Introduction to Data Driven Business Insights

Generating, Visualizing, and Preprocessing Different Types of Time Series

Dr. Stavros K. Stavroglou

December 23, 2024

Outline

Goals of Section 1

- Get an initial taste of what time series look like from different underlying processes
- Compare and contrast three time series:
  1. Brownian Motion (Stochastic)
  2. Lorenz Attractor (Chaotic)
  3. Lotka-Volterra (Deterministic, Predator-Prey)
- Visualize their behavior to see how they differ

Brownian Motion (Random Walk)

Key Idea: A random walk where each step is a draw from some distribution.

Properties:

- Stochastic and unpredictable
- Future value = previous value + random noise
- Often used to model stock prices or physical phenomena (e.g. particle diffusion)

Minimal Code Snippet:

Generating Brownian Motion in Python

```
np.random.seed(42)
n = 300
steps = np.random.normal(loc=0, scale=1, size=n)
brownian = np.cumsum(steps) # Brownian Motion
```

Sample Visualization: Brownian Motion

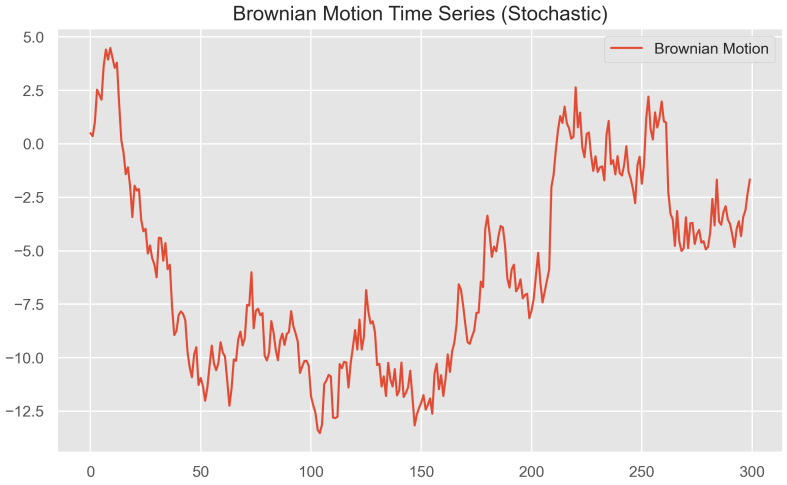


Figure: A realization of a Brownian Motion time series.

Lorenz Attractor (Chaotic System)

Lorenz Equations:

$$\begin{cases} \dot{x} = \sigma(y - x) \\ \dot{y} = x(\rho - z) - y \\ \dot{z} = xy - \beta z \end{cases}$$

Characteristics:

- Deterministic but highly sensitive to initial conditions
- Exhibits chaotic behavior
- Often used as a classic example of chaos (e.g., weather/climate models)

Sample Visualization: Lorenz Attractor (X-component)

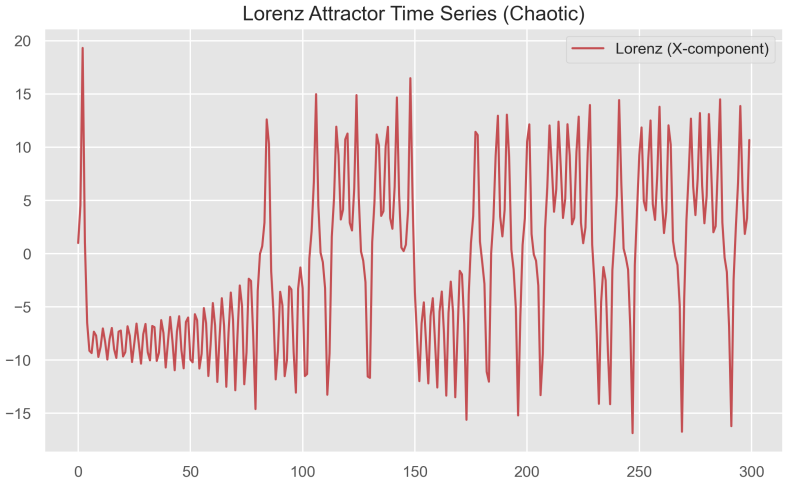


Figure: A single time-series component (x) from the Lorenz attractor.

Lotka-Volterra (Predator-Prey System)

Equations:

$$\begin{cases} \frac{d(\text{prey})}{dt} = \alpha\,\text{prey} - \beta\,\text{prey}\cdot\text{pred} \\ \frac{d(\text{pred})}{dt} = \delta\,\text{prey}\cdot\text{pred} - \gamma\,\text{pred} \end{cases}$$

Interpretation:

- Cyclical oscillations of predator and prey populations
- Deterministic system with non-linear interactions
- Often used in biology and ecology

Sample Visualization: Lotka-Volterra

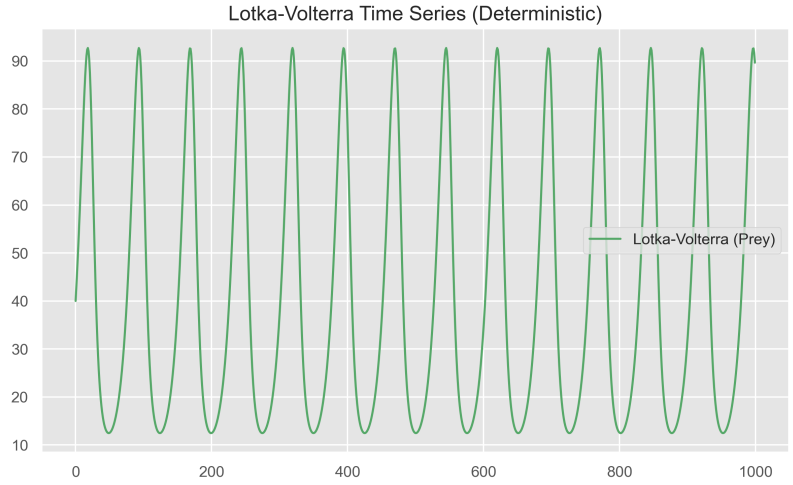


Figure: Time series of prey (green) population.

Visual Comparison: Three Time Series

Figure: Comparison: Brownian (top), Lorenz (middle), Lotka-Volterra (bottom).

Common Approaches to Missing Data

- Real-world data often has gaps
- **Forward Fill (ffill):** Replace missing value with last known value
- **Average of Neighbors:** Replace with mean of immediate neighbors
- **Interpolation:** Estimate values by fitting a function between known points

Pros and Cons of Each Method

Forward Fill

- **Pros:** Very simple, no complicated assumptions
- **Cons:** May create unrealistic plateaus for large gaps

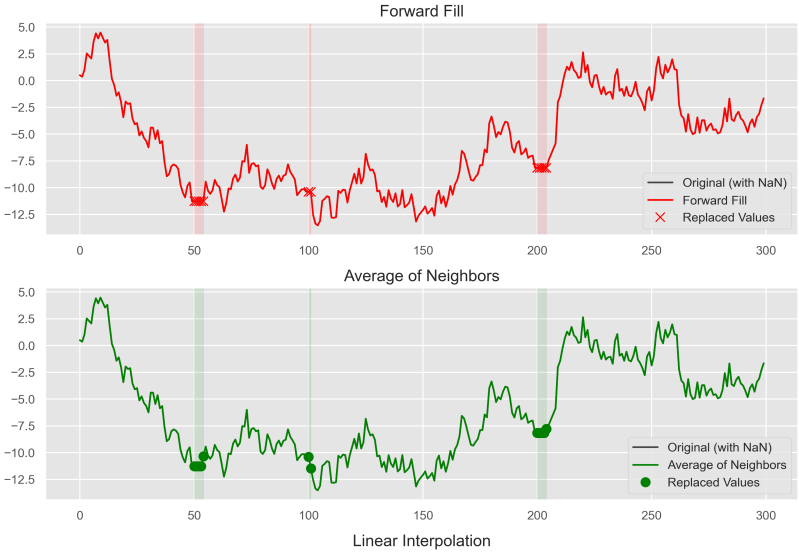
Average of Neighbors

- **Pros:** Smooths out small gaps, simple to compute
- **Cons:** Does not account for larger trends or seasonality

Interpolation

- **Pros:** Creates more "natural" transitions
- **Cons:** Linear interpolation can be too simplistic for non-linear data

Visualizing the Effect of Filling Methods



In the Upcoming Coding Session

- We will demonstrate each approach using `pandas` and `NumPy`
- Compare the filled data with the original
- Understand how the choice of filling strategy can affect subsequent analysis

Motivation for Preprocessing

- Make time series **stationary** or more suitable for forecasting
- **Remove or reduce** trends and seasonality
- Scale data for **machine learning** or statistical models

Common Transformations

(1) Differencing:

$$y'_t = y_t - y_{t-1}$$

- Removes trend, helps achieve stationarity
- Must store initial value to invert

(2) Log-Differencing:

$$y''_t = \ln(y_t) - \ln(y_{t-1})$$

- Stabilizes variance for multiplicative trends
- Requires all values to be positive (may need shifting)

Normalization and Standardization

(1) Normalization (Min-Max):

$$y_t^{\text{norm}} = \frac{y_t - \min(y)}{\max(y) - \min(y)}$$

- Rescales data to $[0, 1]$
- Can be sensitive to out-of-range values

(2) Standardization (Z-Score):

$$y_t^{\text{std}} = \frac{y_t - \mu}{\sigma}$$

- Mean becomes 0, standard deviation becomes 1
- Outliers heavily affect mean and std

Visualizing Preprocessing: Example

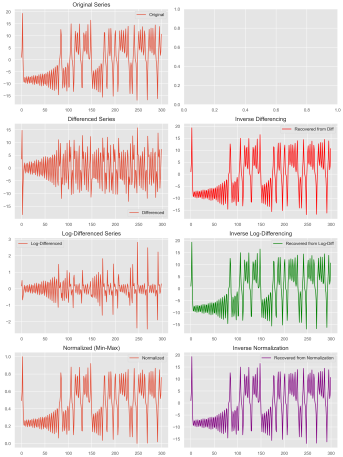


Figure: Different transformations on a Lorenz time series example.

Reversing the Transformations

Why Invert?

- Forecasting is often done in transformed space
- Need to recover original scale for interpretability

Examples:

- **Inverse differencing:** cumulative sum + initial value
- **Inverse log:** exponentiate and subtract shift if applied
- **Inverse normalization:** multiply by range, then add min
- **Inverse standardization:** multiply by σ, then add μ

Understanding Seasonality, Trend, and Residuals

- **Seasonal Decomposition** splits the series into:
  1. Trend
  2. Seasonal component
  3. Residual (unexplained part)
- Helps visualize repeating patterns (e.g., monthly, yearly cycles)
- Informs forecasting approach (e.g., if seasonal patterns are strong)

Sample Visualization: Seasonal Decomposition

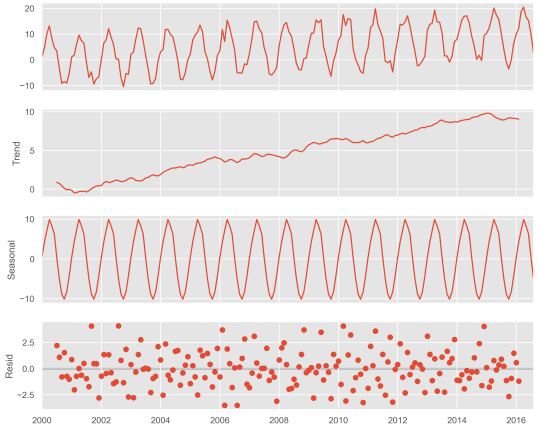


Figure: Additive seasonal decomposition of a synthetic seasonal time series.

Basic Exploratory Data Analysis (EDA)

Key Tools:

- **Rolling Statistics:**
  - Rolling mean (trend)
  - Rolling standard deviation (volatility)
- **Autocorrelation (ACF) and Partial Autocorrelation (PACF)**
  - ACF: correlation of the series with its own past
  - PACF: correlation after removing intermediate lag effects

Why it Matters:

- Detect seasonality (e.g., correlation at lag 12 for monthly data)
- Recognize if the process has patterns useful for forecasting

ACF vs. PACF: Pros and Cons

ACF (Autocorrelation Function)

- **Pros:** Easy to spot periodicities/seasonality
- **Cons:** Includes both direct and indirect correlations

PACF (Partial Autocorrelation)

- **Pros:** Isolates correlation from specific lags, guiding AR order
- **Cons:** More abstract concept for beginners

In Practice: Use both to get a complete picture of the time series structure.

Visualizing ACF/PACF

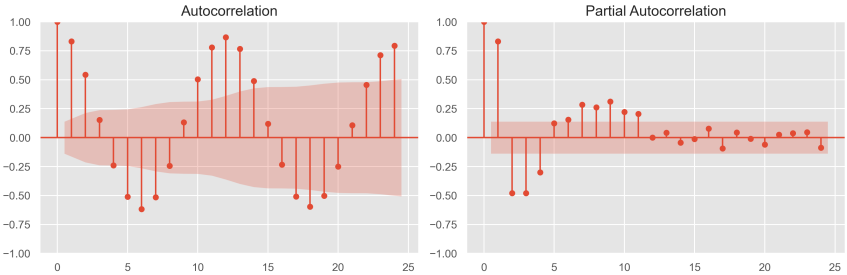


Figure: ACF (left) and PACF (right) plots showing significant lags.

Insights:

- Peaks at certain lags indicate autocorrelation.
- Periodic peaks may hint at seasonality.

Summary of Lecture 1

- **Section 1:** Explored different types of time series (stochastic, chaotic, deterministic)
- **Section 2:** Learned methods to handle missing data and their trade-offs
- **Section 3:** Understood common preprocessing (differencing, log, scaling) and how to invert
- **Section 4:** Performed seasonal decomposition and basic EDA (rolling stats, ACF, PACF)

Key Message:
Properly understanding and preparing your time series data is crucial before forecasting.

Next Steps

- **Hands-On Coding Session:**
  - We will run and dissect the provided Python script
  - Experiment with filling methods, transformations, and decomposition
- **Future Lectures:**
  - Advanced forecasting models (ARIMA, SARIMA, etc.)
  - Stationarity tests (ADF), model evaluation, hyperparameter tuning
  - Handling real-world complexities (irregular sampling, multiple seasonality)

Thank you!

Questions or clarifications before the coding session?