Visual Modeling for Information

Storytelling

M. Sc. Jorge J. Pedrozo Romero
Data Engineering Program

9$^{\text{th}}$ quadrimester September - December (2025)

Chapter 1: Characteristics of Time Series

Time Series Analysis - Python Implementation Guide

Introduction to Time Series

What is Time Series Analysis?

Time series analysis addresses the unique problems that arise when analyzing data observed at different points in time. The key challenge is that **adjacent observations are often correlated**, violating the independence assumption of many classical statistical methods.

Key Characteristics

- **Temporal correlation**: Values at nearby time points are related
- **Time-indexed data**: Observations are ordered by when they occurred
- **Sequential nature**: Past values may influence future values

Applications

Time series analysis appears in diverse fields:

- **Economics**: Stock prices, unemployment rates, GDP
- **Medicine**: Blood pressure over time, EEG signals, fMRI data
- **Environmental Science**: Temperature records, pollution levels
- **Engineering**: Speech signals, seismic data, sensor readings
- **Social Sciences**: Population trends, birth rates

The Nature of Time Series Data

Two Approaches to Time Series Analysis

Time Domain Approach

- Focuses on correlation between values at different times
- Models current values as functions of past values
- Examples: Autoregressive (AR), Moving Average (MA), ARIMA models
- Used for **forecasting** and **prediction**

Frequency Domain Approach

- Analyzes periodic/cyclical patterns in data
- Decomposes series into sinusoidal components
- Uses **spectral analysis** to study periodicities
- Examines variance across different frequencies

> **Note**: These approaches are complementary, not mutually exclusive!

Common Time Series Patterns

Example 1: Trend

- Long-term increase or decrease
- Example: Global temperature rise, company earnings growth

Example 2: Seasonality

- Regular periodic fluctuations
- Example: Quarterly patterns, monthly sales cycles

Example 3: Cycles

- Non-fixed periodic patterns
- Example: Economic cycles, El Niño oscillations

Example 4: Irregular/Random

- Unpredictable short-term fluctuations
- Example: White noise, random shocks

Time Series Statistical Models

Stochastic Processes

A **time series** is formally defined as a collection of random variables indexed by time:

```
{x_t : t ∈ T}
```

where:

- `x_t` = value at time t
- `T` = index set (usually integers: ..., -2, -1, 0, 1, 2, ...)

White Noise

Definition: A sequence of uncorrelated random variables with:

- Mean: $E(w_t) = 0$
- Variance: $Var(w_t) = \sigma^2_w$
- Covariance: $Cov(w_s, w_t) = 0$ for $s \neq t$

Notation: $w_t \sim WN(0, \sigma^2_w)$

Gaussian White Noise: When $w_t \sim N(0, \sigma^2_w)$ (normal distribution)

Properties:

- No predictable pattern
- Completely random
- Foundation for more complex models

Moving Average (MA)

Definition: Smooth white noise by averaging adjacent values

```
v_t = (1/3)[w_{t-1} + w_t + w_{t+1}]
```

General form:

```
x_t = Σ(j=0 to q) θ_j w_{t-j}
```

Effect:

- Reduces high-frequency noise
- Creates smoother appearance
- Introduces correlation between nearby points

Autoregression (AR)

Definition: Current value depends on past values plus noise

```
x_t = φ_1 x_{t-1} + φ_2 x_{t-2} + ... + φ_p x_{t-p} + w_t
```

Example (AR(2)):

```
x_t = x_{t-1} - 0.9x_{t-2} + w_t
```

Properties:

- Creates periodic/cyclical behavior
- Commonly used in forecasting
- Parameters φ control the dynamics

Random Walk

Random Walk (without drift):

```
x_t = x_{t-1} + w_t
```

Random Walk with Drift:

```
x_t = δ + x_{t-1} + w_t
```

Can be rewritten as:

```
x_t = δt + Σ(j=1 to t) w_j
```

Properties:

- **Non-stationary**: Variance increases with time
- Mean function: $\mu_t = \delta t$
- Models trending behavior
- Common in financial data

Signal Plus Noise

Model:

```
x_t = s_t + w_t
```

where:

- s_t = deterministic signal
- w_t = random noise

Example (sinusoidal signal):

```
x_t = A cos(2πωt + φ) + w_t
```

where:

- A = amplitude
- ω = frequency (cycles per time unit)
- φ = phase shift

Signal-to-Noise Ratio (SNR): Higher SNR → easier to detect signal

Measures of Dependence

Mean Function

Definition:

```
μ_t = E(x_t) = ∫_{-∞}^{∞} x f_t(x) dx
```

Interpretation: Expected value of the series at time t

Examples:

- White noise: $\mu_t = 0$ for all t
- Random walk with drift: $\mu_t = \delta t$
- Signal plus noise: $\mu_t = s_t$

Autocovariance Function

Definition:

```
γ(s,t) = Cov(x_s, x_t) = E[(x_s - μ_s)(x_t - μ_t)]
```

Properties:

- Measures **linear dependence** between two time points
- $\gamma(t,t) = Var(x_t)$ (variance)
- $\gamma(s,t) = \gamma(t,s)$ (symmetric)

Examples:

White Noise:

```
γ(s,t) = {σ²_w  if s = t
         {0     if s ≠ t
```

Moving Average (3-point):

```
γ(s,t) = {(3/9)σ²_w   if |s-t| = 0
         {(2/9)σ²_w   if |s-t| = 1
         {(1/9)σ²_w   if |s-t| = 2
         {0           if |s-t| > 2
```

Random Walk:

```
γ(s,t) = min{s,t} × σ²_w
```

Autocorrelation Function

Definition:

```
ρ(s,t) = γ(s,t) / √[γ(s,s) × γ(t,t)]
```

Properties:

- Standardized covariance: **$-1 \le \rho(s,t) \le 1$**
- $\rho(t,t) = 1$ (perfect correlation with itself)
- Measures **predictability** of x_t from x_s

Cross-Covariance and Cross-Correlation

For two series x_t and y_t:

Cross-Covariance:

```
γ_{xy}(s,t) = Cov(x_s, y_t) = E[(x_s - μ_{xs})(y_t - μ_{yt})]
```

Cross-Correlation Function (CCF):

```
ρ_{xy}(s,t) = γ_{xy}(s,t) / √[γ_x(s,s) × γ_y(t,t)]
```

Applications:

- Detect **leading/lagging** relationships
- If $\rho_{xy}(h)$ peaks at $h > 0$: x leads y by h units
- If $\rho_{xy}(h)$ peaks at $h < 0$: x lags y by $|h|$ units

Stationary Time Series

Strict Stationarity

Definition: The joint distribution is **time-invariant**

```
P{x_{t1} ≤ c1, ..., x_{tk} ≤ ck} = P{x_{t1+h} ≤ c1, ..., x_{tk+h} ≤ ck}
```

for all k, all time points $t_1,\ldots,t_k$, all values $c_1,\ldots,c_k$, and all shifts h.

Implication: Statistical properties don't change over time

Weak Stationarity (Covariance Stationarity)

Definition: A process is **weakly stationary** if:

1. **Constant mean**: $\mu_t = \mu$ (independent of t)
2. **Lag-dependent covariance**: $\gamma(s,t)$ depends only on $|s-t|$

Notation for stationary processes:

```
γ(h) = Cov(x_{t+h}, x_t)  [depends only on lag h]
ρ(h) = γ(h) / γ(0)        [ACF as function of lag]
```

Properties of Autocovariance Function

For a stationary process:

1. **At lag 0**: $\gamma(0) = Var(x_t) \geq 0$
2. **Symmetry**: $\gamma(h) = \gamma(-h)$
3. **Bound**: $|\gamma(h)| \leq \gamma(0)$
4. **Non-negative definite**: Ensures valid correlation structure

Examples of Stationarity

Process	Stationary?	Reason
White Noise	Yes	Constant mean (0), covariance depends only on lag
Moving Average	Yes	Constant mean, lag-dependent covariance
Random Walk	**No**	Variance increases with time: $Var(x_t) = t \times \sigma^2_w$
AR(p) with	φ	<1
Trend ($x_t = \beta t$)	**No**	Mean changes with time
Seasonal	**No†**	Mean varies periodically (†unless detrended)

Linear Process

General form:

```
x_t = μ + Σ_{j=-∞}^{∞} ψ_j w_{t-j}
```

where Σ|ψ_j| < ∞

Autocovariance:

```
γ(h) = σ²_w Σ_{j=-∞}^{∞} ψ_{j+h} ψ_j
```

Important: Many time series models (MA, AR, ARMA) are linear processes

Gaussian (Normal) Processes

Definition: All finite-dimensional distributions are multivariate normal

Key fact: For Gaussian processes:

- Weak stationarity $\Longrightarrow$ Strict stationarity
- Zero covariance $\Longrightarrow$ Independence

Multivariate Normal Density:

```
f(x) = (2π)^{-n/2} |Γ|^{-1/2} exp[-½(x-μ)ᵀΓ⁻¹(x-μ)]
```

Estimation of Correlation

Sample Mean

Estimator:

```
x̄ = (1/n) Σ_{t=1}^n x_t
```

Standard Error:

```
SE(x̄) = √Var(x̄) = √[(1/n) Σ_{h=-n}^n (1 - |h|/n)γ(h)]
```

Special case (white noise): $SE(\bar{x}) = \sigma_x / \sqrt{n}$

Sample Autocovariance Function

Estimator:

```
γ̂(h) = (1/n) Σ_{t=1}^{n-h} (x_{t+h} - x̄)(x_t - x̄)
```

Properties:

- Biased but consistent
- Non-negative definite (dividing by n, not n-h)
- $\hat{\gamma}(-h) = \hat{\gamma}(h)$

Sample Autocorrelation Function (ACF)

Estimator:

```
ρ̂(h) = γ̂(h) / γ̂(0)
```

Large-Sample Distribution (for white noise):

```
ρ̂(h) ~ N(0, 1/n)  approximately, for h = 1,2,...,H
```

95% Confidence Bounds: $\pm 2/\sqrt{n}$

Interpretation: If $|\hat{\rho}(h)| > 2/\sqrt{n}$, significant correlation at lag h

Sample Cross-Correlation Function

Cross-Covariance Estimator:

```
γ̂_{xy}(h) = (1/n) Σ_{t=1}^{n-h} (x_{t+h} - x̄)(y_t - ȳ)
```

Cross-Correlation Estimator:

```
ρ̂_{xy}(h) = γ̂_{xy}(h) / √[γ̂_x(0) × γ̂_y(0)]
```

Large-Sample Distribution (under independence):

```
ρ̂_{xy}(h) ~ N(0, 1/n)
```

Python Implementation

Required Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy import signal, stats
from statsmodels.tsa.stattools import acf, pacf, ccf
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
import seaborn as sns

# Set style
plt.style.use('seaborn-v0_8-darkgrid')
sns.set_palette("husl")
```

Generating White Noise

```python
def generate_white_noise(n=500, mean=0, std=1, seed=None):
    """
    Generate white noise series

    Parameters:
    -----------
    n : int
        Number of observations
    mean : float
        Mean of the process
    std : float
        Standard deviation
    seed : int
        Random seed for reproducibility
    """
    if seed:
        np.random.seed(seed)

    wn = np.random.normal(mean, std, n)
    return wn

# Example
wn = generate_white_noise(n=500, seed=42)
```

Moving Average Process

```python
def moving_average(wn, weights=None):
    """
    Apply moving average filter

    Parameters:
    -----------
    wn : array
        Input series (typically white noise)
    weights : array
        MA weights (default: 3-point average)
    """
    if weights is None:
        weights = np.array([1/3, 1/3, 1/3])

    # Use convolution for MA
    ma = np.convolve(wn, weights, mode='same')
    return ma

# Example: 3-point MA
wn = generate_white_noise(500)
ma_series = moving_average(wn)
```

Autoregressive Process

```python
def generate_ar(n=500, phi=[1, -0.9], sigma=1, seed=None):
    """
    Generate AR(p) process

    Parameters:
    -----------
    n : int
        Number of observations
    phi : list
        AR coefficients [1, phi_1, phi_2, ...]
    sigma : float
        Noise standard deviation
    """
    if seed:
        np.random.seed(seed)

    # Generate white noise
    wn = np.random.normal(0, sigma, n + 50)  # Extra for burn-in

    # Generate AR process using lfilter
    ar_series = signal.lfilter([1], phi, wn)

    # Remove burn-in
    return ar_series[50:]

# Example: AR(2)
ar2 = generate_ar(n=500, phi=[1, 1, -0.9], seed=42)
```

Random Walk

```python
def random_walk(n=200, drift=0, sigma=1, x0=0, seed=None):
    """
    Generate random walk with drift

    Parameters:
    -----------
    n : int
        Number of observations
    drift : float
        Drift parameter δ
    sigma : float
        Standard deviation of innovations
    x0 : float
        Initial value
    """
    if seed:
        np.random.seed(seed)

    # Generate innovations
    wn = np.random.normal(0, sigma, n)

    # Cumulative sum
    rw = x0 + drift * np.arange(1, n+1) + np.cumsum(wn)

    return rw

# Example
rw = random_walk(n=200, drift=0.2, seed=42)
```

Signal Plus Noise

```
def signal_plus_noise(n=200, A=2, omega=1/50, phi=0.6*np.pi,
                      sigma=1, seed=None):
    """
    Generate signal plus noise model

    Parameters:
    -----------
    n : int
        Number of observations
    A : float
        Amplitude
    omega : float
        Frequency
    phi : float
        Phase shift
    sigma : float
        Noise standard deviation
    """
    if seed:
        np.random.seed(seed)

    t = np.arange(1, n+1)
    signal = A * np.cos(2 * np.pi * omega * t + phi)
    noise = np.random.normal(0, sigma, n)

    return signal + noise, signal, noise

# Example
x, signal, noise = signal_plus_noise(n=200, sigma=1, seed=42)
```

Computing Sample ACF

```python
def compute_acf(x, nlags=40, alpha=0.05):
    """
    Compute sample ACF with confidence intervals

    Parameters:
    -----------
    x : array
        Time series data
    nlags : int
        Number of lags
    alpha : float
        Significance level for confidence interval

    Returns:
    --------
    acf_vals : array
        ACF values
    confint : array
        Confidence intervals
    """
    from statsmodels.tsa.stattools import acf as sm_acf

    acf_vals = sm_acf(x, nlags=nlags, fft=True)

    # Confidence interval (approximate)
    ci = stats.norm.ppf(1 - alpha/2) / np.sqrt(len(x))

    return acf_vals, ci

# Example
wn = generate_white_noise(500, seed=42)
acf_vals, ci = compute_acf(wn, nlags=40)
```

Computing Sample CCF

```python
def compute_ccf(x, y, nlags=40):
    """
    Compute sample cross-correlation function

    Parameters:
    -----------
    x, y : arrays
        Two time series
    nlags : int
        Number of lags (both positive and negative)

    Returns:
    --------
    lags : array
        Lag values
    ccf_vals : array
        CCF values
    """
    # Standardize series
    x_std = (x - np.mean(x)) / np.std(x)
    y_std = (y - np.mean(y)) / np.std(y)

    # Compute cross-correlation
    ccf_vals = np.correlate(x_std, y_std, mode='full') / len(x)

    # Get lags
    lags = np.arange(-nlags, nlags + 1)
    mid = len(ccf_vals) // 2
    ccf_vals = ccf_vals[mid-nlags:mid+nlags+1]

    return lags, ccf_vals

# Example
x = generate_white_noise(500, seed=42)
y = generate_white_noise(500, seed=123)
lags, ccf_vals = compute_ccf(x, y, nlags=20)
```

Visualization Functions

```
def plot_time_series(x, title='Time Series', figsize=(12, 4)):
    """Plot time series"""
    plt.figure(figsize=figsize)
    plt.plot(x, linewidth=1)
    plt.title(title, fontsize=14, fontweight='bold')
    plt.xlabel('Time')
    plt.ylabel('Value')
    plt.grid(True, alpha=0.3)
    plt.tight_layout()
    plt.show()

def plot_acf_custom(x, nlags=40, title='Autocorrelation Function'):
    """Plot ACF with confidence bands"""
    acf_vals, ci = compute_acf(x, nlags=nlags)

    fig, ax = plt.subplots(figsize=(12, 4))

    # Plot ACF
    ax.stem(range(len(acf_vals)), acf_vals, linefmt='C0-',
            markerfmt='C0o', basefmt='C0-')

    # Confidence bands
    ax.axhline(y=ci, linestyle='--', color='red', alpha=0.5)
    ax.axhline(y=-ci, linestyle='--', color='red', alpha=0.5)
    ax.axhline(y=0, linestyle='-', color='black', linewidth=0.8)

    ax.set_xlabel('Lag')
    ax.set_ylabel('ACF')
    ax.set_title(title, fontsize=14, fontweight='bold')
    ax.grid(True, alpha=0.3)

    plt.tight_layout()
    plt.show()

def plot_ccf_custom(x, y, nlags=40, title='Cross-Correlation Function'):
    """Plot CCF"""
    lags, ccf_vals = compute_ccf(x, y, nlags=nlags)

    fig, ax = plt.subplots(figsize=(12, 4))

    ax.stem(lags, ccf_vals, linefmt='C1-',
            markerfmt='C1o', basefmt='C1-')

    # Confidence bands
    ci = 2 / np.sqrt(len(x))
    ax.axhline(y=ci, linestyle='--', color='red', alpha=0.5)
    ax.axhline(y=-ci, linestyle='--', color='red', alpha=0.5)
    ax.axhline(y=0, linestyle='-', color='black', linewidth=0.8)

    ax.set_xlabel('Lag')
    ax.set_ylabel('CCF')
    ax.set_title(title, fontsize=14, fontweight='bold')
    ax.grid(True, alpha=0.3)

    plt.tight_layout()
    plt.show()
```

Key Takeaways

Fundamental Concepts

1. **Time series data is special** because adjacent observations are correlated
2. **Two complementary approaches**: time domain and frequency domain
3. **Stationarity is crucial** for most analysis methods

Important Models

4. **White noise**: Foundation for all models (random, uncorrelated)
5. **Moving averages**: Smooth data by averaging
6. **Autoregressions**: Current value depends on past values
7. **Random walks**: Non-stationary, variance grows with time

Correlation Measures

8. **ACF measures dependence** between observations at different lags
9. **For white noise**: ACF ≈ 0 for all lags except 0
10. **95% confidence bands**: $\pm 2/\sqrt{n}$

Practical Guidelines

11. **Always plot your data first** to identify patterns
12. **Check for stationarity** before applying many methods
13. **Use ACF to detect** correlation structure
14. **Use CCF to detect** leading/lagging relationships

References

- Shumway, R.H. & Stoffer, D.S. (2017). *Time Series Analysis and Its Applications: With R Examples*. Springer.
- Box, G.E.P., Jenkins, G.M., Reinsel, G.C., & Ljung, G.M. (2015). *Time Series Analysis: Forecasting and Control*. Wiley.
- Brockwell, P.J. & Davis, R.A. (2016). *Introduction to Time Series and Forecasting*. Springer.

Python Libraries Reference

Essential packages for time series in Python:

- `numpy`: Numerical computing
- `pandas`: Data manipulation
- `matplotlib` / `seaborn`: Visualization
- `statsmodels`: Statistical models and tests
- `scipy`: Scientific computing
- `scikit-learn`: Machine learning (for advanced topics)