Modeling and Analysis of Time Series Data
Chapter 3: Stationarity, white noise, and some basic time series models

STATS 531, Winter 2025

Edward Ionides

Concepts of stationarity

Definition: A time series model which is both mean stationary and covariance stationary is **weakly stationary** or **second order stationary**. A time series model for which all joint distributions are invariant to shifts in time is **strongly stationary** or **strictly stationary**.

- Formally, this means that for any collection of times $(t_1, t_2, \ldots, t_K)$, the joint distribution of observations at these times should be the same as the joint distribution at $(t_1 + \tau, t_2 + \tau, \ldots, t_K + \tau)$ for any τ.
- For equally spaced observations, this becomes: for any collection of timepoints $n_1, \ldots, n_K$, and for any lag h, the joint density function of $(Y_{n_1}, Y_{n_2}, \ldots, Y_{n_K})$ is the same as the joint density function of $(Y_{n_1+h}, Y_{n_2+h}, \ldots, Y_{n_K+h})$.

- In our general notation for densities, this strict stationarity requirement can be written as

$$f_{Y_{n_1},Y_{n_2},\ldots,Y_{n_K}}(y_1, y_2, \cdots, y_K) = f_{Y_{n_1+h},Y_{n_2+h},\ldots,Y_{n_K+h}}(y_1, y_2, \cdots, y_K). \quad (1)$$

- Strict stationarity implies weak stationarity (check this).

The moment requirements for weak stationarity correspond to specific integrals being equal over the joint densities.

If the densities are shift-invariant, all integrals over them are shift-invariant.

Assessing stationarity

Question. How could we assess whether a weak stationary model is appropriate for a time series dataset?

We can check the sample ACF is similar on different subsets of the data. Visually, we can check whether the data look consistent with a trend.

Question. How could we assess whether a strictly stationary model is appropriate for a time series dataset?

Too many joint distributions to check.
Usually an assumption not a conclusion.

Prevalence of stationarity

Question. Is it usual for time series to be well modeled as stationary (either weakly or strictly)?

Quite commonly, something changes through time. That may be what we are studying.

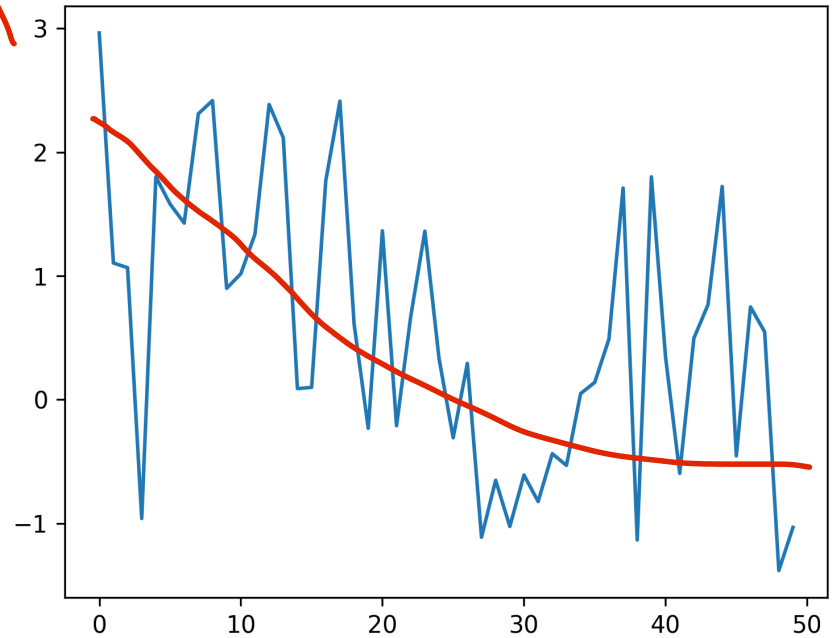
Question. If data often do not show stationary behavior, why do many fundamental models have stationarity?

Build non-stationary models using simpler stationary building blocks.

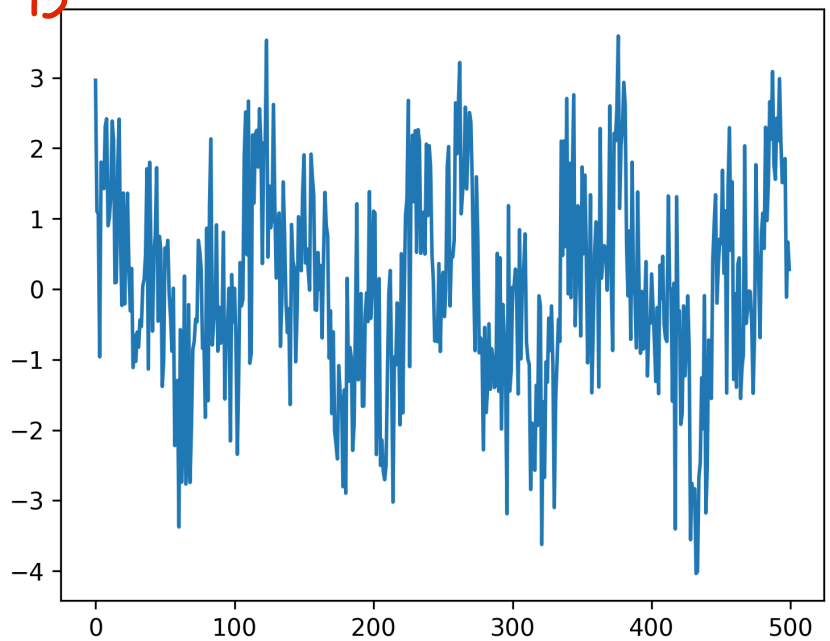
E.g. trend + stationary errors for a regression model.

Question. Is a stationary model appropriate for either (or both) the time series below? Explain.

A



B



A 0-many.

B many-3.

A is the 1st 50 time points of B.

mathematically, they must both be equally stationary or non-stationary.

what is a reasonable model may depend on how long you see it for.

White noise

Definition: A time series model $\epsilon_{1:N}$ which is weakly stationary with

$$\begin{aligned} \mathrm{E}[\epsilon_n] &= 0 \\ \mathrm{Cov}(\epsilon_m, \epsilon_n) &= \begin{cases} \sigma^2, & \text{if } m = n \\ 0, & \text{if } m \neq n \end{cases}, \end{aligned}$$

is said to be **white noise** with variance σ^2.

- "Noise" is because there's no pattern, just random variation. If you listened to a realization of white noise as an audio file, you would hear a static sound.
- "White" is because all frequencies are equally represented. This will become clear when we do frequency domain analysis of time series.
- Signal processing—sending and receiving signals on noisy channels—was a motivation for early time series analysis.

Example: Gaussian white noise

In time series analysis, a sequence of independent identically distributed (iid) Normal random variables with mean zero and variance σ^2 is known as **Gaussian white noise**. We write this model as

$$\epsilon_{1:N} \sim \text{iid}\, N[0, \sigma^2].$$

Example: Binary white noise

Let $\epsilon_{1:N}$ be iid with

$$\epsilon_n = \begin{cases} 1, & \text{with probability } 1/2 \\ -1, & \text{with probability } 1/2 \end{cases}.$$

We can check that $\mathrm{E}[\epsilon_n] = 0$, $\mathrm{Var}(\epsilon_n) = 1$ and $\mathrm{Cov}(\epsilon_m, \epsilon_n) = 0$ for $m \neq n$. Therefore, $\epsilon_{1:N}$ is white noise.

Similarly, for any $p \in (0, 1)$, we could have

$$\epsilon_n = \begin{cases} (1-p)/p, & \text{with probability } p \\ -1, & \text{with probability } 1-p \end{cases}.$$

Example: Sinusoidal white noise

Let $\epsilon_n = \sin(2\pi n U)$, with a single draw $U \sim \text{Uniform}[0,1]$ determining the time series model for all $n \in 1:N$. We will show this is an example of a weakly stationary time series that is not strictly stationary.

Question. Show that $\epsilon_{1:N}$ is weakly stationary, and is white noise!

$$\mathbb{E}[\epsilon_n] = \int_0^1 \sin(2\pi n u)\,du = 0 \quad \text{(integrating a sinusoid over a complete number of cycles).}$$

$$\text{Cov}(\epsilon_m, \epsilon_n) = \mathbb{E}[\epsilon_m \epsilon_n] = \int_0^1 \sin(2\pi m u)\sin(2\pi n u)\,du$$

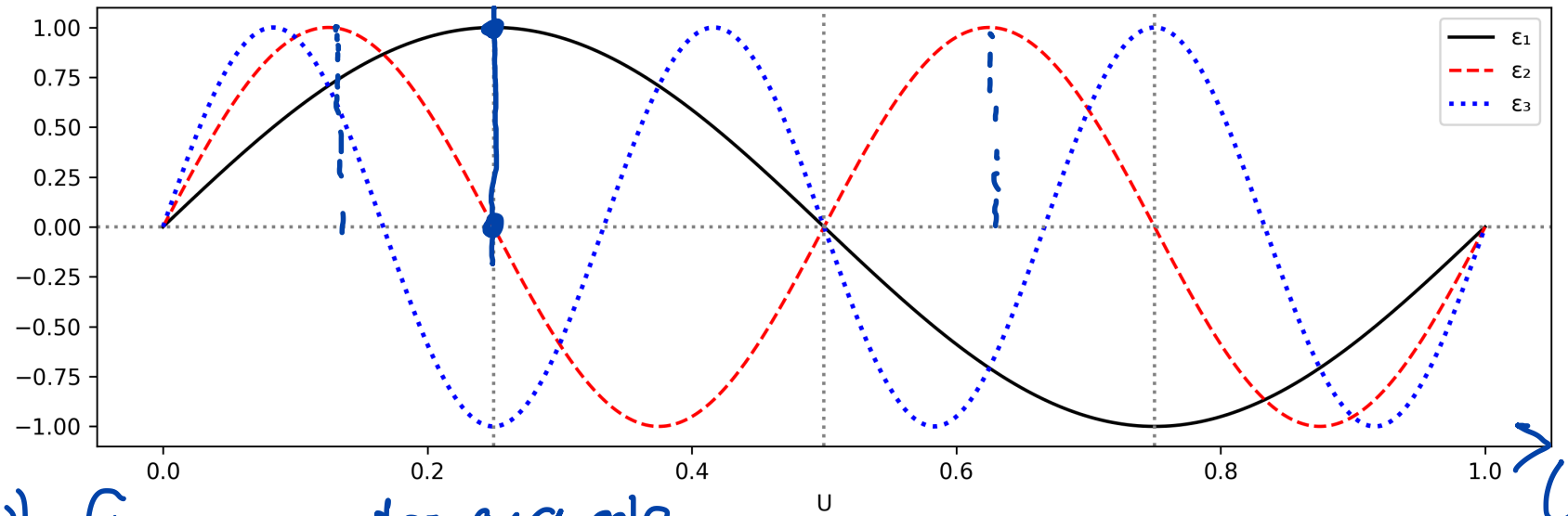
$$= \frac{1}{2}\int_0^1 \left[\cos(2\pi u(m-n)) - \cos(2\pi u(m+n))\right] du$$

$$= \begin{cases} 0 & m \neq n \\ 1/2 & m = n \end{cases}$$

This shows that $\{\epsilon_n\}$ is white noise.

Question. Show that $\epsilon_{1:N}$ is NOT strictly stationary.

Hint: consider the following plot of $\epsilon_{1:3}$ as a function of U.



Look for a counter-example.

$\mathbb{P}[\epsilon_2 = 0 \mid \epsilon_1 = 1] = 1$ since $\{\epsilon_1 = 1\} = \{U = \frac{1}{4}\}$

$\mathbb{P}[\epsilon_3 = 0 \mid \epsilon_2 = 1] = 0$

so, joint probabilities are not shift-invariant, and $\{\epsilon_n\}$ is not strictly stationary.

Building time series models using white noise

Reminder: Why do we need time series models?

- All statistical tests (i.e., whenever we use data to answer a question) rely on having a model for the data. The model is sometimes called the **assumptions** for the test.
- If our model is wrong, then any conclusions drawn from it may be wrong. Our error could be small and insignificant, or disastrous.
- Time series data collected close in time are often more similar than a model with iid variation would predict. We need models that have this property, and we must work out how to test interesting hypotheses for these models.

The AR(p) autoregressive model

- The order p autoregressive model, abbreviated to AR(p), is

[M1] $$Y_n = \phi_1 Y_{n-1} + \phi_2 Y_{n-2} + \cdots + \phi_p Y_{n-p} + \epsilon_n,$$

where $\{\epsilon_n\}$ is a white noise process.

- Often, we consider the **Gaussian AR(p)** model, where $\{\epsilon_n\}$ is a Gaussian white noise process.
- M1 is a **stochastic difference equation**. It is a difference equation (also known as a recurrence relation) since each time point is specified recursively in terms of previous time points. Stochastic just means random.
- M1 is centered, with mean zero. We can add a mean parameter.

Initialization of AR models

- To complete the model, we need to **initialize** the solution to the stochastic difference equation. Supposing we want to specify a distribution for $Y_{1:N}$, we have some choices in how to set up the **initial values**.

1. We can specify $Y_{1:p}$ explicitly, to get the recursion started.
2. We can specify $Y_{1-p:0}$ explicitly.
3. For either of these choices, we can define these initial values either to be additional parameters in the model (i.e., not random) or to be specified random variables.
4. If we want our model is strictly stationary, we must initialize so that $Y_{1:p}$ have the proper joint distribution for this stationary model.

AR(1) example

- Assuming the initialization has mean zero, M1 implies that $\mathrm{E}[Y_n] = 0$ for all n. For additional generality, we could add a constant mean μ.
- Let's investigate a particular Gaussian AR(1) process, as an exercise:
  [M2] $$Y_n = 0.6 Y_{n-1} + \epsilon_n,$$
  where $\epsilon_n \sim \text{iid}\, N[0, 1]$. We will initialize with $Y_1 \sim N[0, 1.56]$.

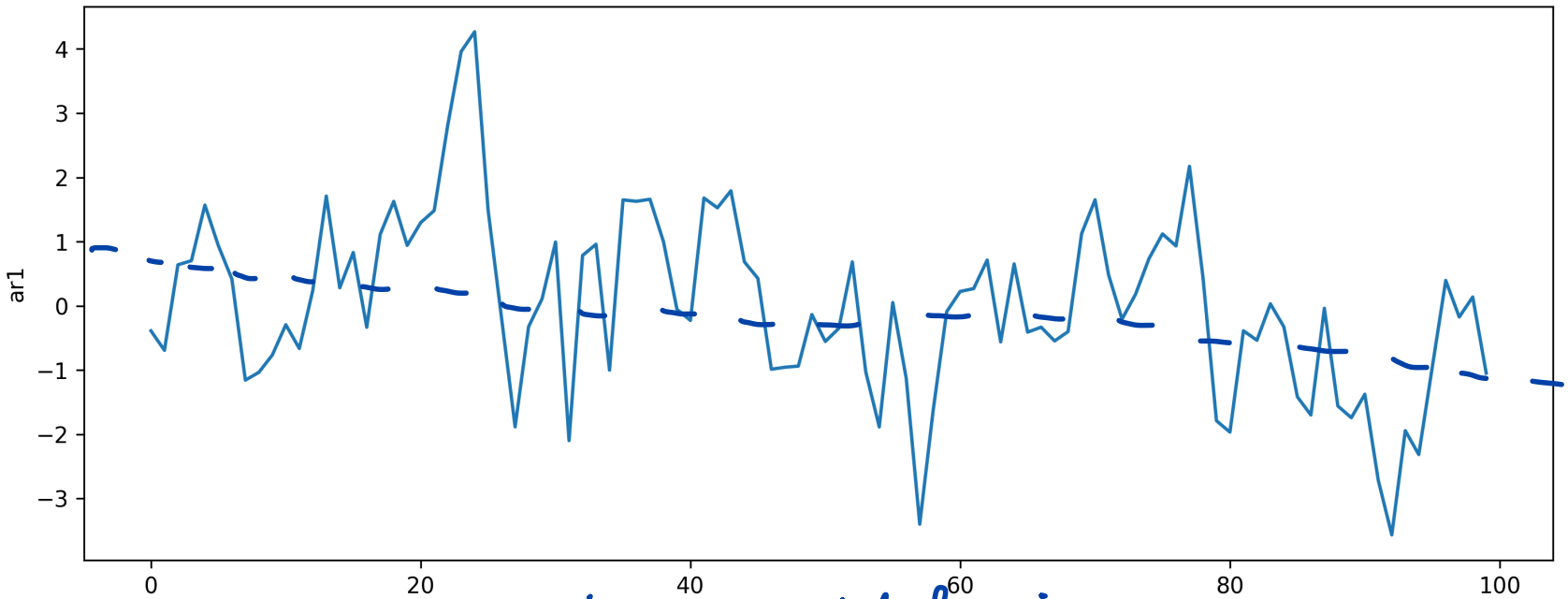
Simulating an autoregressive model

Looking at simulated sample paths is a good way to get intuition about a random process model. First, we do this for the AR(1) model M2 using the ARIMA simulation functionality in statsmodels.

```
np.random.seed(11235)
# Simulate AR(1) with phi=0.6
arparams = np.array([0.6])
ar = np.r_[1, -arparams]
ma = np.array([1])
ar1_process = ArmaProcess(ar, ma)
ar1 = ar1_process.generate_sample(nsample=100,
    scale=1.0)
plt.figure(figsize=(10, 4))
plt.plot(ar1); plt.ylabel('ar1')
plt.tight_layout(); plt.show()
```

note how these specifications relate to the model equations.

AR(1) simulation plot



is well modelled using

does it look like it has a trend?

Interpreting simulations

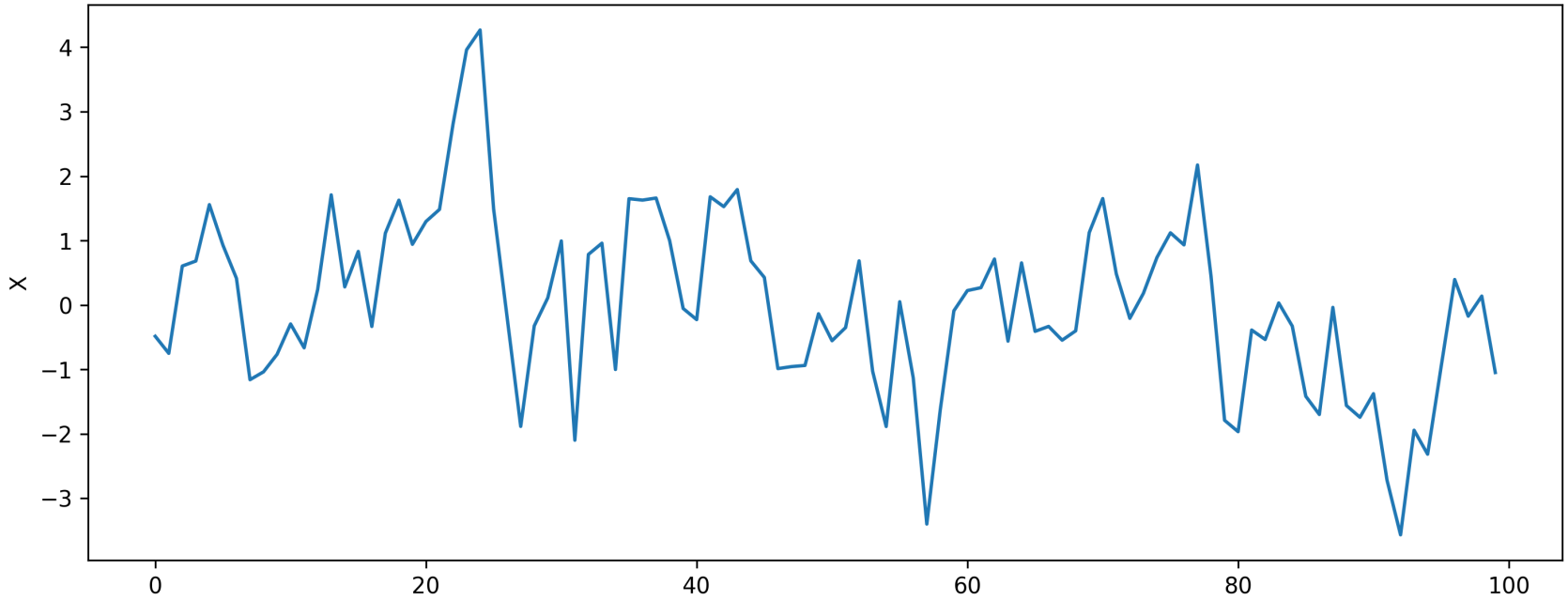
- Does your intuition tell you that these simulated data are evidence for a model with a linear trend?
- The eye looks for patterns in data, and often finds them even when there is no strong statistical evidence. It is easy to see patterns even in white noise. Dependent models produce spurious patterns even more often.
- That is why we need statistical tests!
- Fitting the usual OLS regression model results in a highly statistically significant trend estimate. We need methods that allow for dependence.
- Play with simulating different models with different seeds to train your intuition.

Direct simulation of AR(1)

We can also simulate model M2 directly by writing the model equation:

```
np.random.seed(11235)
N = 100
X = np.zeros(N)
X[0] = np.random.normal(0, np.sqrt(1.56))
for n in range(1, N):
    X[n] = 0.6 * X[n-1] + np.random.normal(0, 1)
plt.figure(figsize=(10, 4))
plt.plot(X)
plt.ylabel('X')
plt.tight_layout()
plt.show()
```

Direct simulation plot



In this case, `generate_sample` exactly matches the direct approach.

Question. What are the advantages and disadvantages of using library simulation functions over the direct simulation method?

direct coding builds understanding

libraries provide high-quality, general, well-tested code.

Sometimes, it is good to do both.

Question. Compute the autocovariance function for model M2.

$Y_n = 0.6\, Y_{n-1} + \varepsilon_n$

$\mathrm{Var}(Y_n) = \mathrm{Var}(0.6\, Y_{n-1} + \varepsilon_n) = 0.6^2 \mathrm{Var}(Y_{n-1}) + \mathrm{Var}(\varepsilon_n) + 1.2\, \mathrm{Cov}(\varepsilon_n, Y_{n-1})$

($\mathrm{Var}(Y_{n-1}) = 1$, $\mathrm{Var}(\varepsilon_n) = 1$, $\mathrm{Cov}(\varepsilon_n, Y_{n-1}) = 0$)

we look for the stationary solution, so $\mathrm{Var}(Y_{n-1}) = \mathrm{Var}(Y_n)$

Y_{n-1} depends on $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_{n-1}$ and noise terms are uncorrelated.

$(1 - 0.36)\mathrm{Var}(Y_n) = 1$, $\quad \mathrm{Var}(Y_n) = \gamma_0 = \frac{1}{0.64} = 1.56.$

The MA(q) moving average model

- The order q moving average model, abbreviated to MA(q), is
  [M3] $$Y_n = \epsilon_n + \theta_1 \epsilon_{n-1} + \cdots + \theta_q \epsilon_{n-q},$$
  where $\{\epsilon_n\}$ is a white noise process.
- To fully specify $Y_{1:N}$ we must specify the joint distribution of $\epsilon_{1-q:N}$.
- Often, we consider the **Gaussian MA(q)** model, where $\{\epsilon_n\}$ is a Gaussian white noise process.
- In M3, we've defined a zero mean process. We could add a mean μ.
- Let's investigate a particular Gaussian MA(2) process, as an exercise.
  [M4] $$Y_n = \epsilon_n + 1.5\epsilon_{n-1} + \epsilon_{n-2},$$
  where $\epsilon_n \sim \text{iid}\, N[0, 1]$.

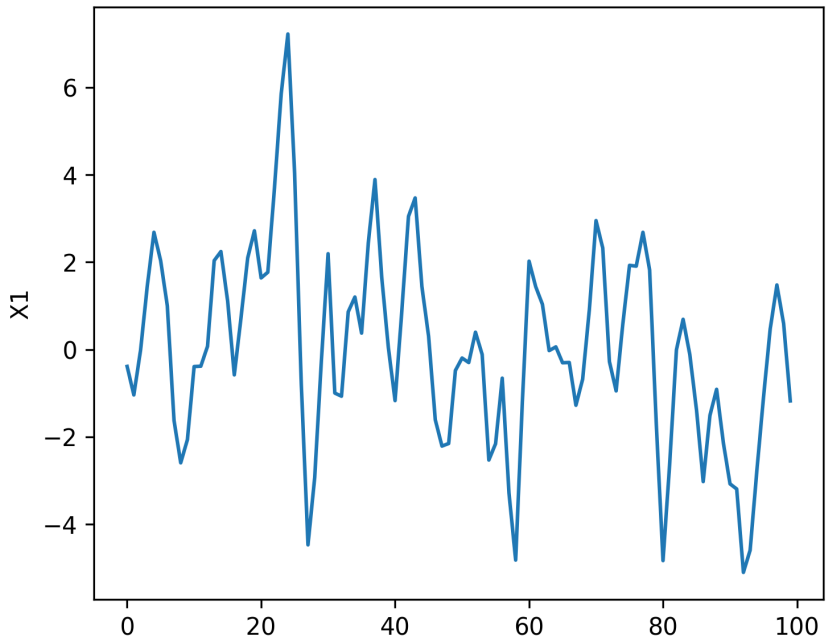
Simulating a moving average model

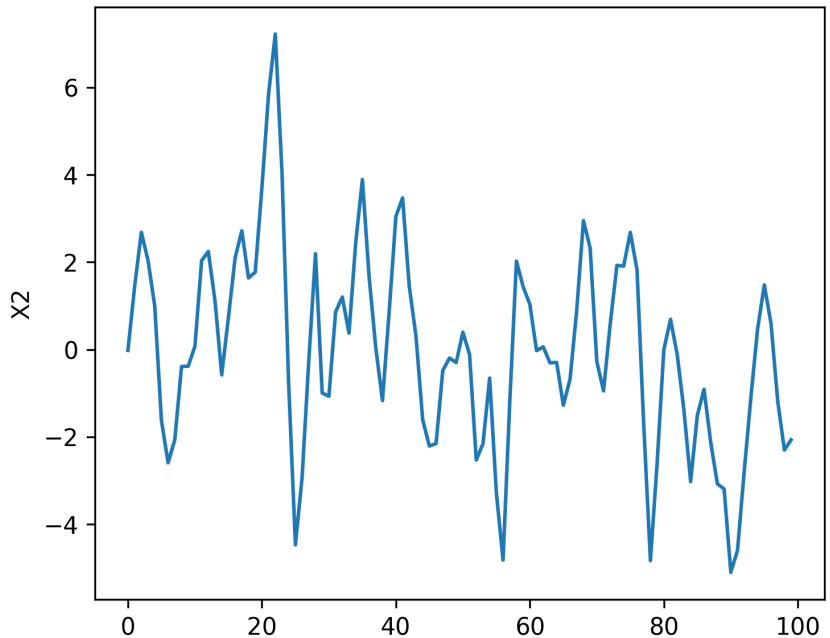
We can simulate M4 using `statsmodels` or directly.

```
N = 100
np.random.seed(11235)
# Using statsmodels
maparams = np.array([1.5, 1])
ar = np.array([1])
ma = np.r_[1, maparams]
ma_process = ArmaProcess(ar, ma)
X1 = ma_process.generate_sample(nsample=N, scale=1.0)

# Direct simulation
np.random.seed(11235)
epsilon = np.random.randn(N+2)
X2 = np.zeros(N)
for n in range(N):
    X2[n] = epsilon[n+2]+1.5*epsilon[n+1]+epsilon[n]
```

MA simulation plots





Comparing simulations

- The two methods look similar, with a lag shift. We can check this is true:

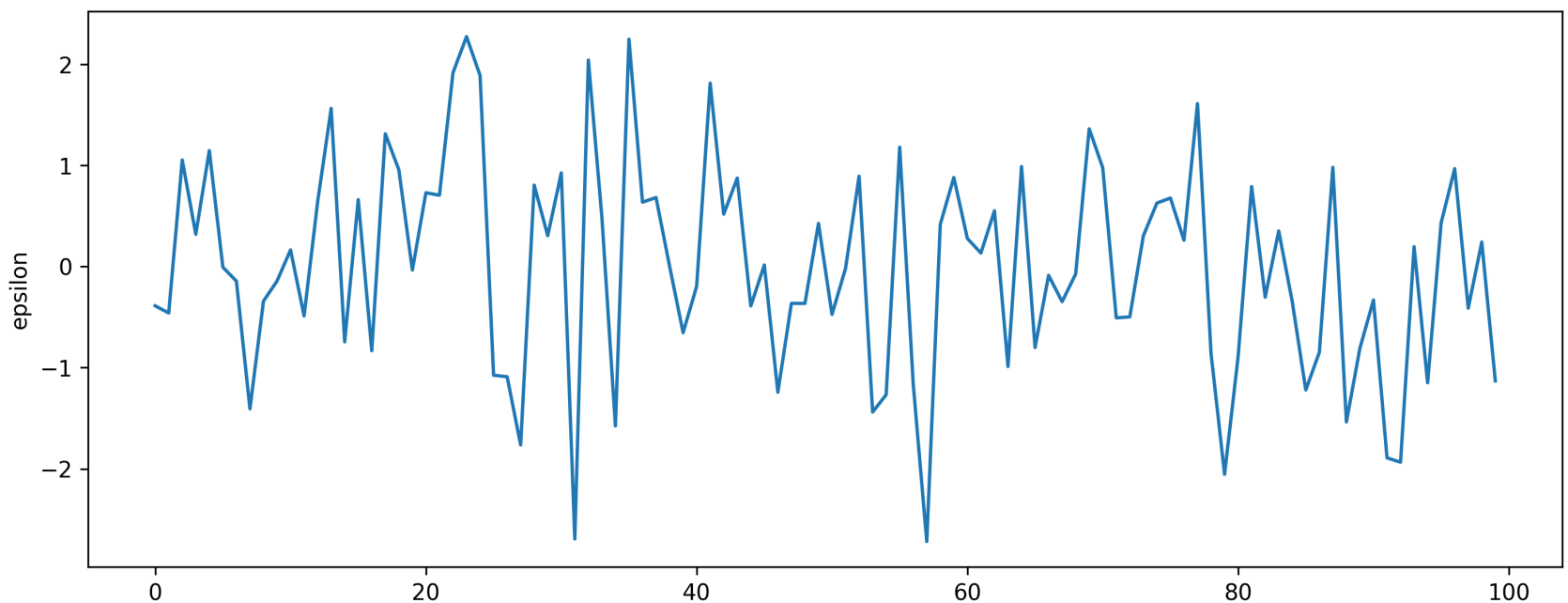
```
test_equal=np.allclose(X1[-(N-2):],X2[:N-2])
print(f"X1 and X2 close: {test_equal}")
```

```
X1 and X2 close: True
```

- Is the spurious evidence for a trend that we saw for the AR(1) model still present for the MA(2) simulation? Let's see if we can also see it in the underlying white noise process:

White noise visualization

```
N = 100
np.random.seed(11235)
epsilon = np.random.randn(N)
plt.figure(figsize=(10, 4))
plt.plot(epsilon); plt.ylabel('epsilon')
plt.tight_layout(); plt.show()
```



Definition:. The **random walk** model is

[M5] $$Y_n = Y_{n-1} + \epsilon_n,$$

where $\{\epsilon_n\}$ is white noise. Unless otherwise specified, we usually initialize with $Y_0 = 0$.

If $\{\epsilon_n\}$ is Gaussian white noise, we have a **Gaussian random walk**.

The random walk model is a special case of AR(1) with $\phi_1 = 1$.

The stochastic difference equation in M5 has an exact solution,

$$Y_n = \sum_{k=1}^{n} \epsilon_k.$$

We can also call $Y_{0:N}$ an **integrated white noise process**. We think of summation as a discrete version of integration.

If data $y_{1:N}$ are modeled as a random walk, the value of Y_0 is usually an unknown. We can treat this as an unknown parameter, or initialize our model at time t_1 with $Y_1 = y_1^*$.

Definition: The **first difference** time series $z_{2:N}$ is defined by

$$z_n = \Delta y_n = y_n - y_{n-1} \tag{2}$$

- From a time series of length N, we only get $N - 1$ first differences.
- A random walk model for $y_{1:N}$ is essentially equivalent to a white noise model for $z_{2:N} = \Delta y_{2:N}$, apart from the issue of initialization.

Definition: The **random walk with drift** model is given by the difference equation

[M6] $$Y_n = Y_{n-1} + \mu + \epsilon_n,$$

driven by a white noise process $\{\epsilon_n\}$. This has solution

$$Y_n = Y_0 + n\mu + \sum_{k=1}^{n} \epsilon_k.$$

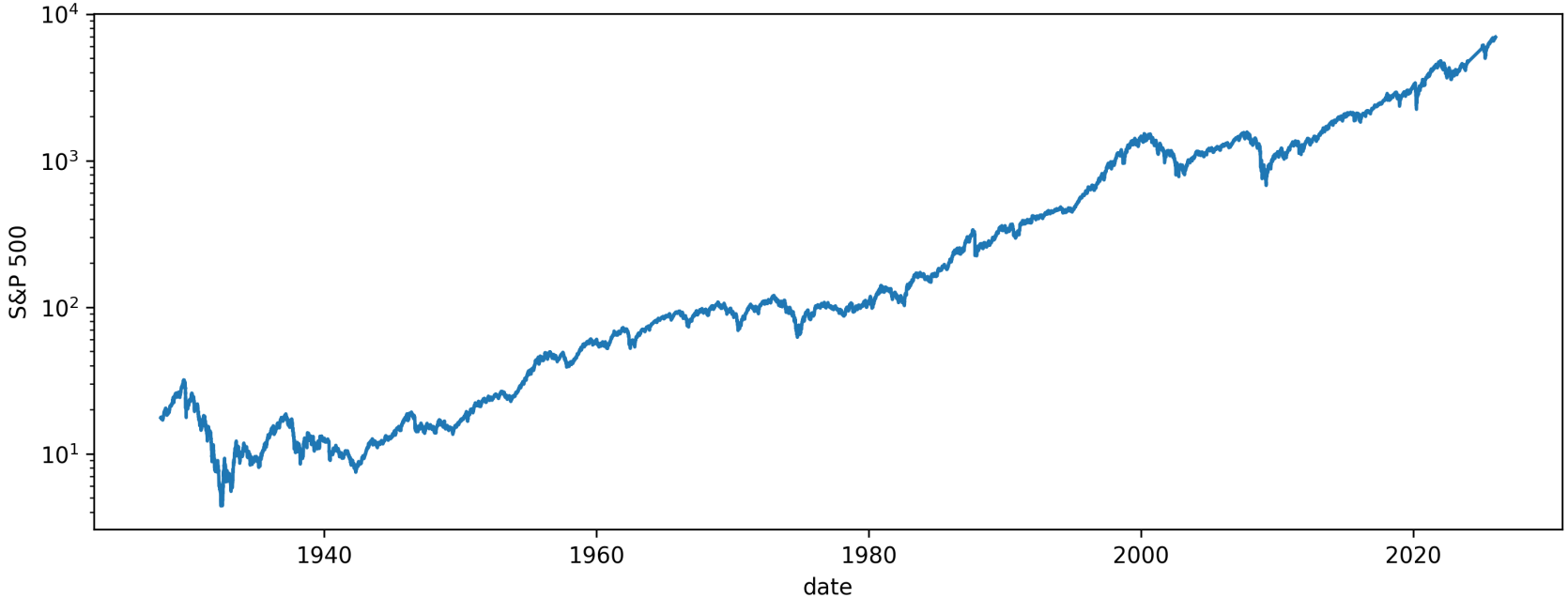
- Here, μ is the mean of the **increments** rather than the random walk process itself.
- As for the random walk without drift, we must define Y_0 to initialize the model and complete the model specification. Unless otherwise specified, we usually initialize with $Y_0 = 0$.

Modeling financial markets as a random walk

The theory of efficient financial markets suggests that the logarithm of a stock market index (or the value of an individual stock, or other investment) might behave like a random walk with drift. We test this on daily S&P 500 data.

```
sp = pd.read_csv("sp500.csv", sep=' ', comment='#',
    index_col=0)
date = pd.to_datetime(sp['Date'])
sp500 = sp['Close'].values
plt.figure(figsize=(10, 4))
plt.semilogy(date, sp500)
plt.xlabel('date')
plt.ylabel('S&P 500')
plt.tight_layout()
plt.show()
```

S&P 500 plot

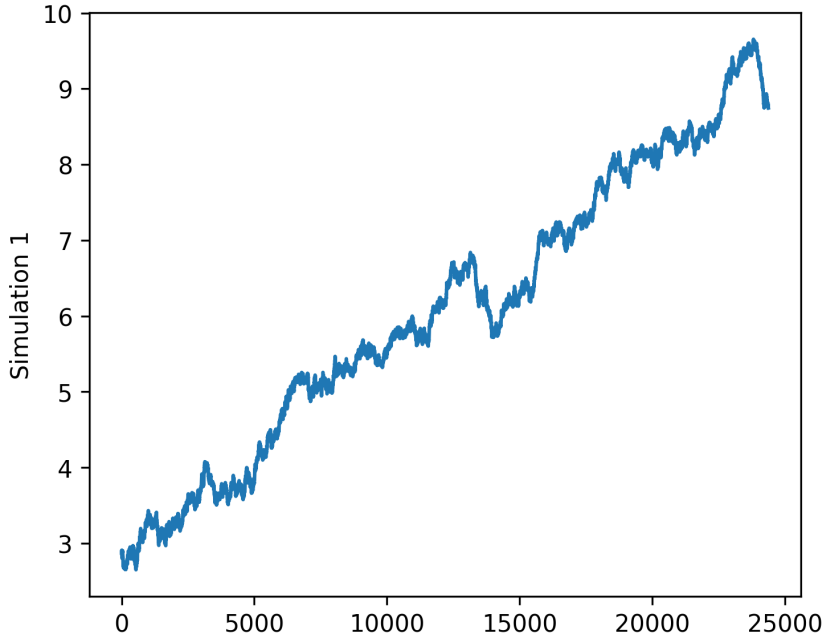


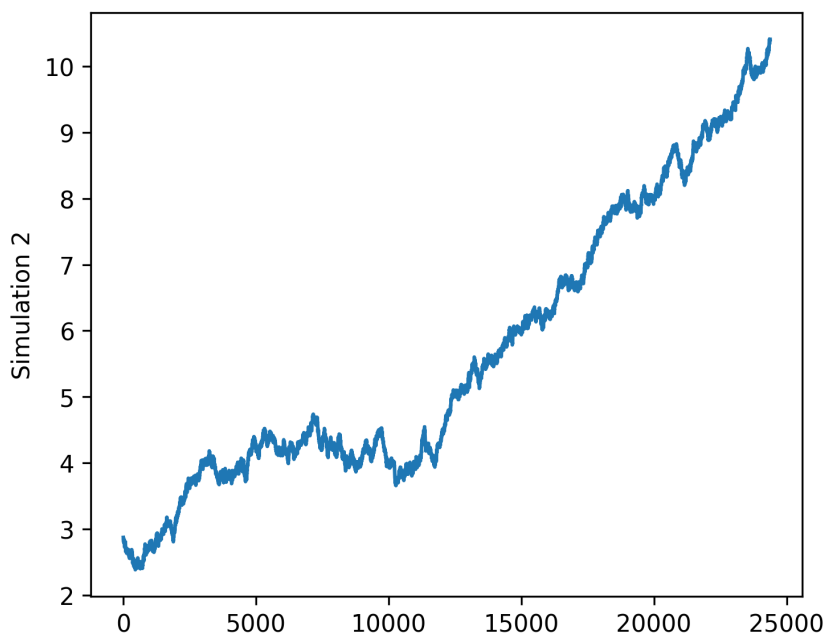
Fitting a random walk model

To train our intuition, we compare the data with simulations from a fitted model. A simple starting point is a Gaussian random walk with drift, having parameters estimated from the data.

```
mu = np.mean(np.diff(np.log(sp500)))
sigma = np.std(np.diff(np.log(sp500)))
N = len(sp500)
X1 = np.log(sp500[0]) + np.cumsum(
    np.r_[0, np.random.normal(mu, sigma, N-1)])
X2 = np.log(sp500[0]) + np.cumsum(
    np.r_[0, np.random.normal(mu, sigma, N-1)])
```

Random walk simulations



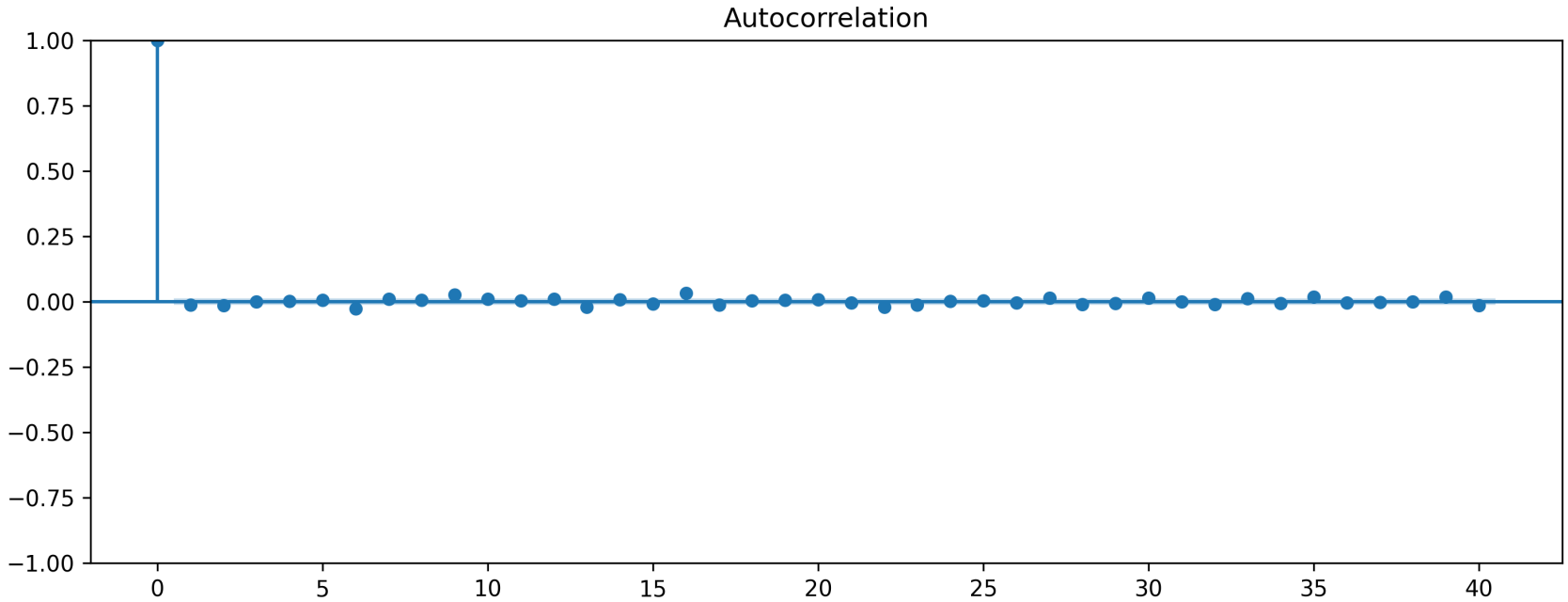


S&P 500 returns ACF

- This seems reasonable so far. Now we plot the sample autocorrelation function (sample ACF) of the log returns.
- It is bad style to refer to quantities using computer code notation. We should set up mathematical notation in the text. Let's try again...
- Let $y_{1:N}$ be the time series of S&P 500 daily closing values downloaded from yahoo.com. Let $z_n = \Delta \log y_n = \log y_n - \log y_{n-1}$.
- We plot the sample autocorrelation function of the time series of S&P 500 returns, $z_{2:N}$.

```
z = np.diff(np.log(sp500))
fig, ax = plt.subplots(figsize=(10, 4))
plot_acf(z, ax=ax, lags=40)
plt.tight_layout()
plt.show()
```

Returns ACF plot



Interpretation

- This looks close to the ACF of white noise. There is some evidence for a small nonzero autocorrelation at some lags.
- Here, we have a long time series. For such a long time series, statistically significant effects may be practically insignificant.

Question. Why may the length of the time series be relevant when considering practical versus statistical significance?

- It seems like the S&P 500 returns (centered, by subtracting the sample mean) may be a real-life time series well modeled by white noise.

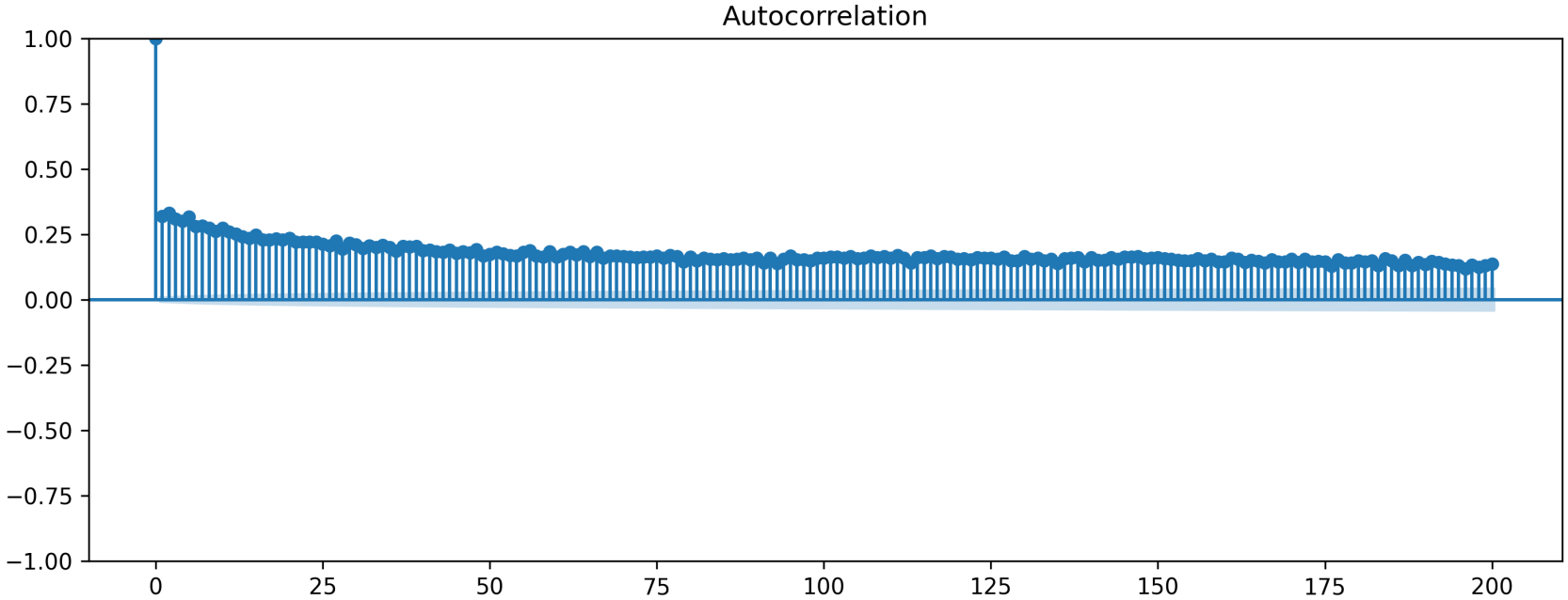
Absolute returns

- Looking at the absolute value of the centered returns is thought-provoking.

Question. How should we interpret the following plot? To what extent does this plot refute the white noise model for the centered returns (or, equivalently, the random walk model for the log index value)?

```
fig, ax = plt.subplots(figsize=(10, 4))
plot_acf(np.abs(z - np.mean(z)), ax=ax, lags=200)
plt.tight_layout()
plt.show()
```

Absolute returns ACF



Volatility and market inefficiencies

- Nowadays, nobody is surprised that the sample ACF of a financial return time series shows little or no evidence for autocorrelation.
- Deviations from the efficient market hypothesis, if you can find them, are of interest.
- Also, it remains a challenge to find good models for **volatility**, the conditional variance process of a financial return model.

Further reading

- Chapter 1 of Shumway and Stoffer [1] provides a complementary introduction to time series analysis.
- If you are relatively new to Python, there are many comprehensive introductions available online.

Acknowledgments

- Compiled on January 14, 2026 using Python.

- We acknowledge previous versions of this course.

References I

[1] Robert H Shumway and David S Stoffer. *Time Series Analysis and its Applications: With R Examples*. 4th. Springer, 2017.