returns configured config

emodpy_malaria.vector_config.**add_species_drivers**(*config*, *manifest*, *species: Optional[str] = None*, *driving_allele: Optional[str] = None*, *driver_type: str = 'CLASSIC'*, *to_copy: Optional[str] = None*, *to_replace: Optional[str] = None*, *likelihood_list: Optional[list] = None*, *shredding_allele_required: Optional[str] = None*, *allele_to_shred: Optional[str] = None*, *allele_to_shred_to: Optional[str] = None*, *allele_shredding_fraction: Optional[float] = None*, *allele_to_shred_to_surviving_fraction: Optional[float] = None*)

Add a gene drive that propagates a particular set of alleles. Adds one **Alleles_Driven** item to the **Alleles_Driven** list, using 'driving_allele' as key if matching one already exists.

Example:

```
{
    "Driver_Type": "INTEGRAL_AUTONOMOUS",
    "Driving_Allele": "Ad",
    "Alleles_Driven": [
        {
            "Allele_To_Copy": "Ad",
            "Allele_To_Replace": "Aw",
            "Copy_To_Likelihood": [
                {
                    "Copy_To_Allele": "Aw",
                    "Likelihood": 0.1
                },
                {
                    "Copy_To_Allele": "Ad",
                    "Likelihood": 0.3
                },
                {
                    "Copy_To_Allele": "Am",
                    "Likelihood": 0.6
                }
            ]
        },
{
    "Driver_Type" : "X_SHRED",
    "Driving_Allele" : "Ad",
    "Driving_Allele_Params" : {
        "Allele_To_Copy"    : "Ad",
        "Allele_To_Replace" : "Aw",
        "Copy_To_Likelihood" : [
            {
                "Copy_To_Allele" : "Ad",
                "Likelihood" : 1.0
            },
            {
                "Copy_To_Allele" : "Aw",
                "Likelihood" : 0.0
            }
        ]
```

(continues on next page)

(continued from previous page)

```
    },
    "Shredding_Alleles" : {
        "Allele_Required"    : "Yw",
        "Allele_To_Shred"    : "Xw",
        "Allele_To_Shred_To" : "Xm",
        "Allele_Shredding_Fraction": 0.97,
        "Allele_To_Shred_To_Surviving_Fraction" : 0.05
    }
    ]
}
```

Parameters

- **config** –
- **manifest** –
- **species** – Name of the species for which we're setting the drivers
- **driving_allele** – This is the allele that is known as the driver
- **driver_type** – This indicates the type of driver. CLASSIC - The driver can only drive if the one gamete has the driving allele and the other has a specific allele to be replaced INTEGRAL_AUTONOMOUS - At least one of the gametes must have the driver. Alleles can still be driven if the driving allele is in both gametes or even if the driving allele cannot replace the allele in the other gamete X_SHRED, Y_SHRED - cannot be used in the same species during one simulation/realization. The driving_allele must exist at least once in the genome for shredding to occur. If there is only one, it can exist in either half of the genome. DAISY_CHAIN - can be used for drives that do not drive themselves but can be driven by another allele.
- **to_copy** – The main allele to be copied **Allele_To_Copy**
- **to_replace** – The allele that must exist and will be replaced by the copy **Allele_To_Replace**
- **likelihood_list** – A list of tuples in format: [(**Copy_To_Allele**, **Likelihood**),(),()] to assign to **Copy_To_Likelyhood** list
- **shredding_allele_required** – The genome must have this gender allele in order for shredding to occur. If the driver is X_SHRED, then the allele must be designated as a Y chromosome. If the driver is Y_SHRED, then the allele must NOT be designated as a Y chromosome
- **allele_to_shred** – The genome must have this gender allele in order for shredding to occur. If the driver is X_SHRED, then the allele must NOT be designated as a Y chromosome. If the driver is Y_SHRED, then the allele must be designated as a Y chromosome
- **allele_to_shred_to** – This is a gender allele that the 'shredding' will change the allele_to_shred into. It can be a temporary allele that never exists in the output or could be something that appears due to resistance/failures
- **allele_shredding_fraction** – This is the fraction of the alleles_to_Shred that will be converted to allele_to_shred_to. Values 0 to 1. If this value is less than 1, then some of the allele_to_shred will remain and be part of the gametes.
- **allele_to_shred_to_surviving_fraction** – A trait modifier will automatically generated for [Allele_To_Shred_To, *], the trait ADJUST_FERTILE_EGGS, and this value

as its modifier. Values 0 to 1. A value of 0 implies perfect shredding such that no allele_to_Shred_To survive in the eggs. A value of 1 means all of the 'shredded' alleles survive.

Returns configured config

emodpy_malaria.vector_config.**set_max_larval_capacity**(*config*, *species_name*, *habitat_type*, *max_larval_capacity*)

Set the Max_Larval_Capacity for a given species and habitat. Effectively doing something like: simulation.task.config.parameters.Vector_Species_Params[i]["Habitats"][j]["Max_Larval_Capacity"] = max_larval_capacity where i is index of species_name and j is index of habitat_type.

Parameters

- **config** –
- **species_name** – string. Species_Name to target.
- **habitat_type** – enum. Habitat_Type to target.
- **max_larval_capacity** – integer. New value of Max_Larval_Capacity.

Returns Nothing.

CHAPTER

THREE

FREQUENTLY ASKED QUESTIONS

As you get started with emodpy-malaria, you may have questions. The most common questions are answered below. The most common questions are answered below. For questions related to functionality in related packages, see the following documentation:

- Frequently asked questions for EMOD
- Frequently asked questions for idmtools
- Frequently asked questions for emod-api
- Frequently asked questions for emodpy

Contents

3.1 What are some of the key differences for people used to using dtk-tools?

1. Schema-Based. The creation of config and campaign files is entirely schema-based now. This means that you can only set parameters that the binary you are using recognizes. And parameter types and ranges are enforced at runtime.
2. Inter-File Dependencies Now Automatic. Before there were lots of parameters in the config that you had to set to correspond to settings in campaign or demographics files. That is no longer the case. We call these 'implicits'. For example, if you add a BirthRate to the demographics, the corresponding parameters in the config.json (Enable_Births) will get set automatically for you. As another example, when you create a campaign and specify various 'events' to be broadcast/published and/or listened/subscribed to, you no longer have to figure out which ones are built-in and which are ad-hoc. It does that for you and populates the Custom_Events param on your behalf.
3. Hierarchical Dependencies Now Automatic. If a parameter depends on another parameter, previously you had to set all the Enables in the dependency tree. Now they get set automatically for you. For example, if Enable_Birth is set (see above), Enable_Vital_Dynamics will be set for you.
4. No JSON manipulation. dtk-tools worked primarily through manipulation of JSON that made up the configuration files. You no longer need to have any knowledge of the internal representation of data in the DTK input files. All configuration should be done via Python functions.
5. Released and Installed Modules. We want users mostly using versioned released modules that have been pip installed, not git cloned, dev-installed code, except during development. The process of getting new code reviewed, tested, and getting the module versioned and released is intended to be smooth and efficient when everyone does their defined role. "Trust The Process and Do Your Job", as someone once said.
6. Blessed Binaries. In dtk-tools you would often BYOB – Bring Your Own Binary – but in emodpy, the idea is that the system pulls down the latest CI (Continuous Integration) build for your disease that passed all the tests. We very much want to noramlize the idea of doing research with versioned software that has come through our professional BVT processes.

3.2 Do I need to install a whole bunch of Python modules? Where do I get that list?

No. You should only have to install emodpy-malaria, and everything else should come as a dependency automatically. If you are starting from a workflow repo, there should be a requirements.txt file that lists a particular version of emodpy-malaria. Then 'pip install -r requirements.txt' is what you should expect to do.

3.3 Is there an easier way than typing –index-url with that long URL every time I use pip?

Yes. You should not be typing that every time. You should update the pip.ini (Windows) or pip.conf globally on your computer with that URL and never have to type it again.

3.4 How do I find the path or file to my pip.ini?

```
pip config -v list.
```

Choose the first or second one.

3.5 What if the pip.ini file (or pip folder) doesn't exist?

Go ahead and create it (at one of the locations specified by 'pip config -v list'). It's your computer.

3.6 What's the text I need to enter into my pip config file?

```
[global]
index-url = https://packages.idmod.org/api/pypi/pypi-production/simple
```

3.7 I was installing the Python modules and it told me I needed a C++ compiler?

This is because of a dependency in some versions of emod-api on lz4. lz4 usually comes as what's called a source package and it has to be compiled as part of the install. This requires a compiler, which doesn't come by default on Windows computers. At this point, if your pip.ini is set up properly, you should be able to get a pre-built lz4 package from our local Artifactory PIP server.

3.8 I was installing and got prompted for JFrog credentials.

You should not have to give credentials to use our JFrog/Artifactory/pip server. Credentials are required to get packages from the staging server. We don't expect end users to be accessing packages from staging, just prod, which is auth-free.

3.9 What if I really actually do want to install something from staging?

You need to specify '–index-url = https://<username>@idmod.org:<shh...password>@packages.idmod.org/api/pypi/pypi-staging/simple' and also provide your creds when prompted.

3.10 What version of Python should I be using?

At least Python 3.7.7. If you are installing a new version of Python, feel free to go all the way forward to a Python 3.9.x. 3.8 is probably the sweet spot of "known to work and still not considered old".

3.11 What if I want a particular version of emodpy-malaria?

```
pip install emodpy-malaria==1.2.3
```

Should get you what you need.

3.12 What if I want to make changes to emodpy-malaria and run with those, instead of a released version?

(This is a duplicate.)

There are a couple of ways of doing that. Option 1: Do a Dev Install:

```
pip install -e .
```

This will make your site packages map to your local code until you do a new pip install of the package.

Option 2: Creating a wheel from your local code and pip install it (each time you make a change).:

```
python setup.py bdist_wheel
pip3 install dist/<newly_create_file.whl>
```

Some people prefer option 1 because it's "one and done". Some people prefer option 2 because it keeps you thinking in terms of packaging, versioning, and installing even while you're developing.

3.13 I pip installed emodpy-malaria, but I want to make changes. How should I do that?

Install at a command prompt using the following:

```
python package_setup.py develop
```

This method is the most popular and proven, though there are some other options. Installing this way means that the emodpy-malaria module in site-packages actually points to the same code as you have checked out in git. For more detail, see this Stack Overflow post.

However, we aim to get the desired changes quickly tested and included in the versioned module we release via pip install.

3.14 How do I set configuration parameters?

Define your own parameter-setting function such as `set_param_fn` and pass that function to the emodpy task creator as the `param_custom_cb` parameter. In that function, you can set the parameters directly. For example:

```
    print("nSims: ", params.nSims)


def set_param_fn(config):
    """
    This function is a callback that is passed to emod-api.config to set parameters The␣
↪Right Way.
    """
    import emodpy_malaria.malaria_config as conf
    config = conf.set_team_defaults(config, manifest)
    conf.add_species(config, manifest, ["gambiae"])

    return config


def build_camp():
    """
        Build a campaign input file for the DTK using emod_api.
        Right now this function creates the file and returns the filename. If calling␣
↪code just needs an asset that's fine.
    """
    import emod_api.campaign as camp

    import emodpy_malaria.interventions.treatment_seeking as ts

    # This isn't desirable. Need to think about right way to provide schema (once)
    camp.schema_path = manifest.schema_file

    ts.add_treatment_seeking(camp, start_day=1, node_ids=[321])
    return camp
```

See examples/start_here/example.py. for additional information.

If you prefer something more modular, you can call a function in a standalone script/file that sets the configuration parameters.

Are there defaults? Great question. If you don't set any configuration parameters, they will have defaults based on the schema. The malaria team has set team defaults in `emodpy_malaria.config.set_team_defaults()`. These defaults can be seen in config.py.

3.15 How do I specify the log level for EMOD? I get a schema error when I try to set it now.

TBD

3.16 How do I specify the vector species for my scenario?

See the excerpt below or the complete example of setting the vector species and parameter values associated with each species in examples/start_here/example.py.

```
    print("nSims: ", params.nSims)


def set_param_fn(config):
    """
    This function is a callback that is passed to emod-api.config to set parameters The␣
↪Right Way.
    """
    import emodpy_malaria.malaria_config as conf
    config = conf.set_team_defaults(config, manifest)
    conf.add_species(config, manifest, ["gambiae"])

    return config


def build_camp():
    """
        Build a campaign input file for the DTK using emod_api.
        Right now this function creates the file and returns the filename. If calling␣
↪code just needs an asset that's fine.
    """
    import emod_api.campaign as camp

    import emodpy_malaria.interventions.treatment_seeking as ts

    # This isn't desirable. Need to think about right way to provide schema (once)
    camp.schema_path = manifest.schema_file

    ts.add_treatment_seeking(camp, start_day=1, node_ids=[321])
    return camp
```

A helper function to make this task even easier may be coming shortly.

3.17 Where else should I search for functions?

Yes, emod-api. Any functionality that is not malaria-specific (or disease-specific) will be found in emod-api. In particular you'll probably find very useful functions for crafting campaigns in emod-api.interventions.common, such as the ScheduledCampaignEvent function and the TriggeredCampaignEvent function.

3.18 Do I need to be connected to the VPN?

The original way of procuring the model binary itself was via a call to get_model_files(). This required you to be VPN-ed in. This is no longer the preferred approach. Instead you will want to use the 'bootstrap' approach. This involves installing the emod_malaria package, which should happen automatically, and using code like:

```
import emod_malaria.bootstrap as dtk
dtk.setup(...)
```

This does not require VPN. The value you pass to setup is the path where the model files will be put.

3.19 Is there an example of creating a demographics file from scratch with the API?

Yes, see examples/create_demographics, there are also examples in emodpy-measles and emodpy-hiv. We are working to add some to emod_api.demographics. The basic idea is you use one of 3 node creators, and then use the Setter API to set up the node defaults for fertility, mortality, age structure, initial immunity, individual 'risk', and initial prevalance. The first node creator, from_template_node, is very basic and usually for quickstarts or toy models. It lets you create a single node demographics file with a given population. The second creator, from_csv, lets you create a multinode demographics using a csv file with population data as an input. The third creator, from_params, lets you create a multinode demographics without specific node data but instead with a few parameters that represent the overall population and the population heterogeneity.

3.20 I see a lot of MALARIA_SIM examples. Are there any VECTOR_SIM examples?

Yes! The following examples use VECTOR_SIM:

- examples/migration_spatial_vector_sim
- examples/vector_basic
- examples/vector_genetics_insecticide_resistance
- examples/vector_genetics_vector_sim

3.21 Is there a multi-node or spatial example?

Yes. See:

- examples/migration_spatial_vector_sim
- examples/migration_spatial_malaria_sim
- examples/jonr_1

3.22 Are there simple campaign/intervention examples?

Yes. See:

- examples/outdoor_rest_kill_male_mosquitoes
- examples/inputEIR
- examples/ivermectin

3.23 Is there a drug campaign example?

Yes. See:

- examples/drug_campaign
- examples/diagnostic_survey

3.24 Is there a campaign sweep example?

Yes. See:

- examples/campaign_sweep

3.25 Is there a demographics sweep example?

Yes. See:

- examples/demographics_sweep

3.26 Is there a serialization/burn-in example?

Yes. See:

- examples/burnin_create
- examples/burnin_use

3.27 Is there a reporter configuration example?

Yes. See:

- examples/add_reports
- examples/filtered_report

PYTHON MODULE INDEX

e

INDEX

A

C

D

E

F

G

H

I

O

P

R

S

T

U

V

W

Y