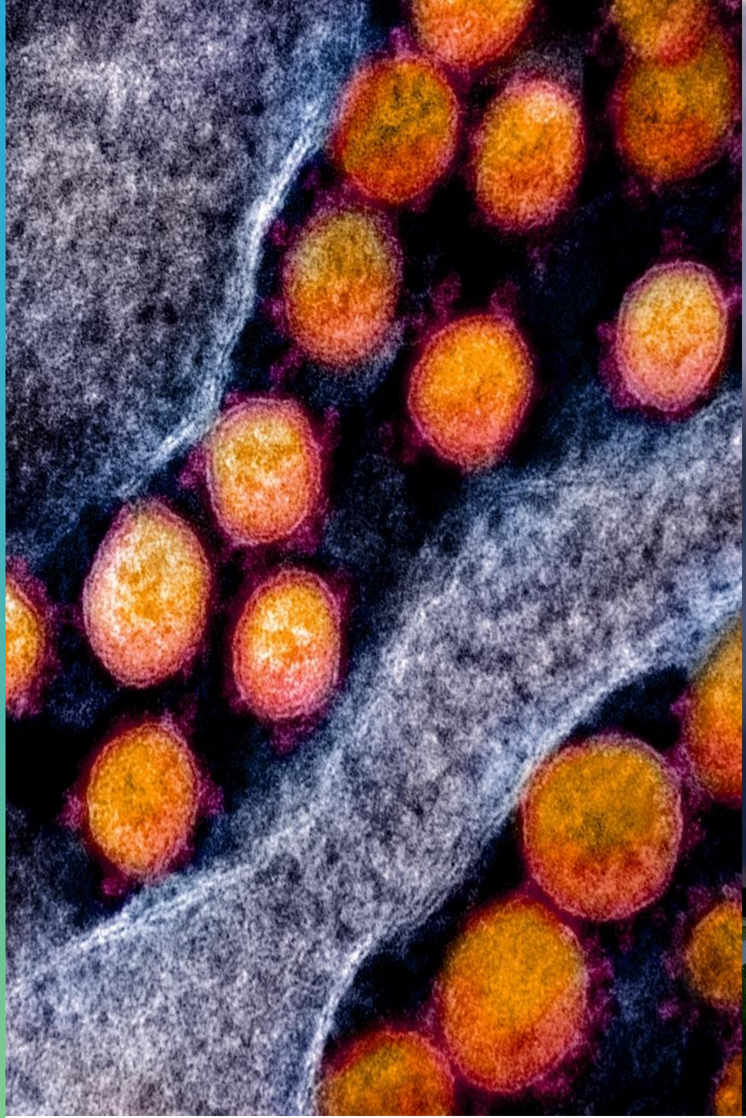
Bringing Randomness to Life: Building a Python tool to tell stories about stochastic processes

Dialid Santiago

Pycon UK 2026

Manchester, UK

Stochastic Processes



Stochastic Processes are all around us

The mathematical definition

In probability theory, a stochastic or random process is a mathematical object usually defined as a in a probability space, family of random variables where the index of the family often has the interpretation of time.

A Vasicek process $X = \{X : t \geq 0\}$ is characterised by the following Stochastic Differential Equation

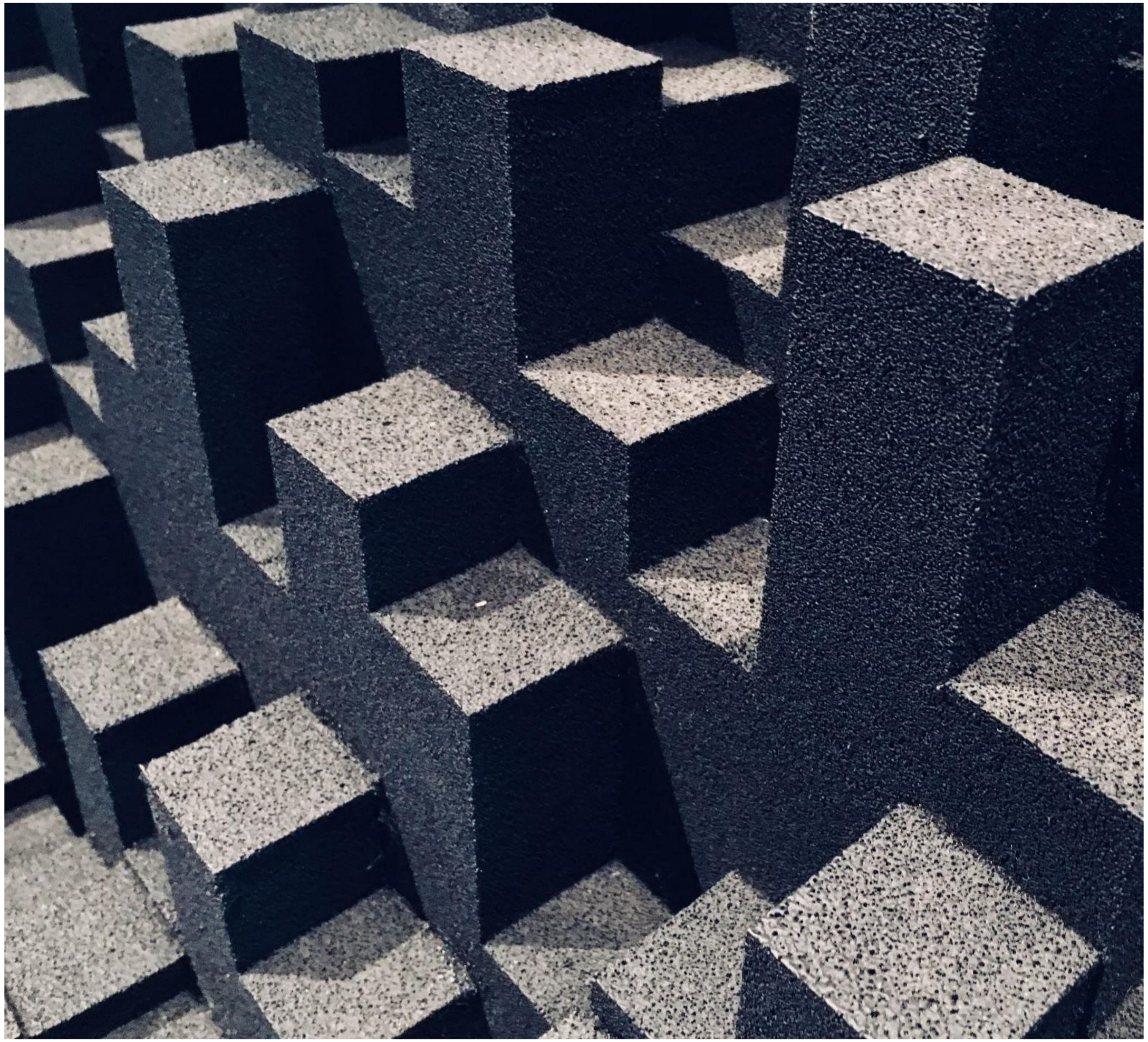
$$dX_t = \theta(\mu - X_t)dt + \sigma dW_t, \quad \forall t \in (0, T],$$

with initial condition $X_0 = x_0$, where

- $\theta > 0$ is the speed of reversion
- μ is the long term mean value.
- $\sigma > 0$ is the instantaneous volatility
- W_t is a standard Brownian Motion.

Example. Orstein Uhlenbeck or Vasicek process

Stochastic processes can be a bit hard to explain using only their mathematical definition



Communication Challenge

- Mathematical equations are precise...
- ... but not always intuitive!
- In many situations, intuition is the most important part
- Students, stakeholders, and decision-makers need clarity

Simulation and Visualization can help us

#NumPy #SciPy #Matplotlib

Task: Use visualization to get a better understanding of the Vasicek process

Reminder of the mathematical definition (This is the last equation, I promise!)

A Vasicek process $X = \{X : t \geq 0\}$ is characterised by the following Stochastic Differential Equation

$$dX_t = \theta(\mu - X_t)dt + \sigma dW_t, \quad \forall t \in (0, T],$$

with initial condition $X_0 = x_0$, where

- $\theta > 0$ is the speed of reversion
- μ is the long term mean value.
- $\sigma > 0$ is the instantaneous volatility
- W_t is a standard Brownian Motion.

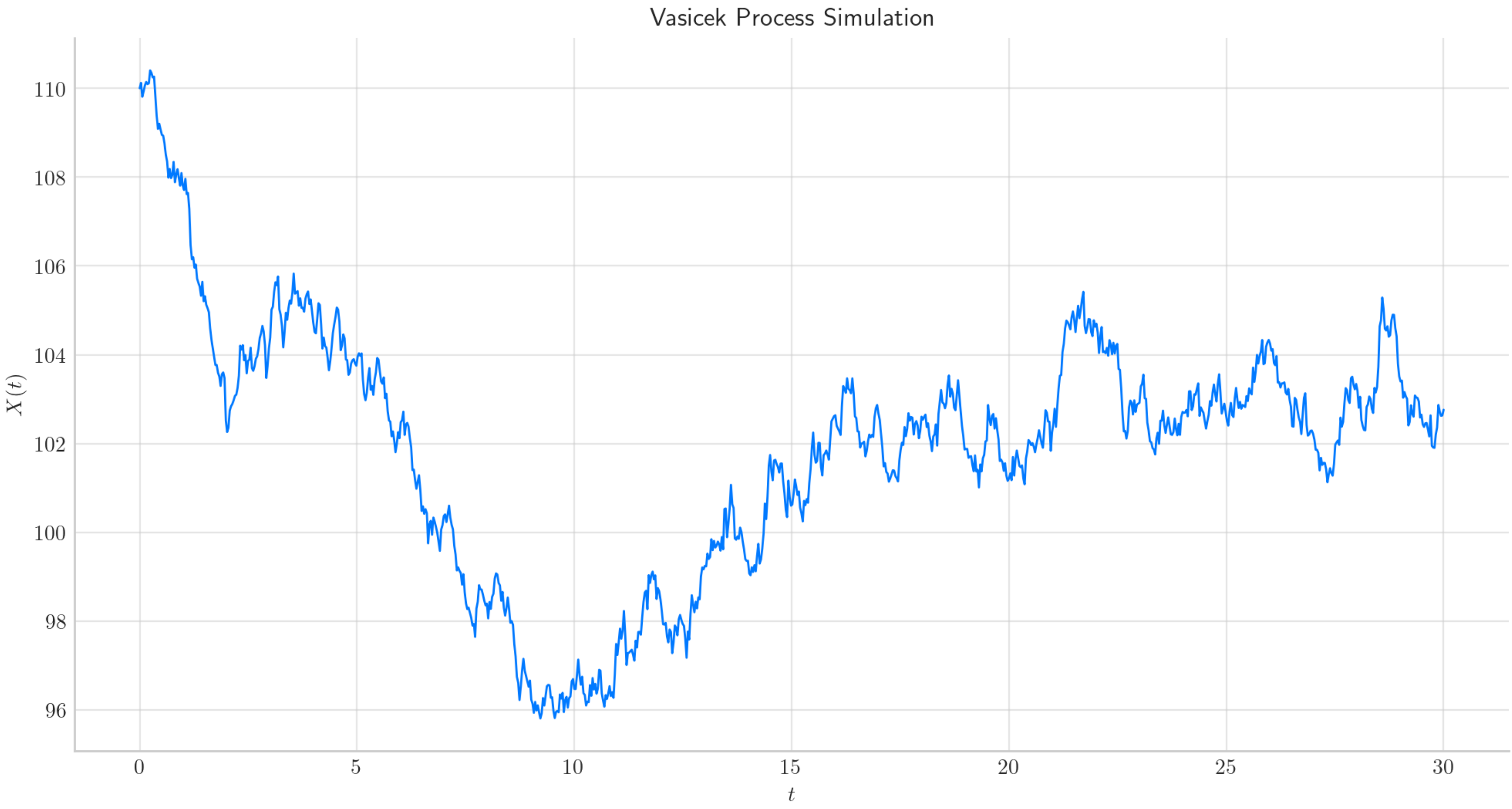
```python
# Process Simulation
def simulate_vasicek(T=1.0, N=500, x0=110, theta=0.25, mu=100, sigma=1.5, seed=None):
    """
    Simulate one path of the Vasicek process using Euler–Maruyama scheme.
    Returns
    -------
    t : np.ndarray
        Time grid.
    X : np.ndarray
        Simulated path.
    """
    if seed is not None:
        np.random.seed(seed)

    dt = T / N
    t = np.linspace(0, T, N+1)
    X = np.zeros(N+1)
    X[0] = x0

    for i in range(N):
        dW = np.sqrt(dt) * np.random.randn()
        X[i+1] = X[i] + theta * (mu - X[i]) * dt + sigma * dW

    return t, X

# Simulate
t, X = simulate_vasicek(T=T, N=500, x0=110, theta=0.25, mu=100, sigma=1.5, seed=42)

# Plot
plt.figure(figsize=(10, 6))
plt.plot(t, X, label="Vasicek path")
plt.title("Simulation of a Vasicek Process")
plt.xlabel("Time")
plt.ylabel("X(t)")
plt.legend()
plt.grid(True, alpha=0.3)
plt.show()
```

Code Snippet

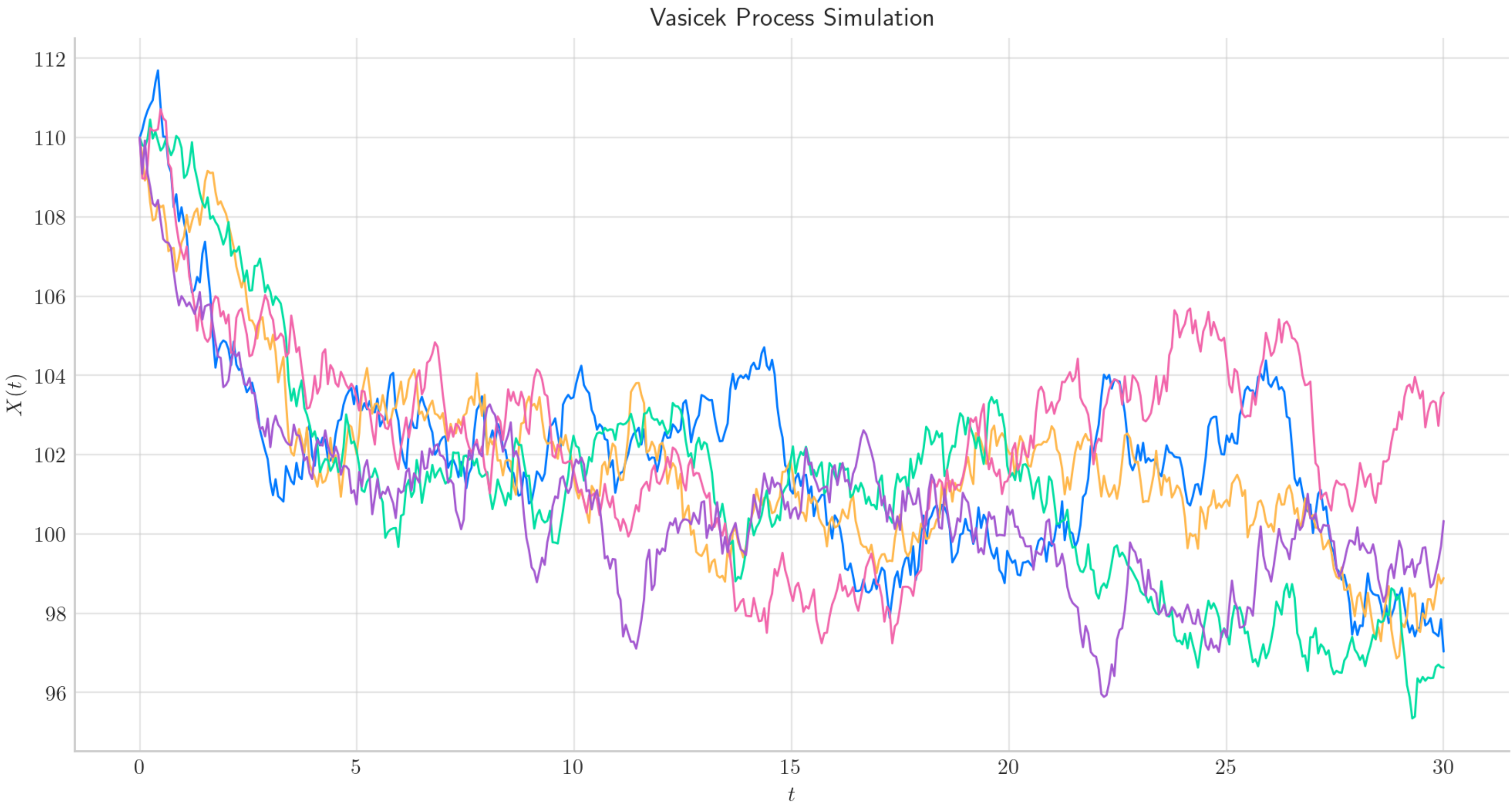
Notes

- There are several ways to do the MC simulation
- We are fixing the parameters
- This code can work for different set of parameters but cannot be easily extended to allow different processes

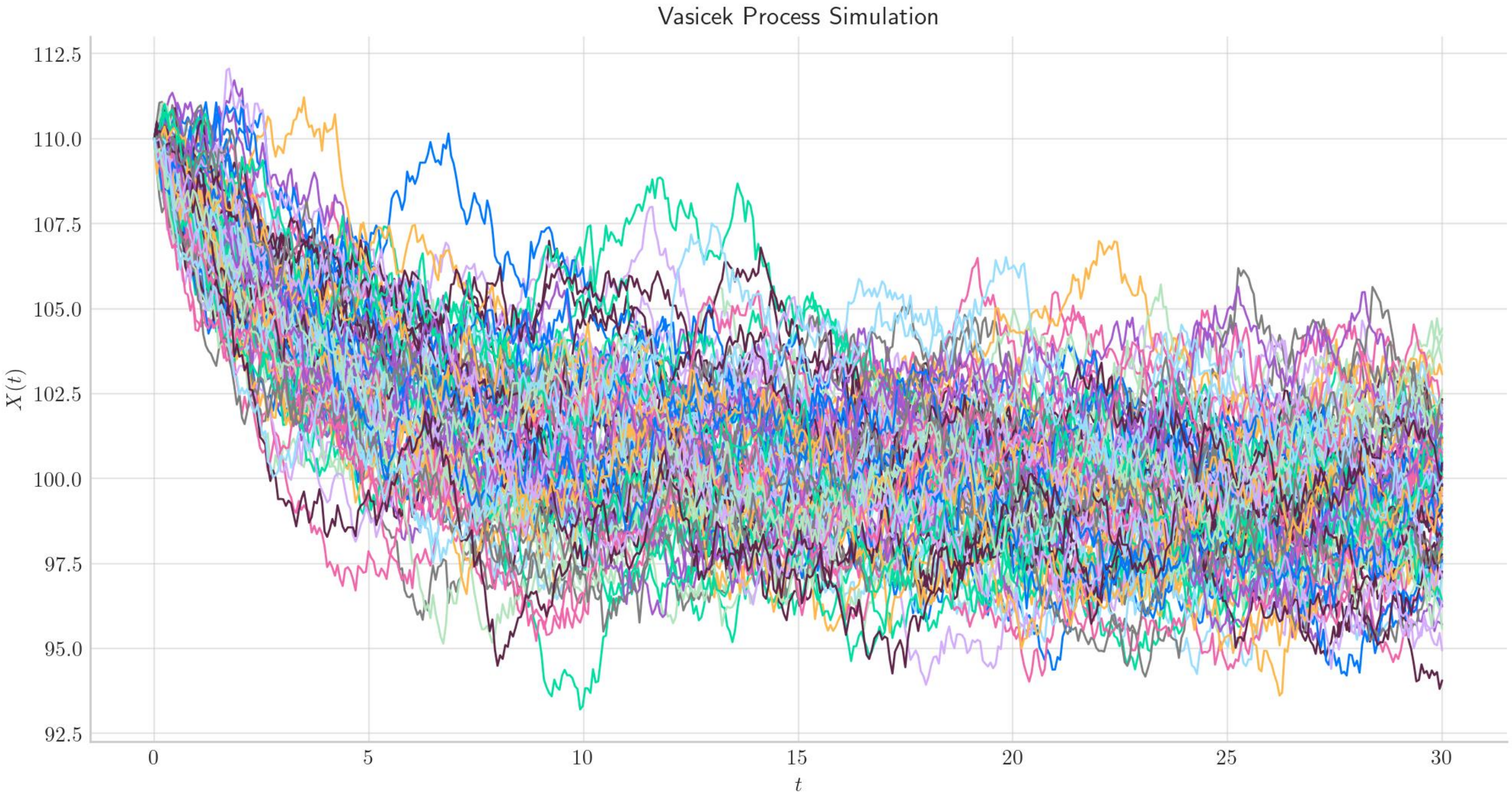
A single path



Plotting multiple paths



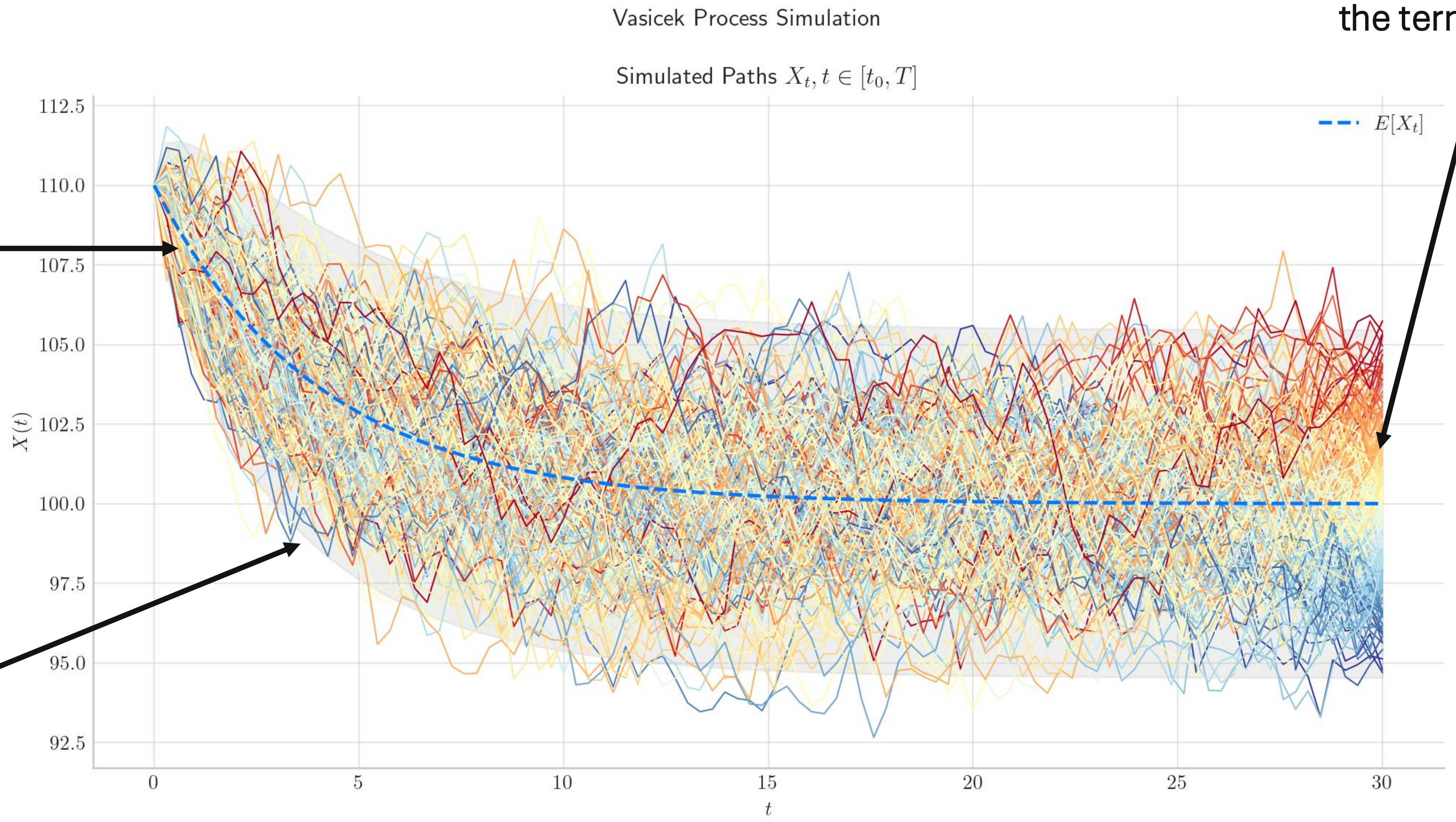
Even more paths?



Adding elements to the plot
Expectation + Envelope + Colormap

Coloring according to the terminal value

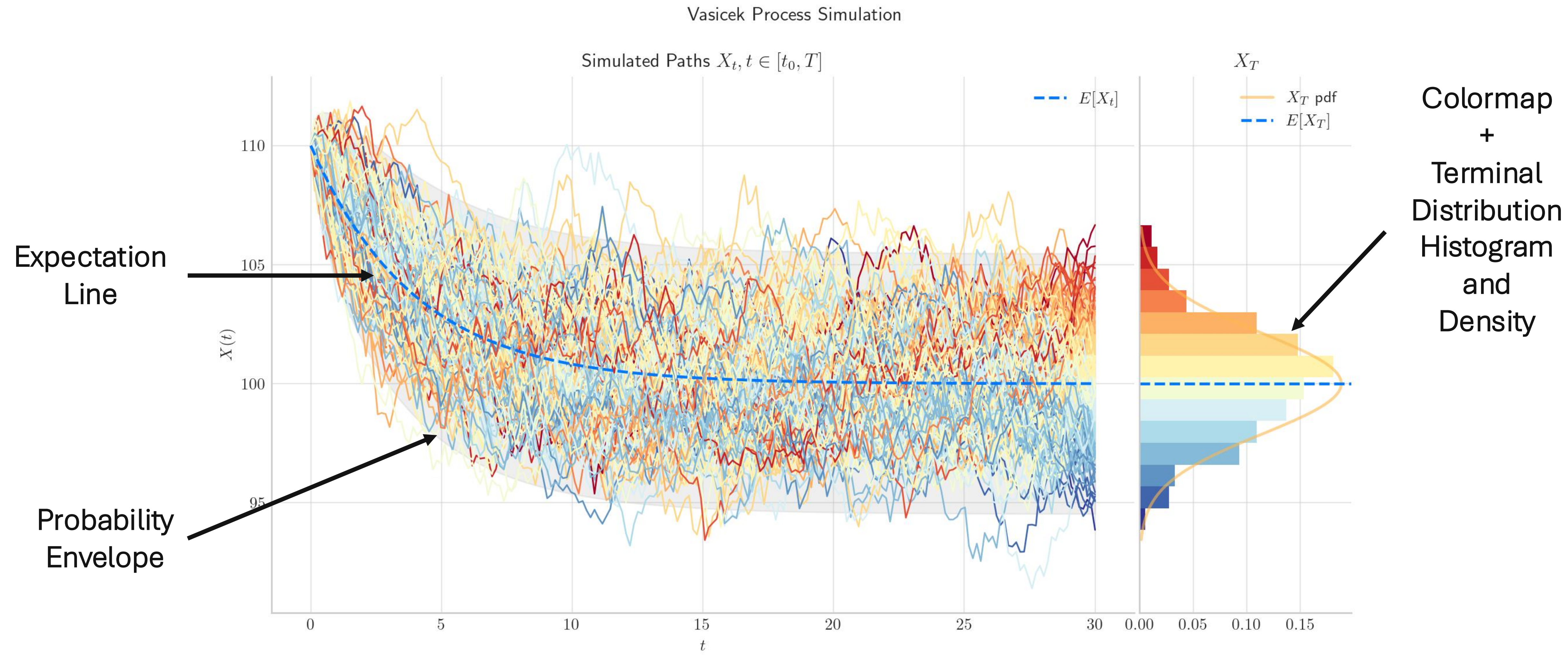
Expectation Line



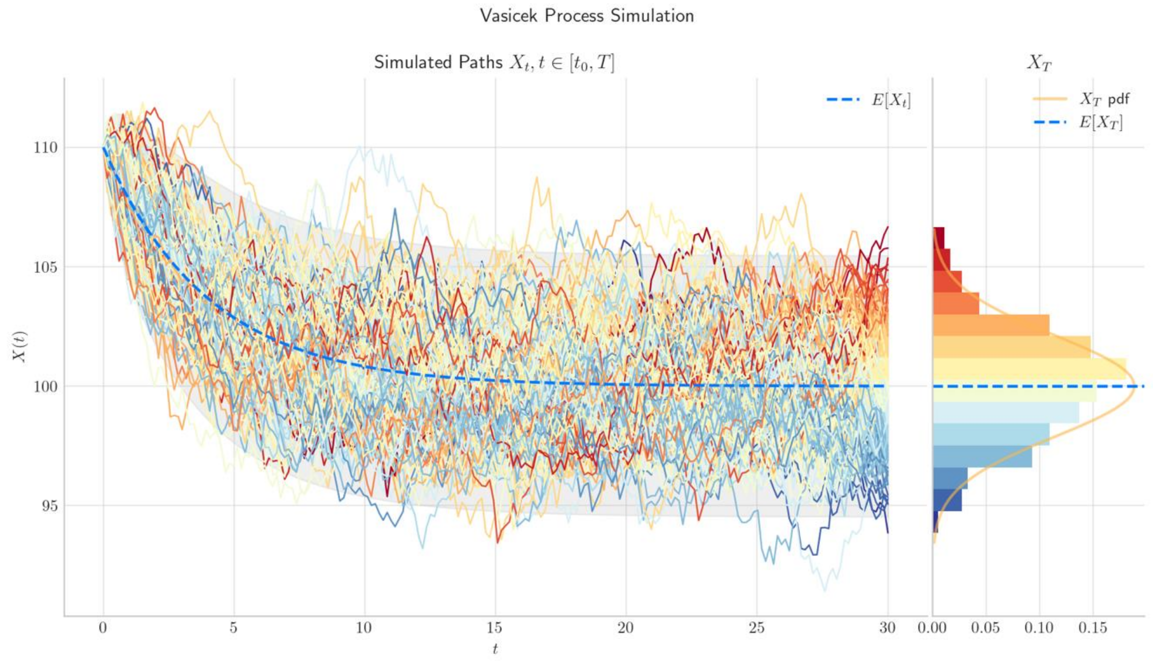
Probability Envelope

DialidSantiago.com

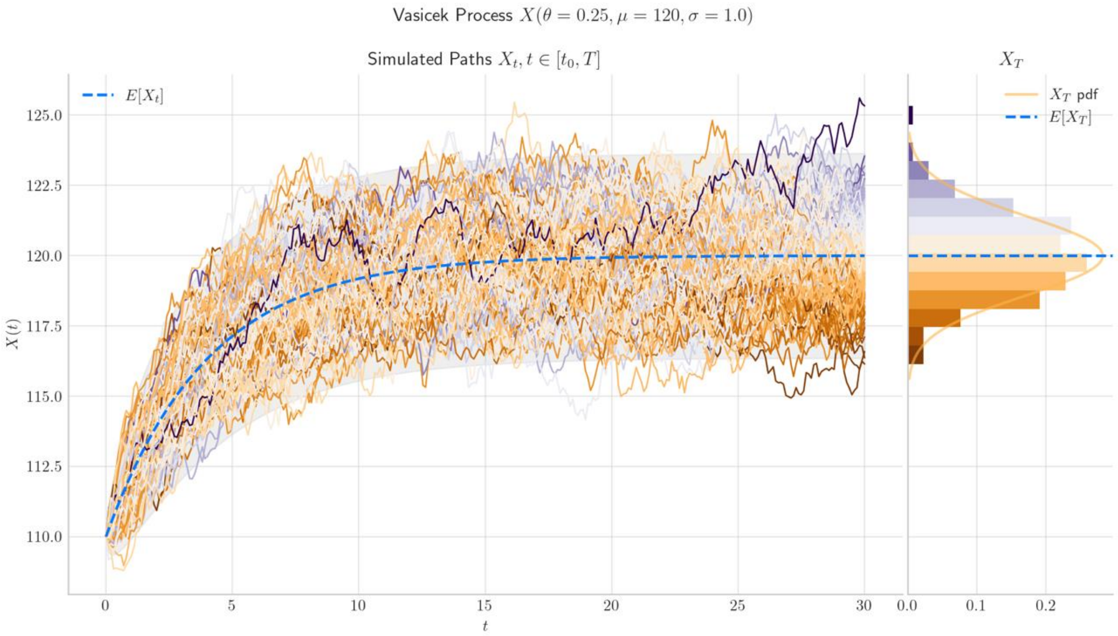
Final Plot
Expectation + Envelope + Colormap + Terminal Distribution

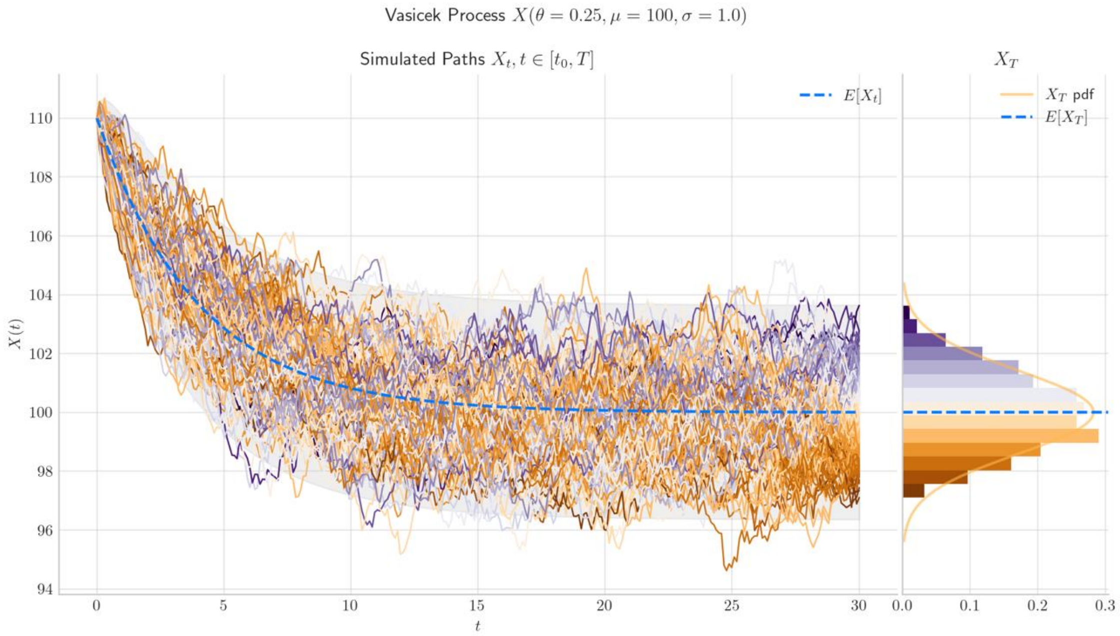


Let’s use this kind of plots to understand the parameters of the process!

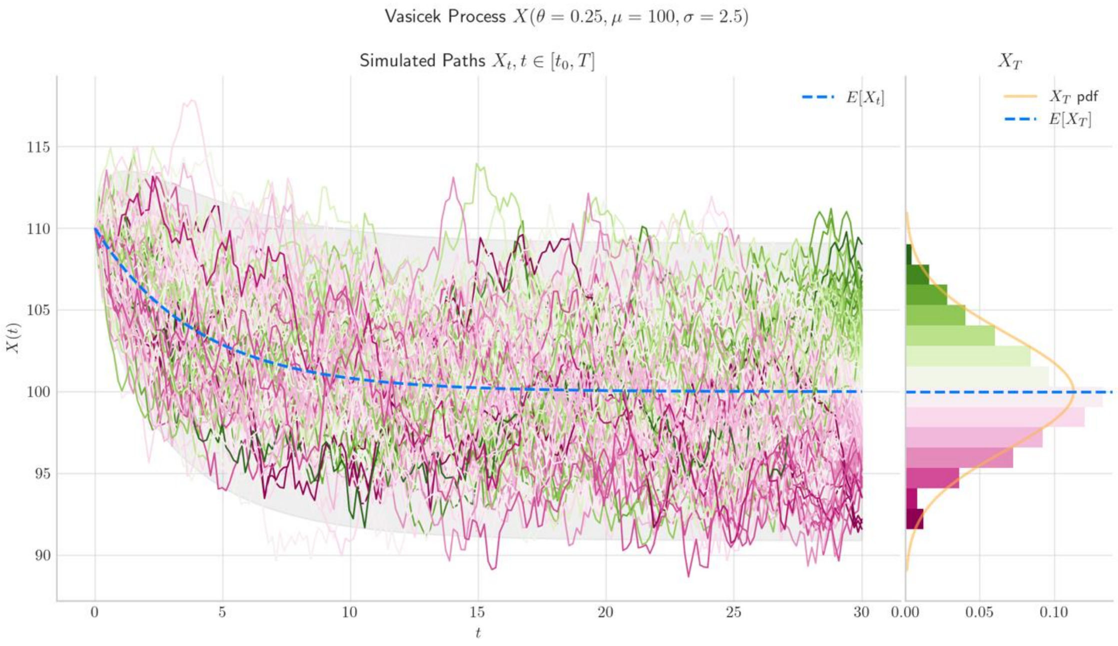


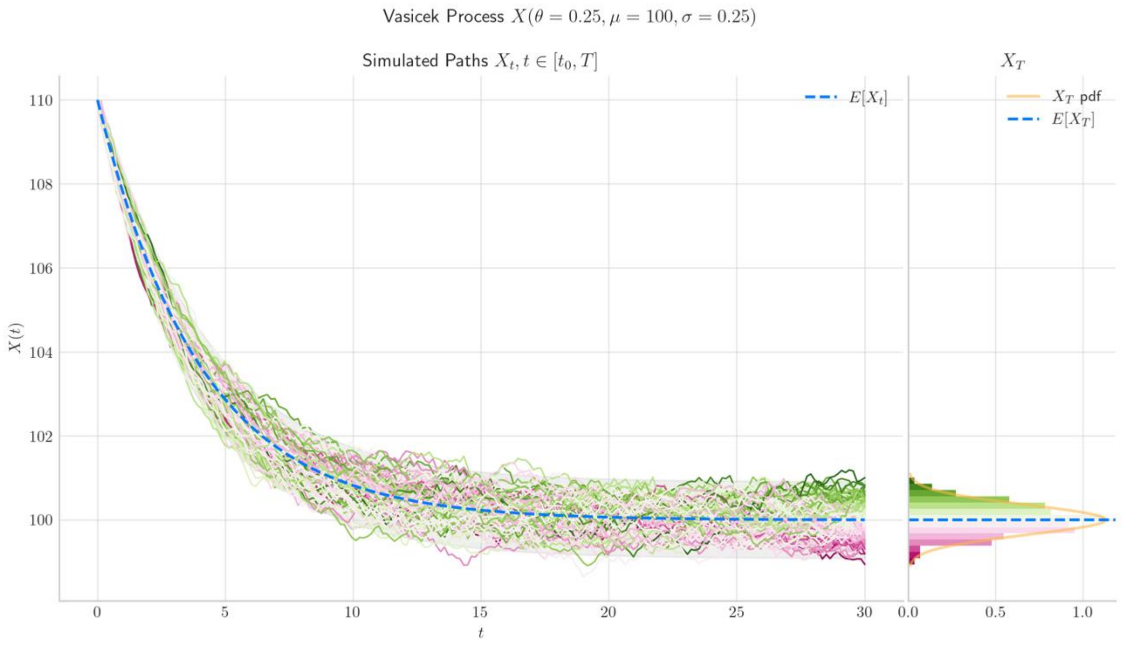
Mu Parameter: Long Term Mean



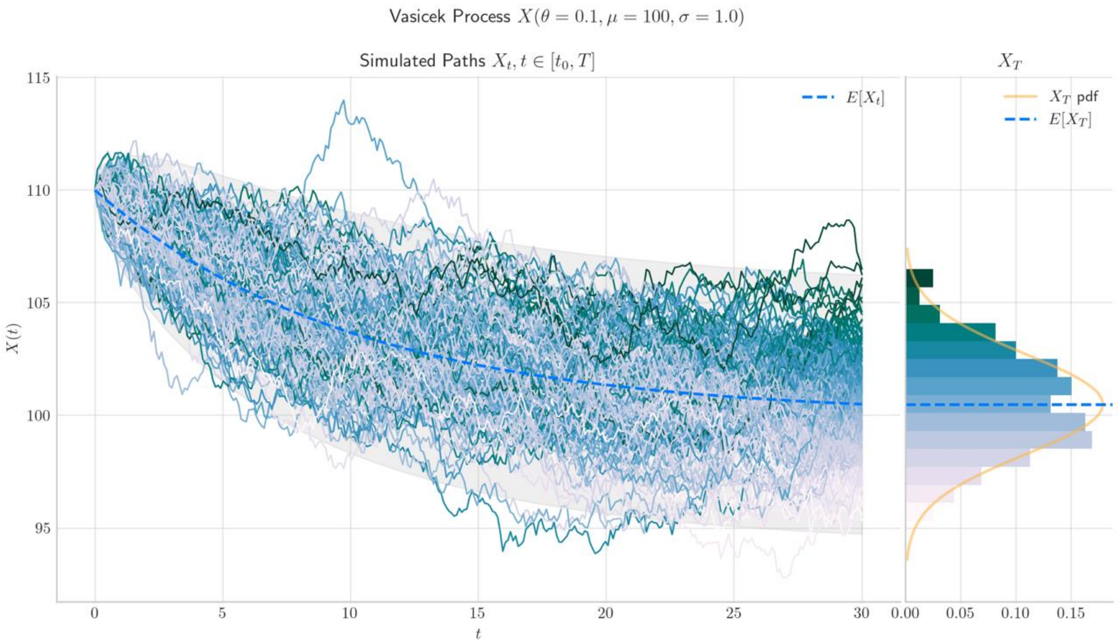


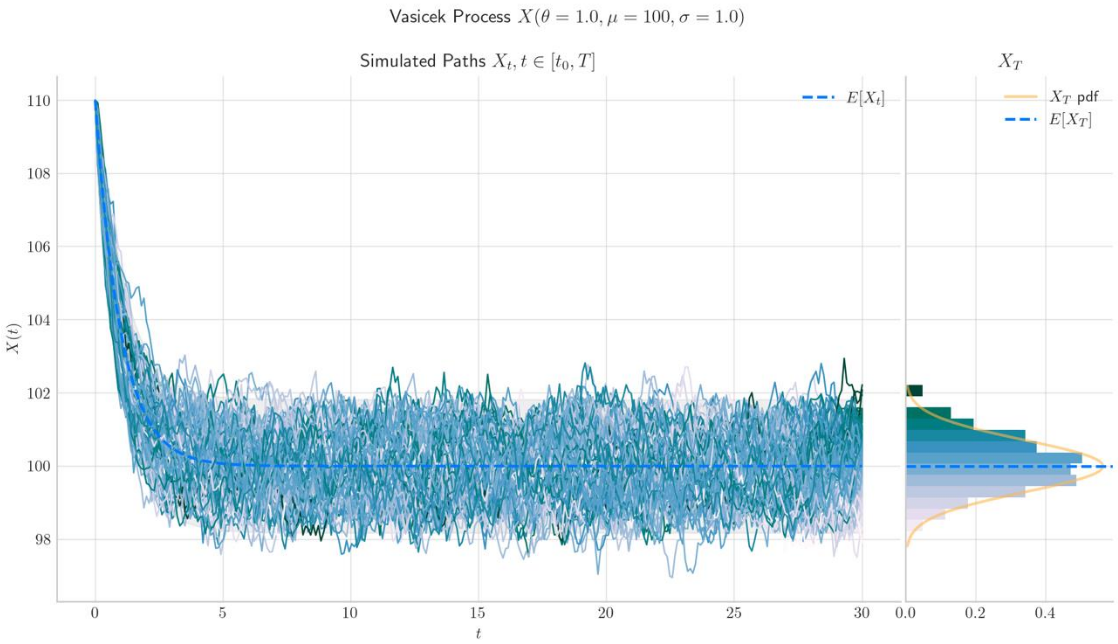
Sigma Parameter: Dispersion





Theta Parameter: Speed





We can expand this idea to other stochastic processes!

Introducing aleatory

aleatory

/ˈeɪlɪət(ə)ri,ˈalɪət(ə)ri/

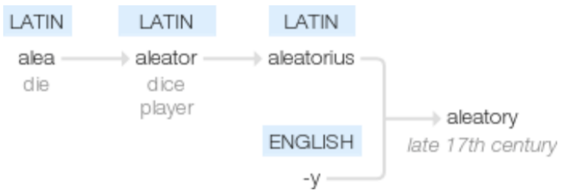
adjective

adjective: **aleatory**; adjective: **aleatoric**

depending on the throw of a dice or on chance; random.

- relating to or denoting music or other forms of art involving elements of random choice (sometimes using statistical or computer techniques) during their composition, production, or performance.
  "aleatory music"

Origin



late 17th century: from Latin *aleatorius*, from *aleator* 'dice player', from *alea* 'die', + -y[1].

A **Python** library to simulate and visualize **stochastic processes**

DialidSantiago.com

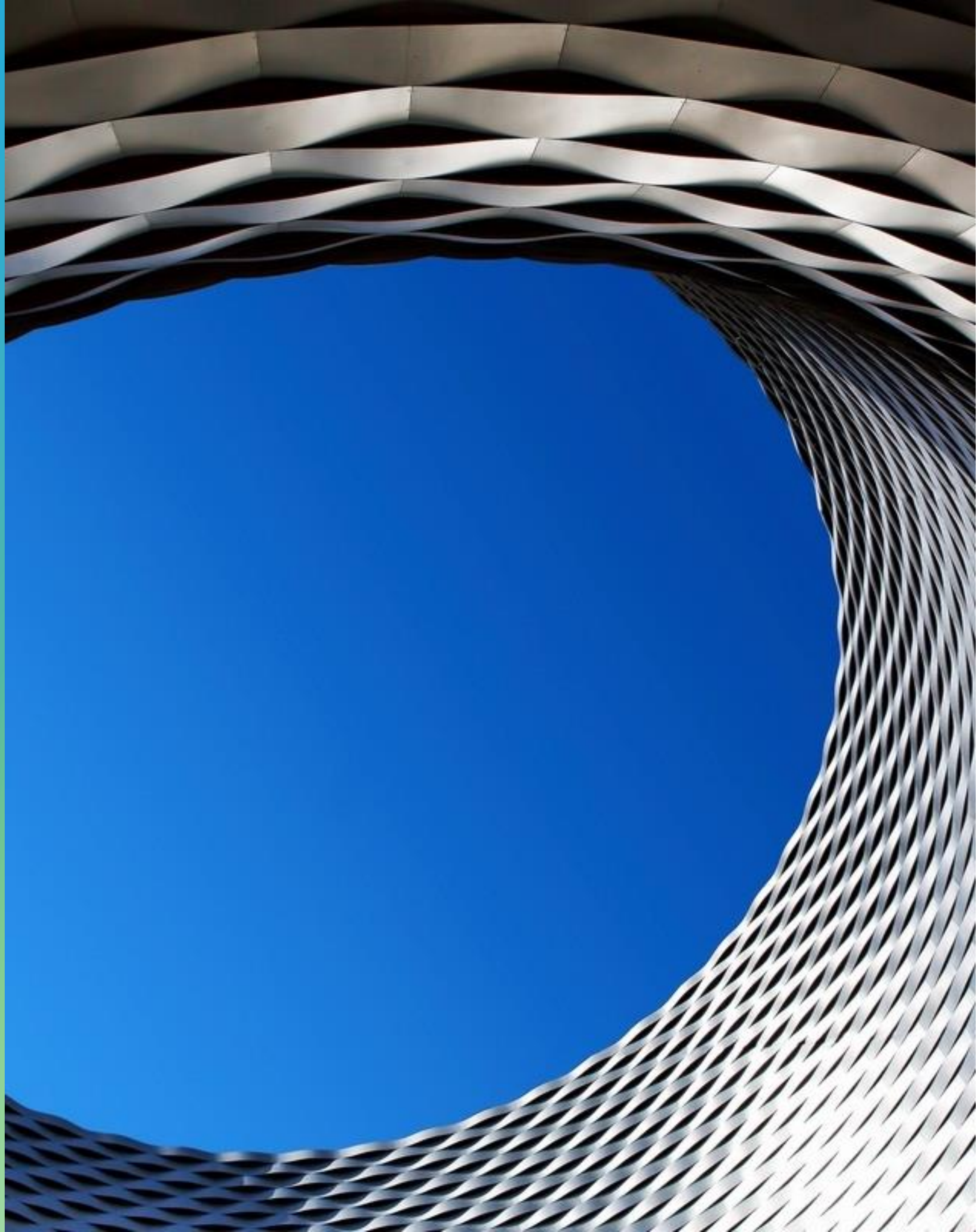
Purpose

Exploratory data analysis

Communication

Education

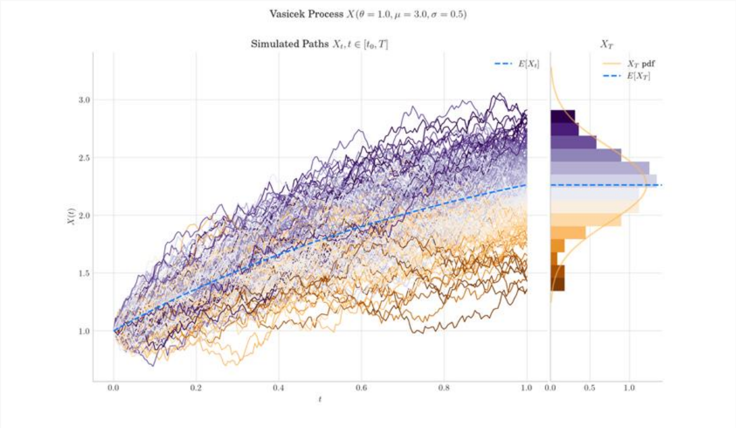
Industry



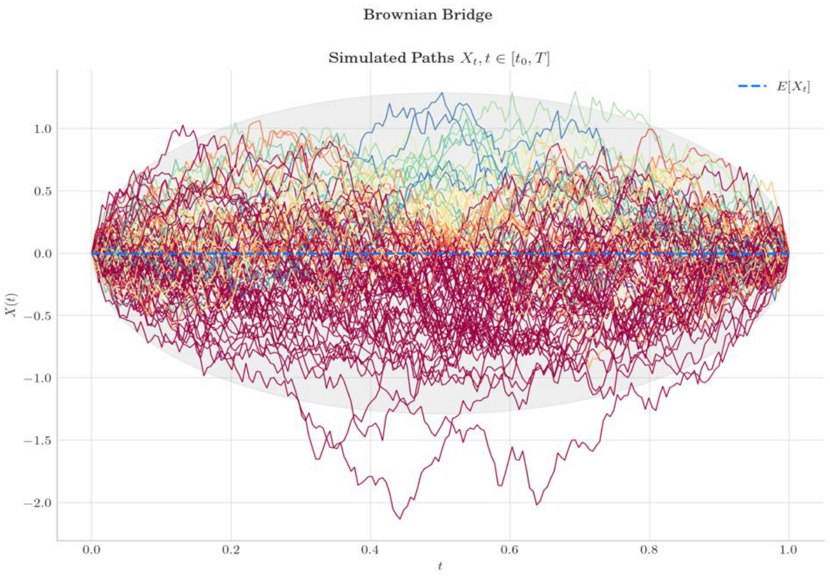
Design

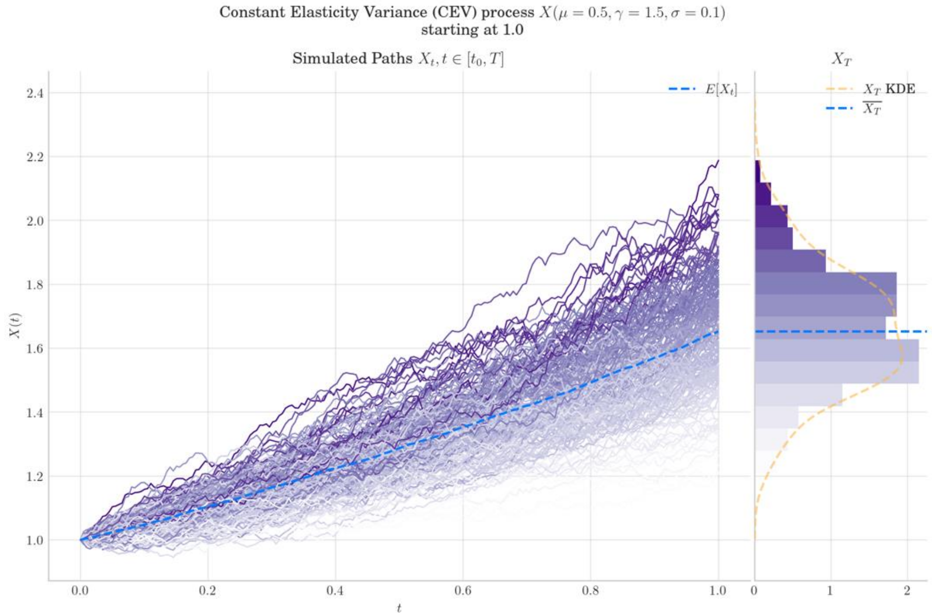
1. Object oriented
2. Cover a wide variety of processes
3. Simple API (three main methods: simulate, plot, draw)
4. Leverage matplotlib features

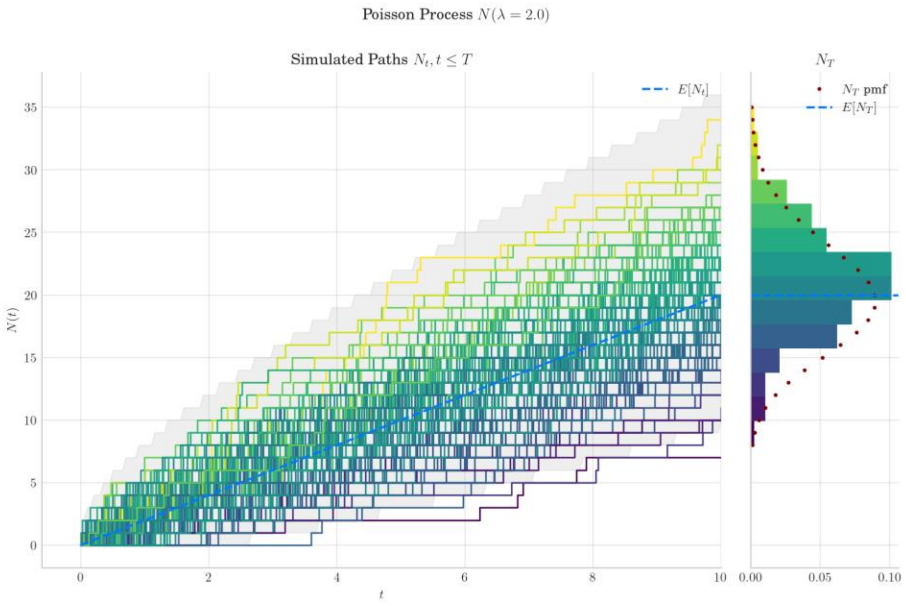
```python
# Author: Dialid Santiago <d.santiago@outlook.com>
# License: MIT
# Description: Simulate and visualise a Vasicek Process

from aleatory.processes import Vasicek
from aleatory.styles import qp_style

qp_style()  # Use quant-pastel-style

p = Vasicek()
fig = p.draw(n=200, N=200, figsize=(12, 7), colormap="PuOr")
fig.show()
```

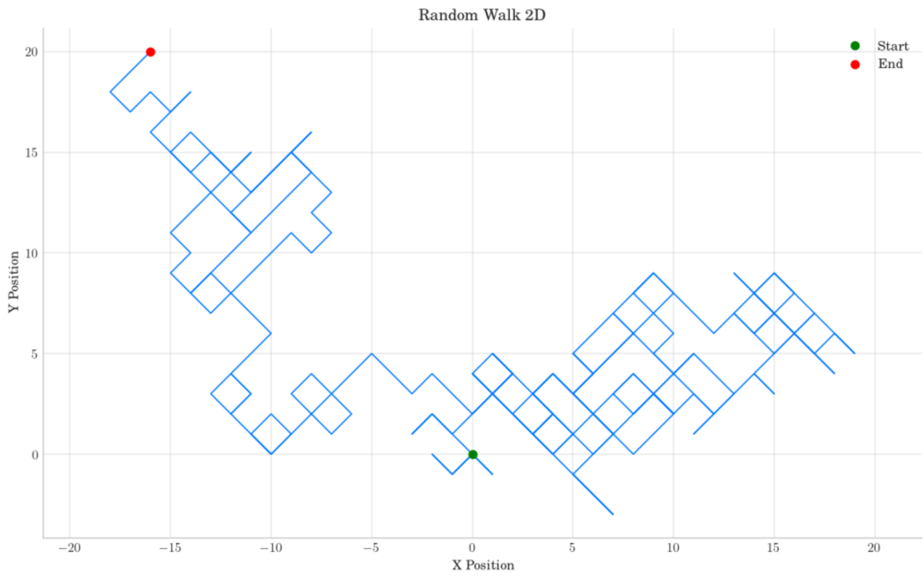


Aleatory Draw Recipe


Brownian Bridge
Simulated Paths $X_t, t \in [t_0, T]$
$E[X_t]$
$X(t)$
t


Constant Elasticity Variance (CEV) process $X(\mu = 0.5, \gamma = 1.5, \sigma = 0.1)$
starting at 1.0
Simulated Paths $X_t, t \in [t_0, T]$
X_T
$E[X_t]$
X_T KDE
$\overline{X_T}$
$X(t)$
t


Poisson Process $N(\lambda = 2.0)$
Simulated Paths $N_t, t \leq T$
N_T
$E[N_t]$
N_T pmf
$E[N_T]$
$N(t)$
t


Random Walk 2D
Start
End
Y Position
X Position

Final Thoughts

- **Stochastic processes** are powerful mathematical objects, but they **don’t have to be intimidating**
- With the right tools, we can tell beautiful **stories about uncertainty**
- aleatory is here to help you

Me 🙋🏽‍♀️

aleatory 💙

Thanks for listening!