emodpy-malaria

Institute for Disease Modeling

Feb 02, 2021

CONTENTS

emodpy-malaria is a collection of Python scripts and utilities created to streamline user interactions with EMOD and idmtools for modeling malaria.

CHAPTER

ONE

INSTALLATION

You can install emodpy-malaria using the instructions in the GitHub repository README.

CHAPTER

TWO

EMODPY_MALARIA

2.1 emodpy_malaria package

2.1.1 Subpackages

emodpy_malaria.demographics package

Submodules

emodpy_malaria.demographics.MalariaDemographics module

class emodpy_malaria.demographics.MalariaDemographics.**MalariaDemographics**(*nodes*, *idref='Gridded world grump2.5arcmin'*, *base_file=None*, *init_prev=0.2*)

Bases: emod_api.demographics.Demographics.Demographics

This class is derived from emod_api.demographics' Demographics class so that we can set certain defaults for Malaria in construction.

emodpy_malaria.demographics.MalariaDemographics.**fromBasicNode**(*lat=0*, *lon=0*, *pop=1000000.0*, *name=1*, *forced_id=1*, *init_prev=0.2*)

This function creates a single-node MalariaDemographics instance from the params you give it.

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

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

emodpy_malaria.interventions package

Submodules

emodpy_malaria.interventions.bednet module

emodpy_malaria.interventions.bednet.**Bednet**(*camp*, *start_day*, *coverage=1.0*, *blocking_eff=1*, *killing_eff=1*, *repelling_eff=1*, *usage_eff=1*, *blocking_decay_rate=0*, *blocking_predecay_duration=365*, *killing_decay_rate=0*, *killing_predecay_duration=365*, *repelling_decay_rate=0*, *repelling_predecay_duration=365*, *usage_decay_rate=0*, *usage_predecay_duration=365*, *node_ids=None*, *insecticide=None*)

Simple Bednet with small param set. Note Start_Day is initialized as 1, recommend that this be aligned with the start of the simulation

emodpy_malaria.interventions.bednet.**BabyBednet**(*camp*, *start_day*, *coverage=1.0*, *blocking_eff=1*, *killing_eff=1*, *repelling_eff=1*, *usage_eff=1*, *insecticide=None*)

BabyBednet is not for babies. It's simpler bednet with just the basic configuration controls.

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

emodpy_malaria.interventions.drug module

emodpy_malaria.interventions.drug.**AntiMalarialDrug**(*camp*, *start_day*, *coverage=1.0*, *drug_name='Chloroquine'*, *node_ids=None*)

AntiMalarialDrug intervention wrapper.

emodpy_malaria.interventions.drug.**new_intervention_as_file**(*camp*, *start_day*, *filename=None*)

emodpy_malaria.interventions.irs module

emodpy_malaria.interventions.irs.**IRSHousingModification**(*camp*, *start_day*, *coverage=1.0*, *blocking_eff=1*, *killing_eff=1*, *insecticide=None*, *node_ids=None*)

MCV1 Campaign :param coverage: Demographic Coverage :param blocking: :param killing: Note Start_Day is initialized as 1, recommend that this be aligned with the start of the simulation

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

emodpy_malaria.interventions.ivermectin module

emodpy_malaria.interventions.ivermectin.**Ivermectin**(*schema_path_container*, *killing_effect*, *start_day=0*, *target_coverage=1.0*, *target_num_individuals=None*, *killing_duration_box=0*, *killing_exponential_decay_rate=0*)

Parameters

- **start_day** – day to give out this ivermectin
- **target_coverage** – probability of choosing an individual
- **target_num_individuals** – number of individuals to choose
- **killing_effect** – initial parasite killing effect
- **killing_duration_box** – box duration for killing effect
- **killing_exponential_decay_rate** – rate at which killing effect decays per day. Use 0 for box duration only.

Returns: campaign event

emodpy_malaria.interventions.outdoorrestkill module

emodpy_malaria.interventions.outdoorrestkill.**OutdoorRestKill**(*schema_path_container*, *killing_effect*, *insecticide_name=None*, *start_day=0*, *target_coverage=1.0*, *killing_predecay_duration=0*, *killing_decay_rate=0*, *node_ids=None*)

Parameters

- **schema_path_container** –
- **killing_effect** –
- **insecticide_name** –
- **start_day** –
- **target_coverage** –
- **killing_predcay_duration** –
- **killing_decay_rate** –

Returns:

emodpy_malaria.interventions.spacespraying module

emodpy_malaria.interventions.spacespraying.**SpaceSpraying**(*camp*, *start_day*, *coverage=1.0*, *killing_eff=1*, *insecticide=None*, *constant_duration=100*, *node_ids=None*)

MCV1 Campaign :param coverage: Demographic Coverage :param blocking: :param killing: Note Start_Day is initialized as 1, recommend that this be aligned with the start of the simulation

emodpy_malaria.interventions.spacespraying.**new_intervention_as_file**(*camp*, *start_day*, *filename=None*)

emodpy_malaria.interventions.sugartrap module

emodpy_malaria.interventions.sugartrap.**SugarTrap**(*camp*, *start_day*, *coverage=1.0*, *killing_eff=1*, *insecticide=None*, *constant_duration=100*, *node_ids=None*)

SugarTrap intervention wrapper.

emodpy_malaria.interventions.sugartrap.**new_intervention_as_file**(*camp*, *start_day*, *filename=None*)

emodpy_malaria.interventions.udbednet module

emodpy_malaria.interventions.udbednet.**UDBednet**(*camp*, *start_day: int = 1*, *discard_config: dict = None*, *coverage: float = 1*, *ind_property_restrictions: list = None*, *blocking_eff: float = 0.9*, *blocking_constant_duration: int = 0*, *blocking_decay_rate: float = 0.0013698630136986301*, *killing_eff: float = 0.6*, *killing_constant_duration: int = 0*, *killing_decay_rate: float = 0.0006849315068493151*, *repelling_eff: float = 0*, *repelling_constant_duration: int = 0*, *repelling_decay_rate: float = 0.0006849315068493151*, *iv_name: str = 'UsageDependentBednet'*, *age_dependence: dict = None*, *seasonal_dependence: dict = None*, *insecticide: str = None*, *cost: int = 5*, *node_ids: list = None*, *triggered_campaign_delay: dict = None*, *triggers: list = None*, *duration: int = -1*, *check_eligibility_at_trigger: bool = False*)

Add an insecticide-treated net (ITN) intervention with a seasonal usage pattern to the campaign using the **UsageDependentBednet** class. The arguments **birth_triggered** and **triggered_condition_list** are mutually ex-

clusive. If both are provided, **triggered_condition_list** is ignored. You must add the following custom events to your config.json:

- Bednet_Discarded
- Bednet_Got_New_One
- Bednet_Using

Parameters

- **start** – The day on which to start distributing the bednets (**Start_Day** parameter).
- **coverage** – Fraction of the population receiving bed nets in a given distribution event
- **blocking_config** – The value passed gets directly assigned to the Blocking_Config parameter. Durations are in days. Default is blocking_config= WaningEffectExponential(Decay_Time_Constant=730, Initial_Effect=0.9)

  This could be dictionary such as:

```
{
    "Box_Duration": 3650,
    "Initial_Effect": 0,
    "class": "WaningEffectBox"
}
```

- **killing_config** – The value passed gets directly assigned to the Killing_Config parameter. Durations are in days. Default is killing_config = WaningEffectExponential(Decay_Time_Constant=1460, Initial_Effect=0.6)

  This could be dictionary such as:

```
{
    "Box_Duration": 3650,
    "Initial_Effect": 0,
    "Decay_Time_Constant": 150,
    "class": "WaningEffectBoxExponential"
}
```

- **repelling_config** – The value passed gets directly assigned to the Repelling_Config parameter. Durations are in days. Default is repelling_config = WaningEffectExponential(Decay_Time_Constant=1460, Initial_Effect=0.0)

  This could be dictionary such as:

```
{
    "Box_Duration": 3650,
    "Initial_Effect": 0,
    "Decay_Time_Constant": 150,
    "class": "WaningEffectBoxExponential"
}
```

- **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}
```

- **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. Default is constant use during the year. 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. Times are days of the year Examples:

```
{"Times":[], "Values":[]} or {"min_cov":0.45, "max_day":300}
cost: The per-unit cost (**Cost_To_Consumer** parameter).
nodeIDs: The list of nodes to apply this intervention to (**Node_
↪List**
parameter). If not provided, set value of NodeSetAll.
```

- **birth_triggered** – If true, event is specified as a birth-triggered intervention.
- **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.
- **triggered_campaign_delay** – (Optional) After the trigger is received, the number of time steps until the campaign starts. Eligibility of people or nodes for the campaign is evaluated on the start day, not the triggered day. triggered_campaign_delay is a dict. Specify the actual delay distribution params, not the distribution type. E.g., { "Delay_Distribution_Constant": 14" } Delay is in days
- **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. This argument cannot configure birth-triggered ITN (use **birth_triggered** instead).
- **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}]`.
- **node_property_restrictions** – The NodeProperty key:value pairs that nodes must have to receive the intervention (**Node_Property_Restrictions** parameter). In the format `[{"Place":"RURAL"}, {"ByALake":"Yes}]`
- **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.

Returns None

NOTE: Previous was 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_ITN_age_season(config_builder, start=1, coverage=1, killing_config=killing_
↪config,
           blocking_config=blocking_config, discard_config = discard_config
           age_dependence=age_dependence, cost=5, birht_triggered=True,␣
↪duration=-1,
           node_property_restrictions=[{"Place": "Rural"]):
```

emodpy_malaria.interventions.udbednet.**new_intervention_as_file**(*camp*, *start_day*, *filename=None*)

emodpy_malaria.reporters package

Submodules

emodpy_malaria.reporters.builtin module

class emodpy_malaria.reporters.builtin.**ReportVectorGenetics**(*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.**ReportVectorStats**(*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.**MalariaSummaryReport**(*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.**MalariaPatientJSONReport**(*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.**MalariaTransmissionReport**(*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

2.1.2 Submodules

emodpy_malaria.config module

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

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

emodpy_malaria.config.**set_team_vs_params**(*config*, *mani*)

emodpy_malaria.config.**get_species_params**(*cb*, *species*)

emodpy_malaria.config.**set_team_drug_params**(*config*, *mani*)

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

emodpy_malaria.config.**set_species**(*config*, *species_to_select*)
Use this function to specify which mosquito species to use in the simulation.

Parameters

- **config** – schema-backed config smart dict
- **species_to_list** – list of 1 or more strings.

Returns None

emodpy_malaria.config.**set_resistances**(*config*)
Use this function after you're done calling add_resistance. config is the input and the output

emodpy_malaria.config.**add_alleles**(*allele_names_in*, *allele_inits_in*)

This is public API function for user to add alleles. User specifies the list of alleles and corresponding initial distribution.

emodpy_malaria.config.**add_mutation**(*from_allele*, *to_allele*, *rate*)

Public API function for user to add mutations as part of vector genetics configuration. A mutation is specified with a source allele, a destination allele, and a rate

emodpy_malaria.config.**add_trait**(*manifest*, *sex_genes*, *allele_pair*, *trait_name*, *trait_value*)

Use this function to add traits as part of vector genetics configuration. Should produce something like:

```
{
   "Allele_Combinations": [["X", "X"],["a0", "a1"]],
   "Trait_Modifiers": {"INFECTED_BY_HUMAN": 0}
},
```

emodpy_malaria.config.**add_resistance**(*manifest*, *insecticide_name*, *species*, *combo*, *blocking=1.0*, *killing=1.0*)

Use this function to add insecticide resistances. An insecticide can have a list of resistances. Add each resistance separately with the same name:

```
Insecticides = [
{
  "Name": "pyrethroid",
  "Resistances": [
    {
      "Allele_Combinations": [
      [
        "a1",
        "a1"
      ]
     ],
    "Blocking_Modifier": 1.0,
    "Killing_Modifier": pyrethroid_killing,
    "Species": "gambiae"
  }
 ]
},
```

emodpy_malaria.config.**set_genetics**(*vsp*, *manifest*)

Don't need to pass these anymore since they are module variables. But actually need to try with more than one set and see where I end up in terms of design.

PYTHON MODULE INDEX

e

INDEX