

Computational writing

In the following, we will enrich the scholarly writing experience with computations.

```
import numpy as np
import matplotlib.pyplot as plt
```

Basic example: predator-prey dynamics

Let's assume we want to study predator-prey dynamics in ecology. The predator-prey equations are a famous model showing that species population sizes do not have to be stable, even in equilibrium. Instead, they can continuously oscillate.

Mathematically, we express the model as follows:

Let $x_t \in \mathbb{R}$ be the number of prey and $y_t \in \mathbb{R}$ the number of predators in the population at time step t . They evolve according to

$$x_{t+1} - x_t =: \Delta x = \alpha x_t - \beta x_t y_t \quad (1)$$

$$y_{t+1} - y_t =: \Delta y = \gamma y_t x_t - \delta y_t \quad (2)$$

where the parameter α represents the preys' birth rate, β the prey's mortality rate, γ the predator efficiency, and δ the predators' death rate.

Right below the mathematical model, we define the computational model:

```
def predprey_model(pre_birth_rate,
                   prey_mortality,
                   predator_efficiency,
                   predator_death_rate,
                   initial_pre,
                   initial_predators,
```

```

        time_length):
    """ Discrete-time predator-prey model. """
    x = -1 * np.ones(time_length)
    y = -1 * np.ones(time_length)
    x[0] = initial_prey
    y[0] = initial_predators
    for t in range(1, time_length):
        x[t] = x[t-1] + prey_birth_rate * x[t-1]\
            - prey_mortality * y[t-1]*x[t-1]
        y[t] = y[t-1] + predator_efficiency * y[t-1]*x[t-1]\
            - predator_death_rate * y[t-1]
    return x, y

```

We test the model,

```

preys, predators = predprey_model(0.1, 0.1, 0.1, 0.01, 1.0, 1.0, 1000)

```

and portray the model output, make the Figure referenceable in by using the Quarto comment commands

```

#| label: fig-modelrun
#| fig-cap: "An exemplary predator-prey model run"

plt.plot(preys, label="preys", color='blue')
plt.plot(predators, label="predators", color='orange')
plt.legend();

```

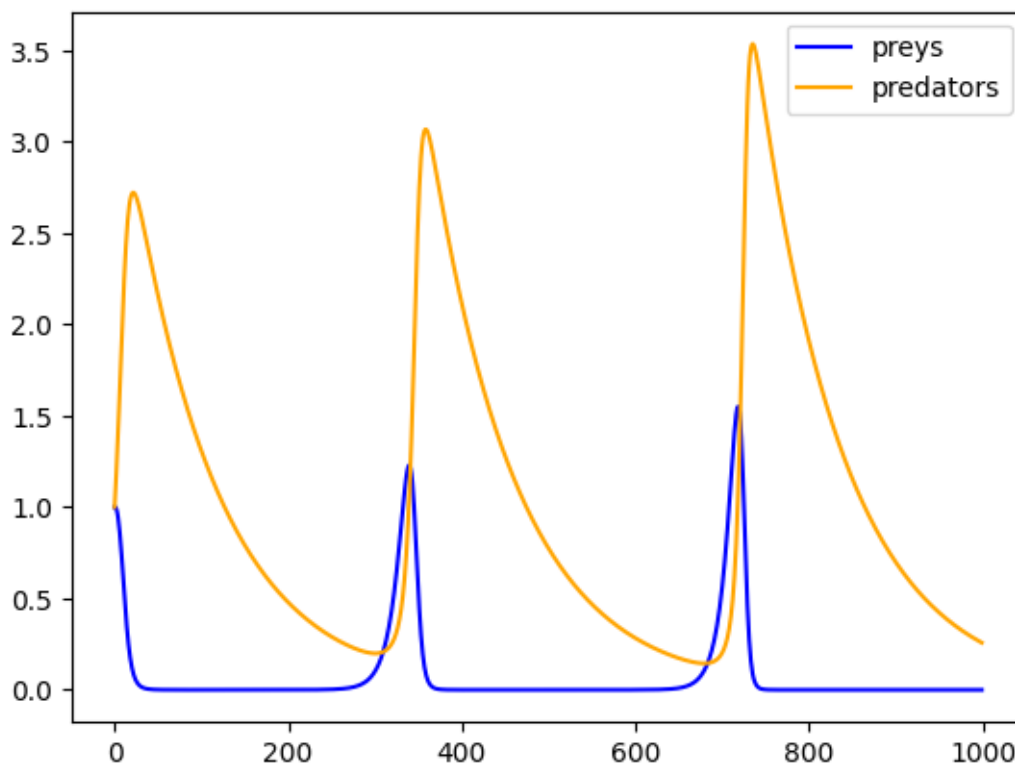


Figure 1: An exemplary predator-prey model run.

Figure 1 shows an exemplary predator-prey model run.

i Note

This basic form of computational writing is the most transparency, collaboration, and reproducibility friendly. This approach allows us to cleanly display our Jupyter notebooks in other places, such as [GitHub](#), [NBViewer](#), or [Google Colab](#).

However, often model code and analysis are too complex to be presented in a single notebook. There are two ways to deal with this problem: Quarto embeds and reusing code written in a notebook with nbdev.

Quarto embeds

Quarto lets you embed the output of another document with the embed shortcode. To do this, simply provide the document path and block or cell identifier, e.g,

```
{{< embed HeavyComputations.ipynb#fig-scatter-plot >}}
```

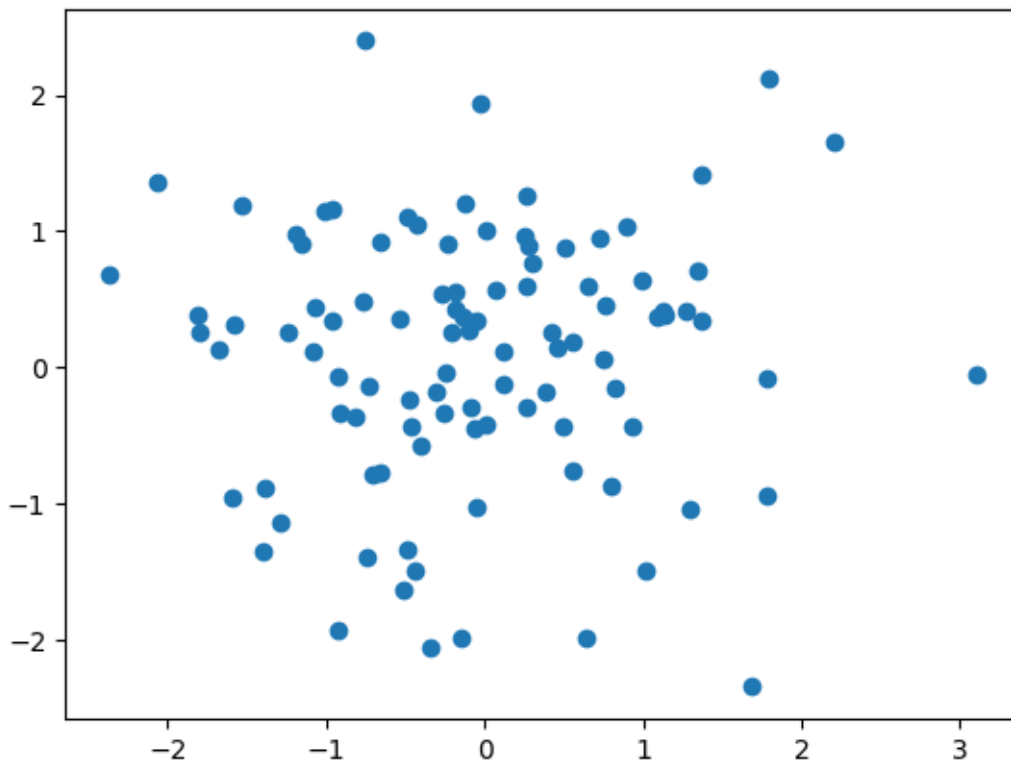


Figure 2

Figure 2 shows a heavy scatter plot.

i Note

Quarto embeds can be a great and easy way to structure a project without using an additional tool. However, the Quarto embeds won't show in any other Jupyter environment.

Another strategy is to reuse code written in a notebook with nbdev.

Code reuse with nbdev

This part makes use of the excellent nbdev Python package ¹. nbdev provides a ready-to-use ecosystem for creating software packages with Jupyter Notebooks, i.e., writing, testing,

¹<https://nbdev.fast.ai/>

documenting, and distributing. We use it, for example, here ². For our computational writing experience, however, we resort to individual components of nbdev.

See nbdev's getting-started guide ³ for how to install it.

One of the key features of nbdev is that it lets you export specific Jupyter Notebook cells to a plain Python file. These Python files can then be easily imported into other Jupyter Notebooks.

You only have to specify the name of Python module file to export to via

```
#|default_exp <name>
```

Then, you can export a Jupyter Notebook cell by simply prepending

```
#|export
```

to it.

... Shows an example

```
from _code.HeavyComputations import noisy_predprey_model

np.random.seed(42)

preys, predators = noisy_predprey_model(0.1, 0.1, 0.1, 0.01, 1.0, 1.0, 1000, 0.0)
plt.plot(preys, lw=3, ls='--', label="preys", color='blue', alpha=0.8)
plt.plot(predators, lw=3, ls='--', label="predators", color='orange', alpha=0.8)

for _ in range(35):
    preys, predators = noisy_predprey_model(0.1, 0.1, 0.1, 0.01, 1.0, 1.0, 1000, 0.025)
    plt.plot(preys, color='blue', alpha=0.1)
    plt.plot(predators, color='orange', alpha=0.1)
plt.legend();
```

Running the model
Running the model
Running the model
Running the model
Running the model

²<https://wbarfuss.github.io/pyCRLD/> and <https://github.com/wbarfuss/pyCRLD>

³https://nbdev.fast.ai/getting_started.html

[illegible]

