

diagram

March 14, 2021

1 Diagrams for cadCAD model structure through cadCAD_diagram

Danilo Lessa Bernardineli

This is a mini-demo showcasing the `cadCAD_diagram` plugin, which allows the user to create visualizations of the structure and flows for any `cadCAD` model! Just jump to the end so that you see the output of it.

In order to install the plugin, just install through pip by passing `pip install cadCAD_diagram`. The most typical use-case is to visualize a given `cadCAD` configuration object through:

```
from cadCAD_diagram import diagram_from_config
from cadCAD import configs

# Generate visualization for the first config
diagram_from_config(configs[0])
```

1.1 Dependences

```
[1]: %%capture
      !pip install cadcad

[2]: import pandas as pd
      import matplotlib.pyplot as plt
      import numpy as np
      from cadCAD.configuration import Experiment
      from cadCAD.configuration.utils import config_sim
      from cadCAD.engine import ExecutionMode, ExecutionContext, Executor
```

1.2 Definitions

1.2.1 Initial conditions and parameters

```
[3]: initial_conditions = {
      'prey_population': 100,
      'predator_population': 15
```

```

    }

params = {
    "prey_birth_rate": [1.0],
    "predator_birth_rate": [0.01],
    "predator_death_const": [1.0],
    "prey_death_const": [0.03],
    "dt": [0.1] # Precision of the simulation. Lower is more accurate / slower
}

simulation_parameters = {
    'N': 7,
    'T': range(200),
    'M': params
}

```

1.2.2 Policies

```

[4]: def p_predator_births(params, step, sL, s):
    dt = params['dt']
    predator_population = s['predator_population']
    prey_population = s['prey_population']
    birth_fraction = params['predator_birth_rate'] + np.random.random() * 0.0002
    births = birth_fraction * prey_population * predator_population * dt
    return {'add_to_predator_population': births}

def p_prey_births(params, step, sL, s):
    dt = params['dt']
    population = s['prey_population']
    birth_fraction = params['prey_birth_rate'] + np.random.random() * 0.1
    births = birth_fraction * population * dt
    return {'add_to_prey_population': births}

def p_predator_deaths(params, step, sL, s):
    dt = params['dt']
    population = s['predator_population']
    death_rate = params['predator_death_const'] + np.random.random() * 0.005
    deaths = death_rate * population * dt
    return {'add_to_predator_population': -1.0 * deaths}

def p_prey_deaths(params, step, sL, s):
    dt = params['dt']
    death_rate = params['prey_death_const'] + np.random.random() * 0.1
    prey_population = s['prey_population']

```

```

predator_population = s['predator_population']
deaths = death_rate * prey_population * predator_population * dt
return {'add_to_prey_population': -1.0 * deaths}

```

1.2.3 State update functions

```

[5]: def s_prey_population(params, step, sL, s, _input):
    y = 'prey_population'
    x = s['prey_population'] + _input['add_to_prey_population']
    return (y, x)

def s_predator_population(params, step, sL, s, _input):
    y = 'predator_population'
    x = s['predator_population'] + _input['add_to_predator_population']
    return (y, x)

```

1.2.4 State update blocks

```

[6]: partial_state_update_blocks = [
    {
        'label': 'Predator dynamics',
        'ignore': False,
        'debug': True,
        'policies': {
            'predator_births': p_predator_births,
            'predator_deaths': p_predator_deaths
        },
        'variables': {
            'predator_population': s_predator_population
        }
    },
    {
        'label': 'Prey dynamics',
        'ignore': False,
        'policies': {
            'prey_births': p_prey_births,
            'prey_deaths': p_prey_deaths
        },
        'variables': {
            'prey_population': s_prey_population
        }
    }
]

```

1.2.5 Configuration and Execution

```
[7]: sim_config = config_sim(simulation_parameters)

exp = Experiment()
exp.append_configs(sim_configs=sim_config,
                  initial_state=initial_conditions,
                  partial_state_update_blocks=partial_state_update_blocks)

from cadCAD import configs
exec_mode = ExecutionMode()
exec_context = ExecutionContext(exec_mode.local_mode)
executor = Executor(exec_context=exec_context, configs=configs)
(records, tensor_field, _) = executor.execute()
```

```

      -----
 /----- / /----- / | /--- \
 /----- / /----- / / | / / / /
 / /----- / / / / / /----- / / / / /
 \--- / \--- / \--- / \----- / / | /----- /
by cadCAD
```

```
Execution Mode: local_proc
Configuration Count: 1
Dimensions of the first simulation: (Timesteps, Params, Runs, Vars) = (200, 5,
7, 2)
Execution Method: local_simulations
SimIDs      : [0, 0, 0, 0, 0, 0, 0]
SubsetIDs: [0, 0, 0, 0, 0, 0, 0]
Ns          : [0, 1, 2, 3, 4, 5, 6]
ExpIDs      : [0, 0, 0, 0, 0, 0, 0]
Execution Mode: parallelized
Total execution time: 0.18s
```

1.2.6 Results

```
[8]: import plotly.express as px
```

```
[9]: df = pd.DataFrame(records)

fig = px.line(df,
              x=df.prey_population,
              y=df.predator_population,
              color=df.run.astype(str))

fig.show()
```

1.3 Diagram

```
[10]: %%capture
!pip install cadCAD_diagram==0.0.1.2
```

```
[11]: from cadCAD_diagram import diagram_from_config
from cadCAD import configs
diagram_from_config(configs[0])
```

```
-----
FileNotFoundError                                Traceback (most recent call last)
~/.local/lib/python3.9/site-packages/graphviz/backend.py in run(cmd, input,
↳ capture_output, check, encoding, quiet, **kwargs)
    163     try:
--> 164         proc = subprocess.Popen(cmd, startupinfo=get_startupinfo(),
↳ **kwargs)
    165     except OSError as e:

/usr/local/lib/python3.9/subprocess.py in __init__(self, args, bufsize,
↳ executable, stdin, stdout, stderr, preexec_fn, close_fds, shell, cwd, env,
↳ universal_newlines, startupinfo, creationflags, restore_signals,
↳ start_new_session, pass_fds, user, group, extra_groups, encoding, errors,
↳ text, umask)
    950
--> 951         self._execute_child(args, executable, preexec_fn, close_fds,
    952                             pass_fds, cwd, env,

/usr/local/lib/python3.9/subprocess.py in _execute_child(self, args, executable,
↳ preexec_fn, close_fds, pass_fds, cwd, env, startupinfo, creationflags, shell,
↳ p2cread, p2cwrite, c2pread, c2pwrite, errread, errwrite, restore_signals, gid,
↳ gids, uid, umask, start_new_session)
    1822             err_msg = os.strerror(errno_num)
-> 1823             raise child_exception_type(errno_num, err_msg,
↳ err_filename)
    1824             raise child_exception_type(err_msg)

FileNotFoundError: [Errno 2] No such file or directory: 'dot'
```

During handling of the above exception, another exception occurred:

```
ExecutableNotFound                                Traceback (most recent call last)
~/.local/lib/python3.9/site-packages/IPython/core/formatters.py in
↳ __call__(self, obj)
    343         method = get_real_method(obj, self.print_method)
    344         if method is not None:
--> 345             return method()
    346         return None
    347     else:
```

```

~/.local/lib/python3.9/site-packages/graphviz/files.py in _repr_svg_(self)
    142
    143     def _repr_svg_(self):
--> 144         return self.pipe(format='svg').decode(self._encoding)
    145
    146     def pipe(self, format=None, renderer=None, formatter=None,
↳ quiet=False):

~/.local/lib/python3.9/site-packages/graphviz/files.py in pipe(self, format,
↳ renderer, formatter, quiet)
    167         data = text_type(self.source).encode(self._encoding)
    168
--> 169         out = backend.pipe(self._engine, format, data,
    170                             renderer=renderer, formatter=formatter,
    171                             quiet=quiet)

~/.local/lib/python3.9/site-packages/graphviz/backend.py in pipe(engine, format
↳ data, renderer, formatter, quiet)
    246         """
    247         cmd, _ = command(engine, format, None, renderer, formatter)
--> 248         out, _ = run(cmd, input=data, capture_output=True, check=True,
↳ quiet=quiet)
    249         return out
    250

~/.local/lib/python3.9/site-packages/graphviz/backend.py in run(cmd, input,
↳ capture_output, check, encoding, quiet, **kwargs)
    165     except OSError as e:
    166         if e.errno == errno.ENOENT:
--> 167             raise ExecutableNotFound(cmd)
    168         else:
    169             raise

ExecutableNotFound: failed to execute ['dot', '-Kdot', '-Tsvg'], make sure the
↳ Graphviz executables are on your systems' PATH

```

[11]: <graphviz.dot.Digraph at 0x7f1b77969a90>

[]: