minimal_cadCAD

March 29, 2021

1 Minimal cadCAD

Danilo Lessa Bernadineli (danilo@block.science)

Emanuel Lima (emanuel@block.science)

This notebook is a minimal implementation of the basic cadCAD structure

---

Github version: https://github.com/cadCAD-org/snippets

Colab version: https://colab.research.google.com/drive/11MEtzHlWLfiflnKg4FaI4cBkvL_6wUtD#scrollTo=9we

[1]:
```
from random import random
from tqdm.auto import tqdm

# Mutable state
variables = {
    'prey_population': 0.5,
    'predator_population': 0.5
}

# Having variables as keys assures
# consistency.
# If it were lists, then a user could provide
# several functions for the same variable,
# which would make the update concurrent.
# wrong_substep_block = [
#     'prey_population': None,
#     'prey_population': None
# ]

### Parameters ###

# Prey Growth Rate
ALPHA = 2 / 3

# Prey Death Rate
BETA = 4 / 3
```

```python
# Predator Growth Rate
DELTA = 1.0

# Predator Death Rate
GAMMA = 1.0

# Time interval
dt = 0.1

### State Update Functions ###

def prey_change(state: dict) -> float:
  x = state['prey_population']
  y = state['predator_population']
  dx = ALPHA * x - BETA * x * y
  value = max(x + dx * dt, 0)
  return value

def predator_change(state: dict) -> float:
  x = state['prey_population']
  y = state['predator_population']
  dy = DELTA * x * y - GAMMA * y
  value = max(y + dy * dt, 0)
  return value

def stochastic_process(state: dict) -> float:
  x = state['prey_population']
  value = x * (1 + random() / 1000)
  return value

### Partial State Update Block ###

## SUBs ##
# Represents intermediate state changes between t and t+1

substep_block_1 = {
    'prey_population': prey_change,
    'predator_population': predator_change,
}

substep_block_2 = {
    'prey_population': stochastic_process
}

# Represents a state change from t -> t+1
timestep_block = [
```

```python
                  substep_block_1,
                  substep_block_2
]

# Simulation config

timestep_count = int(1e3)

### Simulation Execution ###

current_state = variables.copy()
results = []

# Iteration count: (T, S, f given S)
for timestep in tqdm(range(timestep_count)):

  # Iteration count: (S, f given S)
  for substep, substep_block  in enumerate(timestep_block):

    # We copy for making sure that the old state
    # which is history tracked is not changed
    new_state = current_state.copy()

    # Iteration count: f given S
    for variable, variable_update_function in substep_block.items():
      new_state[variable] = variable_update_function(current_state)

    # Append timestep and substep
    new_state.update(timestep=timestep, substep=substep)

    results.append(new_state)
    current_state = new_state


### Prepare results ###

import pandas as pd
df = pd.DataFrame(results)
df.head(10)
```

```
  0%|          | 0/1000 [00:00<?, ?it/s]
```

[1]:

	prey_population	predator_population	timestep	substep
0	0.500000	0.475000	0	0
1	0.500346	0.475000	0	1
2	0.502014	0.451266	1	0
3	0.502452	0.451266	1	1

4	0.505717	0.428814	2	0
5	0.505779	0.428814	2	1
6	0.510580	0.407621	3	0
7	0.510906	0.407621	3	1
8	0.517199	0.387684	4	0
9	0.517338	0.387684	4	1

[2]:
```
import plotly.express as px

px.scatter(df,
           x='predator_population',
           y='prey_population',
           color='timestep')
```

[ ]:
```
```