Batchrunner

A single class to manage a batch run or parameter sweep of a given model.

batch_run(*model_cls: Type[Model], parameters: Mapping[str, Any | Iterable[Any]], number_processes: int | None = 1, iterations: int = 1, data_collection_period: int = -1, max_steps: int = 1000, display_progress: bool = True***)** → **List[Dict[str, Any]]** [source]

Batch run a mesa model with a set of parameter values.

Parameters

model_clsType[Model]

The model class to batch-run

parametersMapping[str, Union[Any, Iterable[Any]]],

Dictionary with model parameters over which to run the model. You can either pass single values or iterables.

number_processesint, optional

Number of processes used, by default 1. Set this to None if you want to use all CPUs.

iterationsint, optional

Number of iterations for each parameter combination, by default 1

data_collection_periodint, optional

Number of steps after which data gets collected, by default -1 (end of episode)

max_stepsint, optional

Maximum number of model steps after which the model halts, by default 1000

display_progressbool, optional

Display batch run process, by default True

Returns

List[Dict[str, Any]]

[description]

exception **ParameterError**(*bad_names*) [source]

exception **VariableParameterError**(*bad_names*) [source]

class **FixedBatchRunner**(*model_cls*, *parameters_list=None*, *fixed_parameters=None*, *iterations=1*, *max_steps=1000*, *model_reporters=None*, *agent_reporters=None*, *display_progress=True*) [source]

This class is instantiated with a model class, and model parameters associated with one or more values. It is also instantiated with model and agent-level reporters, dictionaries mapping a variable name to a function which collects some data from the model or its agents at the end of the run and stores it.

Note that by default, the reporters only collect data at the *end* of the run. To get step by step data, simply have a reporter store the model's entire DataCollector object.

Create a new BatchRunner for a given model with the given parameters.

Args:

model_cls: The class of model to batch-run. parameters_list: A list of dictionaries of parameter sets.

The model will be run with dictionary of parameters. For example, given parameters_list of

[{"homophily": 3, "density": 0.8, "minority_pc": 0.2}, {"homophily": 2, "density": 0.9, "minority_pc": 0.1}, {"homophily": 4, "density": 0.6, "minority_pc": 0.5}]

3 models will be run, one for each provided set of parameters.

fixed_parameters: Dictionary of parameters that stay same through

all batch runs. For example, given fixed_parameters of

{"constant_parameter": 3},

every instantiated model will be passed constant_parameter=3 as a kwarg.

iterations: The total number of times to run the model for each set

of parameters.

max_steps: Upper limit of steps above which each run will be halted

if it hasn't halted on its own.

model_reporters: The dictionary of variables to collect on each run

at the end, with variable names mapped to a function to collect them. For example:

{“agent_count”: lambda m: m.schedule.get_agent_count()}

agent_reporters: Like model_reporters, but each variable is now

collected at the level of each agent present in the model at the end of the run.

display_progress: Display progress bar with time estimation?

run_all() [source]

Run the model at all parameter combinations and store results.

run_model(*model*) [source]

Run a model object to completion, or until reaching max steps.

If your model runs in a non-standard way, this is the method to modify in your subclass.

collect_model_vars(*model*) [source]

Run reporters and collect model-level variables.

collect_agent_vars(*model*) [source]

Run reporters and collect agent-level variables.

get_model_vars_dataframe() [source]

Generate a pandas DataFrame from the model-level variables collected.

get_agent_vars_dataframe() [source]

Generate a pandas DataFrame from the agent-level variables collected.

get_collector_model() [source]

Passes pandas dataframes from datacollector module in dictionary format of model reporters :return: dict {(Param1, Param2,...,iteration): <DataCollector Pandas DataFrame>}

get_collector_agents() [source]

Passes pandas dataframes from datacollector module in dictionary format of agent reporters :return: dict {(Param1, Param2,...,iteration): <DataCollector Pandas DataFrame>}

class **BatchRunner**(*model_cls*, *variable_parameters=None*, *fixed_parameters=None*, *iterations=1*, *max_steps=1000*, *model_reporters=None*, *agent_reporters=None*, *display_progress=True*) [source]

DEPRECATION WARNING: BatchRunner Class has been replaced batch_run function This class is instantiated with a model class, and model parameters associated with one or more values. It is also instantiated with model and agent-level reporters, dictionaries mapping a variable name to a function which collects some data from the model or its agents at the end of the run and stores it.

Note that by default, the reporters only collect data at the *end* of the run. To get step by step data, simply have a reporter store the model's entire DataCollector object.

Create a new BatchRunner for a given model with the given parameters.

Args:

model_cls: The class of model to batch-run. variable_parameters: Dictionary of parameters to lists of values.

The model will be run with every combo of these parameters. For example, given variable_parameters of

{"param_1": range(5),

"param_2": [1, 5, 10]}

models will be run with {param_1=1, param_2=1},

{param_1=2, param_2=1}, …, {param_1=4, param_2=10}.

fixed_parameters: Dictionary of parameters that stay same through

all batch runs. For example, given fixed_parameters of

{"constant_parameter": 3},

every instantiated model will be passed constant_parameter=3 as a kwarg.

iterations: The total number of times to run the model for each

combination of parameters.

max_steps: Upper limit of steps above which each run will be halted

if it hasn't halted on its own.

model_reporters: The dictionary of variables to collect on each run

at the end, with variable names mapped to a function to collect them. For example:

{"agent_count": lambda m: m.schedule.get_agent_count()}

agent_reporters: Like model_reporters, but each variable is now

collected at the level of each agent present in the model at the end of the run.

display_progress: Display progress bar with time estimation?

class **BatchRunnerMP(***model_cls*, *nr_processes=None*, ***kwargs***)** [source]

DEPRECATION WARNING: BatchRunner class has been replaced by batch_run Child class of BatchRunner, extended with multiprocessing support.

Create a new BatchRunnerMP for a given model with the given parameters.

model_cls: The class of model to batch-run. nr_processes: int

the number of separate processes the BatchRunner should start, all running in parallel.

kwargs: the kwargs required for the parent BatchRunner class

run_all() [source]

Run the model at all parameter combinations and store results, overrides run_all from BatchRunner.