Topic 1A: Time Series as Stochastic Processes

Victor M. Preciado

Contents

1 Time-Series Data

Time-series data refers to a sequence of data points indexed in time order, typically collected at successive, equally spaced points in time. This type of data is prevalent in many fields, including:

- ***Finance:*** Stock prices, exchange rates, and economic indicators.
- ***Signal Processing:*** Audio signals, radar signals, and communication signals.
- ***Engineering:*** Sensor data in manufacturing and energy consumption patterns.
- ***Healthcare:*** Patient vital signs, such as heart rate and blood pressure, monitored over time.
- ***Weather Forecasting:*** Temperature, precipitation, and other meteorological variables.

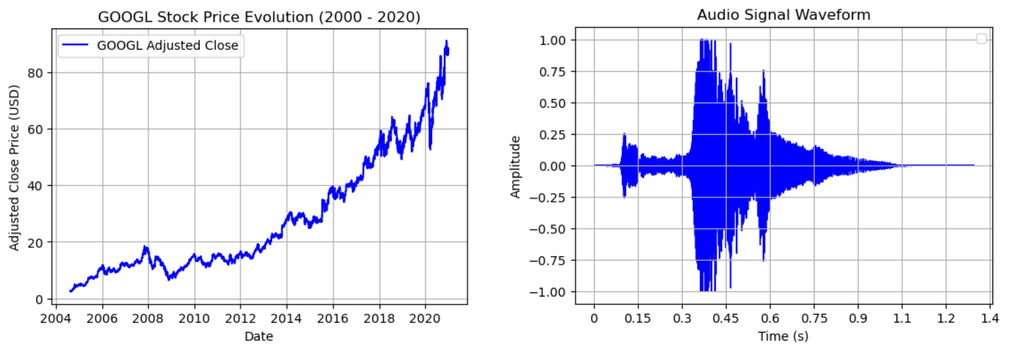

Figure 1: (Left) Evolution of Google stock prices from 2000 to 2020. (Right) Sound waveform.

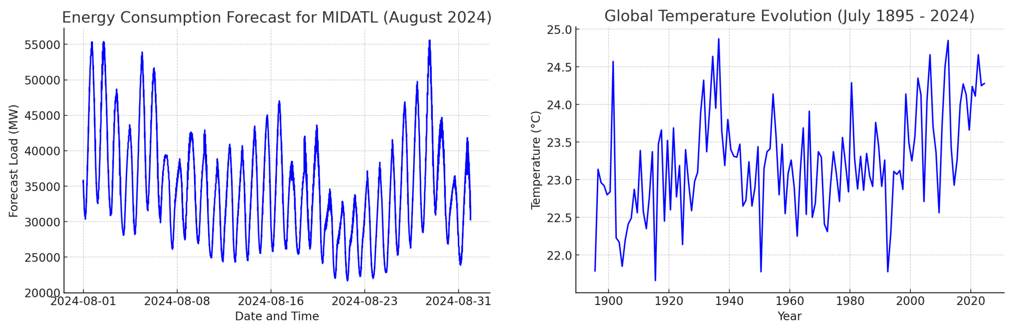

Figure 2: (Left) Energy consumption in the Mid-Atlantic U.S. region during August 2024. (Right) Global temperature evolution from 1895 to 2024

Given the widespread presence of time-series data across these diverse domains, our ability to analyze and interpret such data is of paramount importance. Tools for time series analysis allow practitioners to *uncover patterns, identify trends, and make forecasts*, which are critical for informed decision-making. It is worth noting that certain phenomena are inherently more predictable than others. Our ability to forecast a particular event or value depends on several critical factors, such as:

- **Data Quantity:** The amount of historical data available can significantly impact forecast accuracy. Generally, more data points allow for better model training and more reliable predictions.
- **Data Quality:** The accuracy, consistency, and completeness of the data are crucial. High-quality data leads to more robust models and forecasts.
- **Stationarity:** Time series that exhibit consistent statistical properties over time (stationarity) are often easier to model and forecast.
- **Seasonality and Trends:** The presence of recurring patterns (seasonality) or long-term directions (trends) can aid in forecasting if properly identified and modeled.
- **External Factors:** The influence of external variables not captured in the time series itself can impact predictability.

A central aspect of effective forecasting is determining when accurate predictions are feasible, as opposed to when forecasts offer no advantage over random chance. Reliable forecasts capture authentic patterns and relationships within historical data, without simply replicating past events that are unlikely to recur. Distinguishing between random fluctuations, which should be disregarded, and genuine patterns, which require modeling, is crucial. In this direction, the complexity of time-series data requires specialized techniques to extract meaningful insights.

1.1 The Forecasting Process

The forecasting process typically involves the following steps:

1. **Gathering Information:** Two types of information are essential: *historical data* and *expert knowledge* from those familiar with the system. Expert knowledge helps identify the underlying causal factors, while the data collected must be statistically informative, meaning past observations should provide insights into future outcomes.
2. **Preliminary Analysis:** Graphing the data helps identify trends, seasonality, and other patterns, as well as potential outliers. It also provides insight into relationships between variables and helps guide model selection.

3. **Model Selection and Fitting:** The choice of model will depend on the availability of data, the strength of relationships between variables, the complexity of the system, and the specific objectives of the forecast. In this text, a diverse array of models will be explored, each with its own underlying assumptions and methodological approaches. Understanding these assumptions is crucial, as they guide the model's applicability to different types of data and forecasting scenarios.

4. **Model Evaluation:** Once a model has been selected, forecasts will be generated and their accuracy will be assessed as actual data becomes available. In this text, we will cover several techniques to evaluate forecast accuracy, including out-of-sample testing. Practical challenges such as missing data and limited time series length must be carefully managed during implementation.

The rigorous analysis, interpretation, and forecasting of time-series data require a deep understanding of the underlying stochastic nature of the data. This involves treating time series as random processes, where each observation is seen as a realization of a stochastic process. By exploring time series through this lens, we can better model, predict, and infer patterns that are otherwise obscured by the inherent randomness in the data.

2 Time Series as Stochastic Processes

A time series of length L is defined as a sequence of observations $(y_1, y_2, \ldots, y_L)$ recorded at specific time points $t_1, t_2, \ldots, t_L$. This text primarily focuses on uniformly sampled time series, where $t_k - t_{k-1} = T$ for all $k \in \mathbb{N}$, with T denoting the constant **sampling period**.

From a statistical perspective, the elements of a time series can be interpreted as realizations of a sequence of random variables[1] $(Y_1, Y_2, Y_3, \ldots)$. Specifically, the value y_k observed at time t_k is considered a realization of the random variable Y_k. The collection of these random variables, indexed by time, constitutes a discrete-time **stochastic process**, denoted by $\mathcal{Y} = \{Y_k : k \in \mathbb{N}\}$. The observed sequence $(y_1, y_2, \ldots, y_L)$ is termed a **sample path** of the underlying stochastic process. This probabilistic framework facilitates the application of statistical methods for the analysis and forecasting of time-series data.

2.1 Statistical Properties of Stochastic Processes

The analysis of the statistical properties of a random process is fundamental in time series analysis, as it enables rigorous modeling, prediction, and inference by

[1]All these random variables are defined on the same probability space $(\Omega, \mathcal{F}, P)$.

elucidating the underlying patterns and dependencies in the data. A comprehensive probabilistic description of a stochastic process $\mathcal{Y} = (Y_1, Y_2, \ldots, Y_L)$ of length L is provided by its **joint distribution**. The joint distribution is defined as a multivariate cumulative distribution function (CDF):

$$F_{\mathcal{Y}}(y_1, \ldots, y_L) = \mathbb{P}\{Y_1 \leq y_1, \ldots, Y_L \leq y_L\}.$$

For jointly continuous random variables $(Y_1, \ldots, Y_L)$, $\mathcal{Y}$ possesses a **joint density**, $f_{\mathcal{Y}}(y_1, \ldots, y_L)$. While this CDF/PDF provides a complete description of the stochastic process—encompassing all possible marginal and conditional distributions—it is often computationally intractable and not readily applicable for time series analysis.

The mean, variance, autocovariance, and autocorrelation are fundamental statistical properties of a random process that offer more tractable alternatives to the joint distribution while still capturing essential information. These properties are defined as follows[2]:

- **Mean**: The mean (also referred to as the expected value) of the k-th sample of a random process $\mathcal{Y}$ is defined as:

  $$\mu_{\mathcal{Y}}(k) = \mathbb{E}[Y_k] = \int_{-\infty}^{\infty} y \, f_{Y_k}(y) \, dy,$$

  where $\mathbb{E}[\cdot]$ denotes the expectation operator.

- **Variance**: The variance of the k-th sample of the random process $\mathcal{Y}$ quantifies the dispersion of the random variable around its mean:

  $$\sigma^2_{\mathcal{Y}}(k) = \mathrm{Var}[Y_k] = \mathbb{E}[(Y_k - \mu_{\mathcal{Y}}(k))^2] = \mathbb{E}[Y_k^2] - \mu_{\mathcal{Y}}(k)^2.$$

  A process with constant variance over time is termed **homoskedastic**; otherwise, it is classified as **heteroskedastic**.

- **(Auto)covariance**: The autocovariance between the k-th sample of a random process $\mathcal{Y}$ and the sample lagged by h sampling periods is defined as:

  $$\begin{aligned} C_{\mathcal{Y}}(k, h) &= \mathrm{Cov}[Y_k, Y_{k-h}] \\ &= \mathbb{E}[(Y_k - \mu_{\mathcal{Y}}(k))(Y_{k-h} - \mu_{\mathcal{Y}}(k-h))] \\ &= \mathbb{E}[Y_k Y_{k-h}] - \mu_{\mathcal{Y}}(k)\mu_{\mathcal{Y}}(k-h). \end{aligned}$$

  This measure quantifies the linear dependence between two samples in a stochastic process. It is noteworthy that h can be positive or negative, representing lags or leads respectively. The absence of linear dependence ($C_{\mathcal{Y}}(k, h) = 0$) does not preclude nonlinear dependence between Y_k and Y_{k-h}. For $h = 0$, the autocovariance is equivalent to the variance, $C_{\mathcal{Y}}(k, 0) = \sigma^2_{\mathcal{Y}}(k)$.

[2] It is assumed henceforth that the mean, variance, and autocovariance/autocorrelation exist and are finite.

- **(Auto)correlation**: The autocorrelation function (ACF) is a normalized form of the autocovariance, defined as:

$$R_{\mathcal{Y}}(k,h) = \frac{\mathrm{Cov}[Y_k, Y_{k-h}]}{\sigma_{\mathcal{Y}}(k)\sigma_{\mathcal{Y}}(k-h)} \in [-1,1].$$

  The ACF quantifies the correlation between the current value Y_k of a random process and its past values. As a dimensionless quantity, it offers a more readily interpretable measure of dependence than covariance.

These statistical properties provide critical insights into the behavior of time-series data, facilitating effective modeling, prediction, and analysis of underlying patterns and dependencies.

A random process $\mathcal{Y} = (Y_1, Y_2, \ldots, Y_L)$ is **Gaussian** if the vector of random variables $[Y_1, Y_2, \ldots, Y_L]^\intercal$ follows a multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \Sigma_{\mathcal{Y}})$ with mean vector $\boldsymbol{\mu}_{\mathcal{Y}} = [\mathbb{E}[Y_1], \mathbb{E}[Y_2], \ldots, \mathbb{E}[Y_L]]^\intercal \in \mathbb{R}^L$ and autocovariance matrix $\Sigma_{\mathcal{Y}} \in \mathbb{R}^{L\times L}$. The entries of the covariance matrix $\Sigma_{\mathcal{Y}}$ are given by the autocovariance function, such that the (i,j)-th entry is $[\Sigma_{\mathcal{Y}}]_{i,j} = \mathrm{Cov}[Y_i, Y_j]$. A key property of Gaussian processes is that they are fully characterized by their mean vector $\boldsymbol{\mu}_{\mathcal{Y}}(k)$ and autocovariance matrix $\Sigma_{\mathcal{Y}}$. This implies that these two parameters suffice to describe the statistical properties of the process. Furthermore, any linear combination of the components of a Gaussian process is also Gaussian. This property simplifies the analysis and modeling of such processes, particularly in time series and signal processing applications.

Example 1: Autoregressive Process of Order 1

Consider a random process $\mathcal{Y} = \{Y_k : k \in \mathbb{N}\}$ defined by the following recursion:

$Y_k = \phi Y_{k-1} + \epsilon_k$ for all $k \in \mathbb{N}$, with deterministic initial condition $Y_0 = y_0 \in \mathbb{R}$,

where $|\phi| < 1$ is a constant parameter, and ϵ_k is a sequence of independent and identically distributed (i.i.d.) Gaussian random variables with zero mean and variance σ_ϵ^2. This sequence of random variables is commonly denoted as:

$$\epsilon_k \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma_\epsilon^2).$$

This process is an example of an **autoregressive model of order 1**, denoted as AR(1). Its statistical properties are as follows:

- **Mean**: The mean of the process $\mathcal{Y}$, taking into account the initial condition $Y_0 = y_0$, is:

$$\mu_{\mathcal{Y}}(k) = \mathbb{E}[Y_k] = \phi\mathbb{E}[Y_{k-1}] + \mathbb{E}[\epsilon_k] = \phi\mathbb{E}[Y_{k-1}].$$

Since $\mathbb{E}[\epsilon_k] = 0$ and $Y_0 = y_0$, this recursive relationship leads to:

$$\mu_{\mathcal{Y}}(k) = \phi^k y_0 \quad \text{for all } k.$$

As $k \to \infty$, the mean decays to 0 exponentially, provided that $|\phi| < 1$.

- **Variance**: The variance of Y_k is computed using the recursive relationship:

  $$\sigma^2_{\mathcal{Y}}(k) = \text{Var}[Y_k] = \text{Var}[\phi Y_{k-1} + \epsilon_k] = \phi^2 \text{Var}[Y_{k-1}] + \sigma^2_\epsilon,$$

  given the independence of Y_{k-1} and ϵ_k. The solution to this recursion, with the initial condition $Y_0 = y_0$ and $|\phi| < 1$, is:

  $$\sigma^2_{\mathcal{Y}}(k) = \sigma^2_\epsilon \sum_{i=0}^{k-1} \phi^{2i} = \sigma^2_\epsilon \frac{1 - \phi^{2k}}{1 - \phi^2}.$$

  As $k \to \infty$, the variance converges to:

  $$\sigma^2_{\mathcal{Y}}(\infty) = \frac{\sigma^2_\epsilon}{1 - \phi^2}.$$

- **95% Confidence Interval**: Since Y_k is a linear combination of independent Gaussian random variables, it follows a Gaussian distribution for all k. Therefore, a 95% confidence interval (CI) for Y_k is given by $[\mu_{\mathcal{Y}}(k) \pm 1.96 \cdot \sigma_{\mathcal{Y}}(k)]$. Given that $\mu_{\mathcal{Y}}(k) = \phi^k y_0$, the 95% CI for Y_k is:

  $$Y_k \in \left[\phi^k y_0 - 1.96 \cdot \sigma_\epsilon \sqrt{\frac{1 - \phi^{2k}}{1 - \phi^2}},\ \phi^k y_0 + 1.96 \cdot \sigma_\epsilon \sqrt{\frac{1 - \phi^{2k}}{1 - \phi^2}} \right].$$

  As $k \to \infty$, the confidence interval stabilizes to:

  $$Y_\infty \in \left[-1.96 \cdot \frac{\sigma_\epsilon}{\sqrt{1 - \phi^2}},\ 1.96 \cdot \frac{\sigma_\epsilon}{\sqrt{1 - \phi^2}} \right],$$

  indicating the long-term variability of the process around its mean of zero.

  Figure 3-(left) shows 10 random realizations of an AR(1) process with initial condition $Y_0 = 0$, along with a shaded region representing the 95% confidence interval for the process.

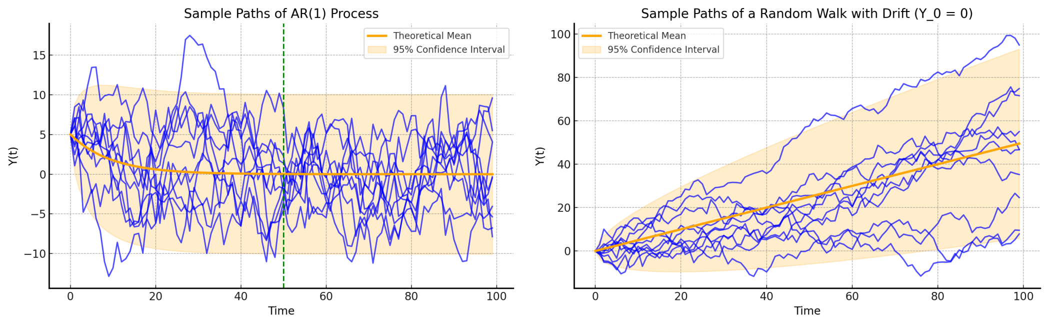

Figure 3: Comparison of 10 sample paths of an AR(1) process (left) and a random walk with drift (right). The theoretical means are plotted in orange. The shaded regions represent the 95% confidence intervals based on the theoretical variance.

Example 2: Random Walk with Drift

Consider a random process $\mathcal{Y} = \{Y_k : k \in \mathbb{N}\}$ defined by the following recursion:

$$Y_k = \delta + Y_{k-1} + \epsilon_k \text{ for all } k \in \mathbb{N}, \text{ with initial condition } Y_0 = 0,$$

where δ is a constant called the **drift**, which induces a linear growth in the mean of the process, and ϵ_k is a sequence of i.i.d. Gaussian random variables with zero mean and variance σ_ϵ^2. This process is an example of a **random walk with drift** and its statistical properties are as follows:

- **Mean**: The mean of the process $\mathcal{Y}$ is given by:

$$\mu_{\mathcal{Y}}(k) = \mathbb{E}[Y_k] = \mathbb{E}[\delta + Y_{k-1} + \epsilon_k] = \delta + \mathbb{E}[Y_{k-1}] + \mathbb{E}[\epsilon_k].$$

  Since $\mathbb{E}[\epsilon_k] = 0$, we have $\mu_{\mathcal{Y}}(k) = \delta + \mu_{\mathcal{Y}}(k-1)$. Since $Y_0 = 0$, this recurrence relation gives:

$$\mu_{\mathcal{Y}}(k) = k\delta.$$

  This indicates that the mean of the process increases linearly over time, reflecting the effect of the drift δ.

- **Variance**: The variance of the k-th sample is:

$$\sigma_{\mathcal{Y}}^2(k) = \mathrm{Var}[Y_k] = \mathrm{Var}[\delta + Y_{k-1} + \epsilon_k].$$

  Given that Y_{k-1} and ϵ_k are independent (and therefore uncorrelated), we have:

$$\sigma_{\mathcal{Y}}^2(k) = \mathrm{Var}[Y_{k-1}] + \mathrm{Var}[\epsilon_k] = \sigma_{\mathcal{Y}}^2(k-1) + \sigma_\epsilon^2.$$

Since $Y_0 = 0$, the variance evolves as:

$$\sigma^2_{\mathcal{Y}}(k) = k\sigma^2_\epsilon.$$

Thus, the variance of the process increases linearly with time (and the standard deviation as the square root of time), indicating that the process becomes more variable as time progresses. This is in contrast to the AR(1) process, where the variance converges to a constant value as k increases.

- **95% Confidence Interval**: Because ϵ_k are Gaussian random variables, Y_k, as a sum of Gaussian random variables, is also Gaussian at each time k. Using this distribution, a 95% confidence interval for Y_k can be computed as:

$$Y_k \in [k\delta - 1.96\,\sigma_\epsilon\sqrt{k},\ k\delta + 1.96\,\sigma_\epsilon\sqrt{k}].$$

This interval gives a range in which we expect Y_k to fall with 95% probability at each time step k, accounting for both the drift δ and the increasing variance over time. Note that the width of this interval grows with $\sqrt{k}$, reflecting the increasing uncertainty in the process as time progresses.

In Fig. 3-(right), we display 10 random realizations of random walk with zero drift, along with a shaded region representing the 95% confidence interval for the process.

Example 3: Moving Average Process of Order 1

Consider a random process $\mathcal{Y} = \{Y_k : k \in \mathbb{N}\}$ defined by the following recursion:

$$Y_k = \epsilon_k + \theta\epsilon_{k-1}, \quad \text{with } \epsilon_k \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2_\epsilon),$$

where θ is a constant parameter. This process is an example of a **moving average model of order 1**, denoted as MA(1). Its statistical properties are as follows:

- **Mean**: The mean of the process $\mathcal{Y}$ is:

$$\mu_{\mathcal{Y}}(k) = \mathbb{E}[Y_k] = \mathbb{E}[\epsilon_k] + \theta\mathbb{E}[\epsilon_{k-1}] = 0, \quad \text{for all } k \in \mathbb{N}.$$

- **Variance**: The variance of Y_k is given by:

$$\sigma^2_{\mathcal{Y}}(k) = \text{Var}[Y_k] = \text{Var}[\epsilon_k + \theta\epsilon_{k-1}] = \sigma^2_\epsilon + \theta^2\sigma^2_\epsilon = \sigma^2_\epsilon(1+\theta^2), \quad \text{for all } k \in \mathbb{N},$$

due to the independence of ϵ_k and ϵ_{k-1}.

- **95% Confidence Interval**: Since Y_k is a linear combination of Gaussian random variables, it follows a Gaussian distribution. Therefore, a 95% confidence interval (CI) for Y_k is given by $[\mu_{\mathcal{Y}}(k) \pm 1.96 \cdot \sigma_{\mathcal{Y}}(k)]$. Given that $\mu_{\mathcal{Y}}(k) = 0$, the 95% CI for Y_k is:

$$Y_k \in \left[-1.96 \cdot \sigma_\epsilon \sqrt{1+\theta^2},\ 1.96 \cdot \sigma_\epsilon \sqrt{1+\theta^2}\right].$$

This confidence interval reflects the variability of the MA(1) process around its mean of zero, showing how the past shock ϵ_{k-1} influences the overall spread of the data.

2.2 Stationarity in Stochastic Processes

Stationarity is a fundamental concept in time series analysis that refers to the idea that the statistical properties of a time series do not change over time. When a process is stationary, it is easier to model and make predictions because its behavior is consistent over time. This consistency allows for the development of more reliable forecasting models, as past patterns are assumed to be representative of future behavior.

2.2.1 Strong-Sense Stationarity (SSS)

One of the strongest forms of stationarity is known as **strong-sense stationarity (SSS)**. A random process $\mathcal{Y} = \{Y_k : k \in \mathbb{N}\}$ is SSS if the joint distribution of any finite collection of random variables from the process is invariant under shifts in time. Specifically, for any collection of discrete time indices $k_1, k_2, \ldots, k_n$, the joint distribution of the corresponding random variables $(Y_{k_1}, Y_{k_2}, \ldots, Y_{k_n})$ remains unchanged if we shift all time indices by a constant h. Formally, the process is SSS if for all $n \in \mathbb{N}$ and any set of time indices $k_1, k_2, \ldots, k_n$:

$$\mathbb{P}(Y_{k_1} \leq y_{k_1}, \ldots, Y_{k_n} \leq y_{k_n}) = \mathbb{P}(Y_{k_1+h} \leq y_{k_1}, \ldots, Y_{k_n+h} \leq y_{k_n}),$$

for all $h \in \mathbb{Z}$ and any values $y_{k_1}, y_{k_2}, \ldots, y_{k_n} \in \mathbb{R}$. In other words, the entire joint probability structure of the process is **shift-invariant**, i.e., does not depend on the value of the lag. The condition of strong-sense stationarity is very stringent and has several important implications for the random process $\mathcal{Y}$. In particular, when a process is SSS, it satisfies the following conditions:

- **Time-Invariance of Means and Variances:** Since the entire joint distribution is invariant under time shifts, this implies that the mean and variance of the process must be constant over time, i.e.,

$$\mu_{\mathcal{Y}}(k) = \mu_{\mathcal{Y}} \text{ and } \sigma_{\mathcal{Y}}(k) = \sigma_{\mathcal{Y}} \text{ for all } k \in \mathbb{N}.$$

This indicates that the expected value of the process does not change as time progresses.

- **Shift-Invariance of the Covariances and Autocorrelation:** Strong-sense stationarity implies that the pairwise statistical dependencies within the process remain constant over time. As a result, the autocovariance (and autocorrelation) between two random variables Y_k and Y_{k+h} depends only on the time difference h, not on the specific times k or $k+h$. In a stationary process, we can express the autocovariance and autocorrelation functions using a single argument (with a slight abuse of notation), i.e.,

  $$C_{\mathcal{Y}}(k,h) = C_{\mathcal{Y}}(h) \text{ for all } k \in \mathbb{N},\ h \in \mathbb{Z}.$$

  Similarly, the autocorrelation function is also independent of k and thus shift-invariant:

  $$R_{\mathcal{Y}}(k,h) = R_{\mathcal{Y}}(h) \text{ for all } k \in \mathbb{N},\ h \in \mathbb{Z}.$$

  This shift-invariance property of both the autocovariance and autocorrelation functions is a key characteristic of stationary processes, reflecting their consistent statistical behavior over time.

Example 4: White Noise Process

A simple example of a strongly stationary process is the **white noise process**. In this case, the process is the same as the noise term:

$$Y_k = \epsilon_k, \quad \text{with } \epsilon_k \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma_\epsilon^2).$$

Some relevant statistics of a WN are the following:

- **Mean:** The mean of the white noise process is constant and given by:

  $$\mu_{\mathcal{Y}}(k) = \mathbb{E}[\epsilon_k] = 0 \text{ for all } k.$$

  This reflects the fact that a white noise fluctuates around zero (see Fig. 4 for a typical sample path).

- **Autocovariance:** Due to the independence of the random variables in the white noise process, we have that:

  $$C_{\mathcal{Y}}(k,h) = \mathbb{E}[\epsilon_k \epsilon_{k-h}] = \sigma_\epsilon^2 \mathbb{1}_{\{k=k-h\}} = \sigma_\epsilon^2 \mathbb{1}_{\{h=0\}} = \begin{cases} \sigma_\epsilon^2 & \text{if } h = 0, \\ 0 & \text{if } h \neq 0, \end{cases}$$

where we have used the indicator function $\mathbb{1}_{\{\cdot\}}$, defined as $\mathbb{1}_{\{P\}} = 1$ if the Boolean expression P is true; 0 otherwise. This means that the random variables are uncorrelated unless they coincide in time. It is important to note that while independence implies uncorrelatedness, the converse is not generally true; uncorrelated random variables are not necessarily independent. One notable exception is when the random variables are Gaussian, in which case uncorrelatedness does imply independence.

- **Autocorrelation:** The autocorrelation function is computed by normalizing the autocovariance as:

$$R_{\mathcal{Y}}(k,h) = \frac{C_{\mathcal{Y}}(k,h)}{\sigma_\epsilon^2} = \begin{cases} 1 & \text{if } h = 0, \\ 0 & \text{if } h \neq 0, \end{cases}$$

  This indicates that each random variable in the white noise process is only correlated with itself and uncorrelated with all other variables, regardless of the time lag. In Fig. 4-(right), we plot the *empirical* autocorrelation function of a sample path of the white noise.

Note that the sequence $(Y_1, Y_2, \ldots)$ is a collection of joint Gaussians. Hence, we have that for all $n \in \mathbb{N}$ and any set of time indices $k_1, k_2, \ldots, k_n$:

$$(Y_{k_1}, Y_{k_2}, \ldots, Y_{k_n}) \sim \mathcal{N}(\mathbf{0}_n, \mathbb{I}_n)$$

i.e., a multivariate Gaussian with mean vector $\mathbf{0}_n$ (the n-dimensional vector of all zeros) and covariance matrix $\mathbb{I}_n$ (the $n \times n$ identity matrix). This distribution is time-invariant, meaning it depends only on the number of variables n, not on the specific time indices chosen. To prove that a Gaussian white process is strongly stationary stochastic (SSS), we need to show that the joint distribution of any subset of the process is invariant under time shifts. For any time shift $h \in \mathbb{Z}$:

$$(Y_{k_1+h}, Y_{k_2+h}, \ldots, Y_{k_n+h}) \sim \mathcal{N}(\mathbf{0}_n, \mathbb{I}_n)$$

for any choice of n, $k_1, \ldots, k_n$, and h, thus satisfying the definition of strong stationarity.

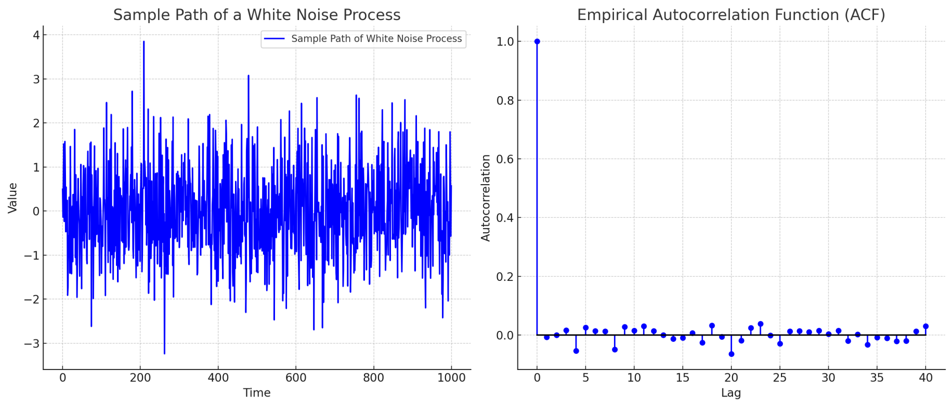

Figure 4: (Left) A typical sample path of a white noise and (right) its empirical ACF.

Strong-sense stationarity (SSS) is a fundamental concept in time series analysis and stochastic processes. Its shift-invariance property facilitates consistent estimation and prediction methods across different time periods, particularly valuable in fields such as finance and economics. However, the SSS assumption often proves too restrictive for many real-world time series, which frequently exhibit non-stationary characteristics such as heteroskedasticity or time-varying trends.

2.2.2 Weak-Sense Stationarity (WSS)

While strong-sense stationarity requires that the entire joint distribution of the process be invariant under time shifts, **Weak-Sense Stationarity (WSS)**, also called *wide-sense stationarity*, is a less restrictive condition. WSS requires only that the first two moments (mean and autocovariance) of the process be time-invariant. These conditions are often sufficient for many practical applications and are more readily satisfied by real-world processes.

Formally, a random process $\mathcal{Y} = \{Y_k : k \in \mathbb{N}\}$ is said to be Weak-Sense Stationary if it satisfies the following conditions:

1. **Time-Invariant Mean and Variance**: The mean and variance of the process must be constant for all $k \in \mathbb{N}$, i.e., $\mu_{\mathcal{Y}}(k) = \mu_{\mathcal{Y}}$ and $\sigma^2_{\mathcal{Y}}(k) = \sigma^2_{\mathcal{Y}}$ for all $k \in \mathbb{N}$. This means the expected value and the volatility of the process do not change over time.

2. **Shift-Invariant Autocovariance/Autocorrelation**: The autocovariance $C_{\mathcal{Y}}(k, h)$ depends only on the time difference (or lag) h, and not on the specific times k or $k + h$. Thus, we have: $C_{\mathcal{Y}}(k, h) = C_{\mathcal{Y}}(h)$ for all $k \in \mathbb{N}$, $h \in \mathbb{Z}$. This implies that the covariance between two points in the process is determined

only by the lag h, not by their absolute positions in the time series[3].

The shift-invariant autocovariance property of WSS processes carries significant practical implications, particularly in ensuring consistent estimation and enabling reliable modeling and analysis. For example, the **portion of the variance** of Y_k explained by its past values is a crucial concept in time series forecasting because it quantifies the **predictability** of the process based on its previous observations. For WSS processes, the portion of the variance explained by the past values at lag indices $\mathcal{H} \subset \mathbb{N}$ is computed by summing the squares of the autocorrelation coefficients at those lags, as follows:

$$P_{\mathcal{H}}(\mathcal{Y}) = \sum_{h \in \mathcal{H}} R_{\mathcal{Y}}(h)^2.$$

For example, if the autocorrelation values at lags 1, 2, and 3 are $R_{\mathcal{Y}}(1) = 0.6$, $R_{\mathcal{Y}}(2) = 0.4$, and $R_{\mathcal{Y}}(3) = 0.2$, then summing the squared autocorrelations gives $0.6^2 + 0.4^2 + 0.2^2 = 0.56$. This means that 56% of the variance of Y_k is explained by the lagged observations Y_{k-1}, Y_{k-2}, and Y_{k-3}.

In the case of a white noise process, none of the variance of Y_k is explained by its past values, since $R_{\mathcal{Y}}(h) = 0$ for all $h \neq 0$.

Example 5: (Asymptotic) Autocovariance of AR(1)

Consider the AR(1) stochastic process, defined by the recursion:

$$Y_k = \phi Y_{k-1} + \epsilon_k, \quad \text{with } \epsilon_k \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma_\epsilon^2), Y_0 = y_0 \text{ and } |\phi| < 1$$

As observed in Fig. 5-(left), the AR(1) process is not stationary for small k, since the deterministic initial condition induces an initial growth of the variance. However, as established in Example 1, the mean is zero and the variance converges exponentially fast to the constant $\frac{\sigma_\epsilon^2}{1-\phi^2}$ as $k \to \infty$.
To derive the asymptotic autocovariance function, we express Y_k in terms of Y_{k-h} using the recursion:

$$Y_k = \phi^h Y_{k-h} + \sum_{i=1}^{h} \phi^{h-i} \epsilon_{k-h+i}. \tag{1}$$

This expression can be verified by nested substitution of the AR(1) recursion.

[3]Whenever the autocovariance/autocorrelation is described as a function of a single argument, i.e., the lag h, it implicitly suggests that the stochastic process is WSS.

Multiplying this expression by Y_{k-h} and taking expectations, we obtain:

$$\begin{aligned}\mathbb{E}[Y_k Y_{k-h}] &= \mathbb{E}\left[\left(\phi^h Y_{k-h} + \sum_{i=1}^{h} \phi^{h-i}\epsilon_{k-h+i}\right) Y_{k-h}\right] \\ &= \phi^h \mathbb{E}[Y_{k-h}^2] + \sum_{i=1}^{h} \phi^{h-i}\mathbb{E}[\epsilon_{k-h+i} Y_{k-h}].\end{aligned}$$

Since Y_{k-h} and ϵ_{k-h+i} are uncorrelated for all i, the cross terms vanish, leaving:

$$\mathbb{E}[Y_k Y_{k-h}] = \phi^h \mathbb{E}[Y_{k-h}^2].$$

As $k \to \infty$, $\mathbb{E}[Y_{k-h}^2]$ approaches the asymptotic variance $\frac{\sigma_\epsilon^2}{1-\phi^2}$, yielding:

$$\lim_{k\to\infty} \operatorname{Cov}[Y_k, Y_{k-h}] = \phi^h \frac{\sigma_\epsilon^2}{1-\phi^2} = C_{\mathcal{Y}}(h).$$

The asymptotic autocorrelation function is thus:

$$\lim_{k\to\infty} R_{\mathcal{Y}}(k,h) = \phi^h.$$

For $|\phi| < 1$, this autocorrelation decays exponentially with increasing lag h, reflecting that the AR(1) process becomes progressively uncorrelated for distant observations. Fig. 5 presents a sample path of length $L = 1,000$, with the right subplot showing both the empirical and theoretical autocorrelation functions.

In conclusion, the AR(1) process asymptotically approaches WSS as the influence of the initial condition diminishes exponentially over time. This property makes AR(1) processes useful in modeling phenomena with short-term memory effects, such as certain financial time series or physical systems with gradual decay of autocorrelations.

Example 6: Autocovariance of a random walk

Consider a random walk process defined by:

$$Y_k = Y_{k-1} + \epsilon_k, \quad \text{with } \epsilon_k \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma_\epsilon^2) \text{ and } Y_0 = 0.$$

For this random walk, we already showed in Example 2 that the mean is zero for all k (since the drift $\delta = 0$). Therefore, the covariance becomes $C_{\mathcal{Y}}(k,h) = \mathbb{E}[Y_k Y_{k-h}]$.

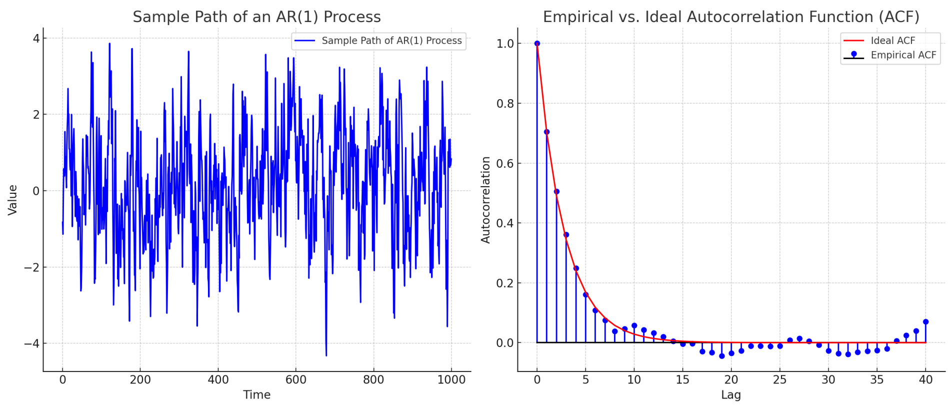

Figure 5: (Left) A sample path of the AR(1) process. (Right) Empirical and theoretical ACF.

We can express Y_k and Y_{k-h} as a sum of the increments ϵ_k:

$$Y_k = Y_0 + \sum_{i=1}^{k} \epsilon_i \text{ and } Y_{k-h} = Y_0 + \sum_{i=1}^{k-h} \epsilon_i.$$

Now, we compute the covariance:

$$C_{\mathcal{Y}}(k,h) = \mathbb{E}\left[\left(\sum_{i=1}^{k} \epsilon_i\right)\left(\sum_{j=1}^{k-h} \epsilon_j\right)\right] = \sum_{i=1}^{k}\sum_{j=1}^{k-h} \mathbb{E}[\epsilon_i \epsilon_j].$$

Since ϵ_i are i.i.d., $\mathbb{E}[\epsilon_i \epsilon_j] = 0$ for $i \neq j$, and for $i = j$, we have $\mathbb{E}[\epsilon_i^2] = \sigma_\epsilon^2$. Therefore, the only terms that contribute to the sum are those with $i = j$, yielding:

$$C_{\mathcal{Y}}(k,h) = \sum_{i=1}^{k-h} \sigma_\epsilon^2 = (k-h)\sigma_\epsilon^2 \quad \text{for } h \leq k.$$

This shows that the covariance depends linearly on k, the current time, and decreases with increasing lag h. Also, since the covariance depends on both k and the lag h, a random walk is *not* stationary (in any sense). The non-stationarity of the random walk process has significant practical implications. Unlike stationary processes, random walks do not have a constant mean or variance, making traditional time series analysis techniques less applicable.

Example 7: (Asymptotic) Autocovariance of MA(1)

Consider the MA(1) stochastic process, defined by the equation:

$$Y_k = \epsilon_k + \theta\epsilon_{k-1}, \quad \text{with } \epsilon_k \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma_\epsilon^2).$$

As observed in the MA(1) process, the structure is stationary for all time indices since there are no recursive terms involving past values of Y_k. The mean of the process is zero, and the variance is constant for all k, i.e., $\text{Var}[Y_k] = \sigma_\epsilon^2(1+\theta^2)$ (see Example 3).
To derive the autocovariance function, we express Y_k and Y_{k-h} in terms of the noise terms:

$$Y_k = \epsilon_k + \theta\epsilon_{k-1} \quad \text{and} \quad Y_{k-h} = \epsilon_{k-h} + \theta\epsilon_{k-h-1}.$$

For a generic lag h, the autocovariance between Y_k and Y_{k-h} is calculated as:

$$\begin{aligned}\text{Cov}[Y_k, Y_{k-h}] &= \mathbb{E}[(\epsilon_k + \theta\epsilon_{k-1})(\epsilon_{k-h} + \theta\epsilon_{k-h-1})] \\ &= \mathbb{E}[\epsilon_k\epsilon_{k-h}] + \theta\mathbb{E}[\epsilon_k\epsilon_{k-h-1}] + \theta\mathbb{E}[\epsilon_{k-1}\epsilon_{k-h}] + \theta^2\mathbb{E}[\epsilon_{k-1}\epsilon_{k-h-1}].\end{aligned}$$

Given the independence of the noise terms ϵ_k, the expected values of products of non-identical noise terms are zero. Therefore, only terms where the indices of the noise terms coincide contribute to the autocovariance. For $h = 1$, the relevant term is:

$$\text{Cov}[Y_k, Y_{k-1}] = \theta\mathbb{E}[\epsilon_{k-1}^2] = \theta\sigma_\epsilon^2,$$

since the cross terms vanish due to independence. For $h \geq 2$, since all terms involve expectations of products of independent noise terms at different time points, all these terms vanish, resulting in $\text{Cov}[Y_k, Y_{k-h}] = 0$. Thus, the autocovariance and autocorrelation functions of the MA(1) process are:

$$C_{\mathcal{Y}}(h) = \begin{cases} \sigma_\epsilon^2(1+\theta^2) & \text{if } h = 0, \\ \theta\sigma_\epsilon^2 & \text{if } h = 1, \\ 0 & \text{if } h \geq 2. \end{cases} \qquad R_{\mathcal{Y}}(h) = \begin{cases} 1 & \text{if } h = 0, \\ \frac{\theta}{1+\theta^2} & \text{if } h = 1, \\ 0 & \text{if } h \geq 2. \end{cases}$$

This illustrates the finite dependence structure of the MA(1) process, where autocorrelations drop to zero beyond lag 1, highlighting the short memory inherent to moving average models.

2.2.3 Testing Stationarity

Testing for stationarity is a crucial step in time series analysis, as it directly impacts the choice of modeling techniques and the validity of subsequent analyses. Stationarity ensures that the statistical properties of a time series remain constant over time, which is a fundamental assumption for many forecasting models and statistical tests. Incorrectly assuming stationarity when it does not hold can lead to spurious results and unreliable predictions. Conversely, treating a stationary series as non-stationary may result in unnecessarily complex models and loss of efficiency in estimation. Therefore, it is crucial to determine whether the observed sample path is likely to be a realization of a stationary stochastic process.

Stationarity can be empirically verified using several methods, as detailed below:

1. **Basic Statistical Tests**: Stationarity implies that the mean, variance, and autocovariance of the time series do not vary with time. A preliminary assessment of stationarity can be performed using the following techniques:

   - **Plot the Time Series**: Visually inspect the time series to check whether it fluctuates around a constant mean and exhibits consistent variability. If clear trends or changes in variance are observed, the process is likely non-stationary.
   - **Rolling Statistics**: Compute and plot the rolling empirical mean and rolling empirical variance using a fixed window size. If the rolling statistics remain approximately constant, this is indicative of stationarity. Significant shifts in these statistics over time suggest non-stationarity.
   - **Autocorrelation Function (ACF)**: Plot the autocorrelation function to assess how the autocorrelations evolve with increasing lags. For a stationary process, the autocorrelation function typically decays towards zero relatively quickly. A persistent autocorrelation at higher lags may signal non-stationarity.

   While these methods provide initial insights, formal statistical tests are required to confirm stationarity.

   TBD: Appendix on multivariate hypothesis testing, p values, example with Granger's causality F-test...

2. **Augmented Dickey-Fuller (ADF) Test**: The ADF test is a commonly employed hypothesis test to formally assess the stationarity of a time series. The null and alternative hypotheses for the ADF test are:

   $$H_0 : (\text{the series is non-stationary}) \text{ vs. } H_A : (\text{the series is stationary}).$$

   The ADF test yields a p-value, and if this value falls below a chosen significance level (commonly 0.05), we reject the null hypothesis, indicating that the time series is stationary. While we do not cover the underlying theoretical foundations of the test here, the interested reader may explore the full details in [1]. This test is widely applied in practice and is available through the `adfuller` function in the Python `statsmodels` package. The function returns both the test statistic γ and the corresponding p-value, allowing users to formally assess the stationarity of a time series based on the test outcomes.

In addition to the ADF test, practitioners often employ the Phillips-Perron (PP) test (implemented in Python by the method `phillips_perron`), the Zivot-Andrews test (`zivot_andrews`), and the variance ratio test (`variance_ratio`), all available in the `statsmodels` library, to further assess stationarity under various conditions.

Python Lab: ADF test for AR(1) and Random Walk processes.

In this example, we will analyze two types of time series: An AR(1) process, which is stationary, and a random walk, which is non-stationary. We will use the Augmented Dickey-Fuller (ADF) test to determine whether each time series is stationary or not. Below is the step-by-step explanation of the code used.

1. **Import Libraries:** First, we import the necessary libraries for generating the time series and running the ADF test.

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.stattools import adfuller
```

   `adfuller` from `statsmodels` is the method we will use to perform the Augmented Dickey-Fuller test for stationarity.

2. **Generate Time Series Functions**: Next, we define two functions to generate the AR(1) and the random walk processes. The AR(1) process is generated using the recursion $Y_k = \phi Y_{k-1} + \epsilon_k$, and the random walk is generated using $Y_k = Y_{k-1} + \epsilon_k$.

```python
# Function to generate an AR(1) process
def generate_ar1(phi, sigma, n):
    ar1_process = np.zeros(n)
    epsilon = np.random.normal(0, sigma, n)
    for t in range(1, n):
        ar1_process[t] = phi * ar1_process[t-1] + epsilon[t]
    return ar1_process

# Function to generate a random walk process
def generate_random_walk(n):
    random_walk = np.zeros(n)
    epsilon = np.random.normal(0, 1, n)
    for t in range(1, n):
        random_walk[t] = random_walk[t-1] + epsilon[t]
    return random_walk
```

3. **Generate Time-Series Data**: We then generate both the AR(1) and random walk series. For the AR(1) process, we choose $\phi = 0.7$ and set the length of both series to $L = 1,000$ observations.

```
# Length of the series
L = 1000

# Generate AR(1) process with phi=0.7 and sigma=1
phi = 0.7
sigma = 1
ar1_series = generate_ar1(phi, sigma, L)

# Generate random walk series
random_walk_series = generate_random_walk(L)
```

4. **Define ADF Test Function**: Next, we define a helper function to run the ADF test and print the results. The function prints the ADF statistic, p-value, and other relevant details.

```
# Perform Augmented Dickey-Fuller test on both series
def adf_test(series, series_name):
    result = adfuller(series)
    print(f"ADF Test for {series_name}:")
    print(f"ADF Statistic: {result[0]}")
    print(f"p-value: {result[1]}")
    print(f"Lags used: {result[2]}")
    print(f"Number of observations: {result[3]}")
    print(f"Critical values: {result[4]}")
    if result[1] < 0.05:
        print(f"Conclusion: {series_name} is stationary (reject
            H0).")
    else:
        print(f"Conclusion: {series_name} is non-stationary (fail
            to reject H0).")
    print("\n")
```

This function runs the Augmented Dickey-Fuller (ADF) test on the given time series. The key outputs are:

- **ADF Statistic**: A large negative value indicates a stronger rejection of the null hypothesis (non-stationarity).
- ***p*-value**: If the p-value is below a chosen threshold (typically 0.05), we reject the null hypothesis and conclude that the series is stationary.
- **Lags used**: The number of lagged differences used in the test.

- **Critical values**: Threshold values at different significance levels (1%, 5%, 10%) that allow us to judge the ADF statistic.

5. **Apply ADF Test**: Finally, we apply the ADF test to both the AR(1) process and the random walk, and interpret the results.

```python
# ADF test on AR(1) process
adf_test(ar1_series, "AR(1) Process")

# ADF test on random walk process
adf_test(random_walk_series, "random walk")
```

The output of this last piece of code is the following:

```
ADF Test for AR(1) Process:       ADF Test for Random Walk (RW):

ADF Statistic:  -12.241           ADF Statistic:  -0.0939
p-value:  1.001869e-22            p-value:  0.95003604
Conclusion:  AR(1) is stationary  Conclusion:  RW is non-stationary
(reject H0).                      (fail to reject H0).
```

As expected from our theoretical analysis, the random walk process is non-stationary, while the AR(1) process is stationary—except for a brief transient period at the beginning. However, this initial phase is not significant enough to influence the result of the ADF test.

2.3 Markov Processes

Consider a stochastic process where the present time is denoted by the index k. A **Markov process** is a type of stochastic process in which the distribution of the next value, Y_{k+1}, depends only on the current value Y_k and not on any past values. Formally, the **information set**[4] of a process $\mathcal{Y}$ at time k is defined as:

$$\mathcal{F}_k = \{Y_k = y_k, Y_{k-1} = y_{k-1}, Y_{k-2} = y_{k-2}, \ldots\}.$$

t includes all the information gathered about the random process up to time k (time k included).

Mathematically, the process $\mathcal{Y} = \{Y_k : k \in \mathbb{N}\}$ is said to be **Markovian** if, for all $k \in \mathbb{N}$, the conditional probability distribution of the next observation Y_{k+1} depends only on the current observation Y_k. This is formally expressed as:

$$\mathbb{P}(Y_{k+1} \leq y \mid \mathcal{F}_k) = \mathbb{P}(Y_{k+1} \leq y \mid Y_k = y_k) \text{ for all } k \in \mathbb{N} \text{ and all } y \in \mathbb{R},$$

[4]A fundamental concept in the study of stochastic processes is the notion of information sets and *filtrations*. A **filtration** is a sequence of information sets that evolve over time, encapsulating the cumulative knowledge available up to each point. For a formal treatment of filtrations, from the point of view of σ-algebras, can be found in [3].

indicating that the observation Y_k encapsulates all the necessary information to predict the value of Y_{k+1}. In other words, *a process is Markovian if, given the current value Y_k, the past and future are conditionally independent of each other.* A Markovian process is often described as *memoryless* because the next variable in the process, Y_{k+1}, depends only on the current observation $Y_k = y_k$ and not on any past observations.

Example 8: AR(1) is a Markov Process

Consider the autoregressive model of order 1, AR(1), defined by the recursion:

$$Y_{k+1} = \phi Y_k + \epsilon_k, \quad \text{with } \epsilon_k \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma_\epsilon^2), Y_0 = y_0 \text{ and } |\phi| < 1.$$

The AR(1) model is a Markov process because the conditional distribution of Y_{k+1} depends only on the value of Y_k, not on any earlier values. Specifically, the conditional distribution of Y_{k+1}, given the past values $Y_k, Y_{k-1}, \ldots$, depends only on Y_k, due to the autoregressive nature of the model. Formally, the AR(1) model satisfies:

$$(Y_{k+1} \mid \mathcal{F}_k) = (Y_{k+1} \mid Y_k = y_k) \sim \mathcal{N}(\phi\, y_k, \sigma^2),$$

which is the defining characteristic of a *Markov process*.

2.3.1 Higher-Order Markov Processes

In some cases, the future value of a process may depend not only on the current value but also on a finite number of past values. Consider a stochastic process where the present time is denoted by the index k. Define the information set:

$$\mathcal{F}_{k,m} = \{Y_k = y_k, Y_{k-1} = y_{k-1}, \ldots, Y_{k-m} = y_{k-m}\},$$

i.e., the set including all the information gathered about the process during the previous m lags up to time k. Formally, a **higher-order Markov process** with **memory** m satisfies the condition:

$$\mathbb{P}(Y_{k+1} \leq y \mid \mathcal{F}_k) = \mathbb{P}(Y_{k+1} \leq y \mid \mathcal{F}_{k,m}) \text{ for all } k \in \mathbb{N} \text{ and all } y \in \mathbb{R}.$$

This implies that the conditional distribution given all past observations depends solely on the present value and the m most recent past values. Observations further back than m lags do not provide additional information for predicting future values. By convention, a Markov process with memory m is said to be of **order** $m + 1$.

Markov processes, due to their finite memory property, provide a practical framework for modeling time series with short-term dependencies. Higher-order Markov processes are particularly useful when the dependence on past values extends beyond just the immediate prior value, capturing more complex dynamics that simple first-order models may miss. This flexibility allows them to model systems where history plays a role over multiple time steps. For instance, in financial markets, higher-order models may capture the influence of trends or cycles over time. In speech recognition, higher-order models help account for dependencies spanning multiple phonemes or syllables. In biological processes, such models can describe phenomena where the outcome is influenced not only by the most recent state but also by patterns occurring over a longer horizon.

Example 9: AR(2) is a Second-Order Markov Process

Consider the autoregressive model of order 2, denoted by AR(2), and defined by the recursion:

$$Y_{k+1} = \phi_0 Y_k + \phi_1 Y_{k-1} + \epsilon_k, \quad \text{with } \epsilon_k \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma_\epsilon^2) \text{ and } |\phi| < 1.$$

This model describes a process where each observation is a linear combination of the previous two observations, plus a normally distributed error term. Note that the AR(2) process requires the first two consecutive observations to initialize the recursion. By convention, we assign time indices less than or equal to zero to specify the initial conditions, i.e., $Y_0 = y_0$ and $Y_{-1} = y_{-1}$. Since the future value Y_{k+1} depends only on the two most recent observations, AR(2) is a *second-order* Markov process. Formally, the AR(2) model satisfies:

$$(Y_{k+1} \mid \mathcal{F}_k) = (Y_{k+1} \mid Y_k = y_k, Y_{k-1} = y_{k-1}) \text{ for all } k \in \mathbb{N},$$

for all $k \in \mathbb{N}$, which is the defining characteristic of a *second-order Markov process*, i.e., a process with memory $m = 1$. In particular, the last conditional density function is given by a following Gaussian density with mean $\phi_0 y_k + \phi_1 y_{k-1}$ and variance σ_ϵ^2.

2.3.2 Markov Chains

A **Markov chain**, denoted by $\mathcal{X} = \{X_k : k \in \mathbb{N}\}$, is a specific type of Markov process where the system evolves over a countable set of *states*. For each $k \in \mathbb{N}$, the random variable X_k assumes values in a discrete **state space** $\mathcal{S} = \{s_1, s_2, \ldots\}$, where $\mathcal{S}$ is either finite or countably infinite. A fundamental characteristic of Markov chains is that they satisfy the **Markov property**, i.e., the system is *memoryless*. Specifically, the conditional distribution of the future state X_{k+1}, given both the past and present states, depends solely on the present state X_k. Formally, this is expressed as:

$$\mathbb{P}(X_{k+1} = x_{k+1} \mid \mathcal{F}_k) = \mathbb{P}(X_{k+1} = x_{k+1} \mid X_k = x_k).$$

Thus, the one-step-ahead distribution is determined entirely by the current state, independent of the preceding history.

At each time step k, the Markov chain transitions between states according to a set of **transition probabilities**[5]. These probabilities describe the likelihood of moving from one state s_i to another s_j at any given time step:

$$p_{ij} = \mathbb{P}(X_{k+1} = s_j \mid X_k = s_i), \qquad \text{for all } i, j \in \mathcal{S},\ k \in \mathbb{N}.$$

The transition probabilities can be arranged into a **transition matrix** $P \in [0,1]^{|\mathcal{S}| \times |\mathcal{S}|}$, where the (i,j)-th entry, $[P]_{ij} = p_{ij}$, represents the probability of transitioning from state s_i to state s_j. Importantly, P is a **row-stochastic** matrix, meaning its entries are nonnegative and the sum of each row equals 1:

$$\sum_{j \in \mathcal{S}} p_{ij} = 1 \qquad \text{for all } i \in \mathcal{S}.$$

This structure encapsulates the probabilistic dynamics governing the evolution of the Markov chain over time.

Example 10: Weather Prediction Using a Markov Chain

In this example, we aim to predict the weather tomorrow based on the weather conditions of today. The weather can transition between three distinct states: $\mathcal{S} = \{s_1 = \texttt{Sunny}, s_2 = \texttt{Cloudy}, s_3 = \texttt{Rainy}\}$. We assume that the weather dynamics follows a time-homogeneous Markov chain, where the transition probabilities between weather states are constant over time and depend only on the current weather condition.
The weather transitions are described by the following **transition matrix** P, which specifies the conditional probabilities of moving from one weather state to another:

$$P = \begin{bmatrix} 0.7 & 0.2 & 0.1 \\ 0.3 & 0.4 & 0.3 \\ 0.2 & 0.3 & 0.5 \end{bmatrix}$$

Each entry p_{ij} represents the probability of transitioning from state i to state j in one time step. For instance, if the weather today is in the `Sunny` state, there is a 70% chance that tomorrow will also be `Sunny`, a 20% chance of being `Cloudy`, and a 10% chance of being in the `Rainy` state.

[5]Here, we restrict our discussion to *homogeneous* Markov chains, where transition probabilities are time-invariant.

At time k, the **state probability distribution** of a Markov chain $\mathcal{X}$ is represented by the probability vector $\boldsymbol{\pi}_k \in [0,1]^{|\mathcal{S}|}$, where each component $\pi_{k,i}$ denotes the likelihood that the system occupies state s_i at time k, conditioned on the initial information $\mathcal{F}_0$. More formally, the state probability distribution is expressed as:

$$\boldsymbol{\pi}_k = \left[\mathbb{P}(X_k = s_1 \mid \mathcal{F}_0), \quad \mathbb{P}(X_k = s_2 \mid \mathcal{F}_0), \quad \cdots, \quad \mathbb{P}(X_k = s_n \mid \mathcal{F}_0)\right]^\intercal.$$

This vector is stochastic, meaning that the sum of its entries is always equal to 1. The evolution of the state distribution vector is governed bt the transition matrix P, as follows. Using the total probability theorem, the probability that the system is in state s_j at time k, can be expanded as:

$$\underbrace{\mathbb{P}(X_k = s_j \mid \mathcal{F}_0)}_{\pi_{k,j}} = \sum_{i=1}^{n} \underbrace{\mathbb{P}(X_k = s_j \mid X_{k-1} = s_i)}_{p_{ij}} \underbrace{\mathbb{P}(X_{k-1} = s_i \mid \mathcal{F}_0)}_{\pi_{k-1,i}},$$

for all $s_j \in \mathcal{S}$ and all $k \in \mathbb{N}$. The last equation results in $\pi_{k,j} = \sum_{i=1}^{n} p_{ij}\pi_{k-1,i}$. Using the rules of matrix-vector multiplication, this summation can be interpreted as the j-th entry of the vector resulting from the product $P^\intercal \boldsymbol{\pi}_{k-1}$. Therefore, we have that:

$$\boldsymbol{\pi}_k = P^\intercal \boldsymbol{\pi}_{k-1}. \tag{2}$$

In other words, the state probability distribution at time k can be computed by pre-multiplying the state probability distribution at time $k-1$ by the transpose of the transition matrix P, providing a clear and concise mathematical mechanism for tracking the system's evolution.

This recursive relationship allows us to express the state probability distribution at any time step k in terms of the initial distribution $\boldsymbol{\pi}_0$. By applying the recursive formula in (2) iteratively, we derive the following result through induction:

$$\boldsymbol{\pi}_1 = P^\intercal \boldsymbol{\pi}_0 \Rightarrow \boldsymbol{\pi}_2 = P^\intercal \boldsymbol{\pi}_1 = (P^\intercal)^2 \boldsymbol{\pi}_0 \Rightarrow \cdots \Rightarrow \quad \boxed{\boldsymbol{\pi}_k = (P^\intercal)^k \boldsymbol{\pi}_0}.$$

This compact formula provides a powerful tool for analyzing the state distribution at any given time, based solely on the initial distribution and the system's transition dynamics. In this expression, $(P^\intercal)^k$ is the called the **k-step transition matrix**. Specifically, the (i,j)-th entry of $(P^\intercal)^k$, $[(P^\intercal)^k]_{ij}$, represents the probability of transitioning from state s_i at time 0 to state s_j at time k. Mathematically, this can be expressed as:

$$[P^k]_{ij} = \mathbb{P}(X_k = s_j \mid X_0 = s_i).$$

In other words, the k-step transition matrix encodes the probabilities of moving between any two states over a time span of k steps. This is an extension of the one-step transition matrix P, where the probabilities for longer time steps can be obtained by raising $P^\intercal$ to the power of k.

2.3.3 Higher-Order Markov Chains

A higher-order Markov chain generalizes the concept of a first-order Markov chain by considering the influence of multiple past states on the current state. In Markov chain with memory m, the future state depends not only on the current state but also on the previous m states. Mathematically, this is expressed as:

$$\mathbb{P}(X_{k+1} \mid \mathcal{F}_k) = \mathbb{P}(X_{k+1} \mid X_k = x_k, X_{k-1} = x_{k-1}, \ldots, X_{k-m} = x_{k-m}),$$

where the one-step-ahead probability mass function (PMF) depends on present and the previous m states. Higher-order Markov chains can capture more complex dependencies over time, making them useful for modeling processes where the system's evolution is influenced by longer histories.

In order to reduce a higher-order Markov chain to an equivalent first-order chain, we augment the size of the state space by incorporating the memory of previous states. This process allows us to retain the essential structure of the higher-order dependencies while transforming the problem into a first-order Markov chain framework, where the Markov property holds. This state space augmentation is performed as follows: Consider a Markov chain with state space $\mathcal{S}$ and memory m. To reduce this process into a first-order Markov chain, we define a new state space as the *Cartesian product*[6] of the original state space with itself m times, resulting in $\mathcal{S}^{m+1} = \mathcal{S} \times \cdots \times \mathcal{S}$. In other words, the augmented state space $\mathcal{S}^{m+1}$ is defined as:

$$\mathcal{S}^{m+1} = \{(s_k, s_{k-1}, \ldots, s_{k-m}) \colon s_i \in \mathcal{S} \text{ for all } i = k, k-1, \ldots, k-m\}.$$

If the Markov chain has memory m, all the necessary information for predicting the future state is encoded into a single composite state.

Example 11: Weather Prediction Using a Second-Order Markov Chain

Extending on Example 10, we now consider a more complex weather model where the weather on a given day depends not only on today's weather but also on yesterday's. The new weather model is a *second-order Markov chain* (with memory $m = 1$), where the future state of the system depends on the last two days of weather conditions. The same set of weather states applies: $\mathcal{S} = \{s_1 = \texttt{Sunny}, s_2 = \texttt{Cloudy}, s_3 = \texttt{Rainy}\}$, but the system now uses information from both yesterday and today to predict tomorrow's weather. To manage this second-order dependency, we expand the state space by considering ordered pairs of weather states from the last two days. The new state space $\mathcal{S}^2$ is the Cartesian product of the original state space $\mathcal{S}$ with itself:

$$\mathcal{S}^2 = \mathcal{S} \times \mathcal{S} = \{(s_1, s_1), (s_1, s_2), (s_1, s_3), (s_2, s_1), \ldots, (s_3, s_3)\}.$$

[6]The Cartesian power $\mathcal{S}^q$ consists of all q-tuples $(s_{i_1}, s_{i_2}, \ldots, s_{i_q})$ where $s_{i_r} \in \mathcal{S}$ for all r.

This yields $|\mathcal{S}|^2 = 9$ possible states, each representing the weather over the last two days. For example, the state (s_1, s_2) indicates that today is `Sunny` (state s_1) and yesterday was `Cloudy` (state s_2).
The transition probabilities in this second-order model is a 9×9 matrix where each entry describe the probability of transitioning from a 2-tuple of states to a new 2-tuple of states. Note that it is not possible to transition from any 2-tuple to any other 2-tuples. In particular, if the current state is (s_k, s_{k-1}), the next state is formed by shifting the history and incorporating tomorrow's weather, resulting in (s_{k+1}, s_k). Therefore, the first entry in the current 2-tuple state is the same as the second entry in tomorrow's 2-tuple state.

The transition probabilities in the augmented state space $\mathcal{S}^{m+1}$ are defined between pairs of composite states. Specifically, if the current composite state is the $(m+1)$-tuple $\mathbf{z}_{k,m} = (x_k, x_{k-1}, \ldots, x_{k-m}) \in \mathcal{S}^{m+1}$, the next composite state is formed by shifting the $m+1$-tuple forward and appending the next state x_{k+1}, resulting in the new composite state $\mathbf{z}_{k+1,m} = (x_{k+1}, x_k, \ldots, x_{k-m+1})$. The transition probability from the current composite state to the next composite state is governed by the higher-order Markov transition probabilities:

$$\mathbb{P}\left(\mathbf{Z}_{k+1,m} \mid \mathbf{Z}_{k,m}\right) = \mathbb{P}(X_{k+1} \mid X_k, X_{k-1}, \ldots, X_{k-m}),$$

where $\mathbf{Z}_{k,m} = (X_k, X_{k-1}, \ldots, X_{k-m})$ denotes an $(m+1)$-tuple of random variables for all $k, m \in \mathbb{N}$. Therefore, the augmented first-order Markov chain is characterized by transitions among the composite states in $\mathcal{S}^{m+1}$, with transition probabilities derived from the original higher-order process.

In the case of a higher-order Markov chain, the transition probabilities can be arranged into a transition matrix P. The number of rows and columns in this augmented transition matrix is equal to the dimension of the set of composite states, i.e., $|\mathcal{S}|^{m+1}$. Therefore, the augmented transition matrix contains $|\mathcal{S}|^{2(m+1)}$ entries. Alternatively, we can use tensor notation to arrange the set of transition probabilities more effectively. Specifically, a **transition tensor**[7] $\mathcal{P}$ is used to capture the transition probabilities, where the tensor has $m+2$ dimensions to account for all possible prior states. Formally, the tensor $\mathcal{P} \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}| \times \cdots \times |\mathcal{S}|}$ (with $m+2$ dimensions) is defined such that each entry $\mathcal{P}_{i_0 i_1 i_2 \ldots i_m, j}$ represents the following transition probability

$$\mathcal{P}_{i_0 i_1 \ldots i_m, j} = \mathbb{P}(X_{k+1} = s_j \mid \mathbf{Z}_{k,m} = (s_{i_0}, s_{i_1}, \ldots, s_{i_m})).$$

Thus, the tensor $\mathcal{P}$ organizes all the transition probabilities in a Markov chain with memory m in an orderly fashion.

[7] A tensor is a generalization of matrices to higher dimensions. While a matrix is a two-dimensional array (with rows and columns), a tensor can have three or more dimensions, making it a multidimensional array. Tensors are used to represent data with multiple indices.

The state probability distribution of a higher-order Markov chain can be trivially defined as a stochastic vector of dimension $|\mathcal{S}|^{m+1}$ by assigning an arbitrary order to the elements in the augmented set of states, $|\mathcal{S}|^{m+1}$. A more natural representation of the probability distribution of a higher-order Markov chain at time k is the **probability tensor** $\boldsymbol{\pi}_k \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|\times\cdots\times|\mathcal{S}|}$ (with $m+1$ dimensions). Each entry $\boldsymbol{\pi}_{k,i_0 i_1\ldots i_m}$ represents the joint probability:

$$\boldsymbol{\pi}_{k,i_0 i_1\ldots i_m} = \mathbb{P}(\mathbf{Z}_{k,m} = (s_{i_0}, s_{i_1}, \ldots, s_{i_m}) \mid \mathcal{F}_0),$$

where $\mathcal{F}_0$ is the initial information set.

To propagate the **state distribution tensor** forward in time, we perform a tensor update using the transition tensor $\mathcal{P}$ and the current state distribution tensor $\boldsymbol{\pi}_k$, as follows. Applying the total probability theorem, we have that

$$\begin{aligned}\boldsymbol{\pi}_{k+1,j\,i_0 i_1\ldots i_{m-1}} &= \mathbb{P}\left(\mathbf{Z}_{k+1,m} = (j, i_0, i_1, \ldots, i_{m-1}) \mid \mathcal{F}_0\right) \\ &= \sum_{\mathbf{z}_{k,m}\in\mathcal{S}^{m+1}} \mathbb{P}\left(\mathbf{Z}_{k+1,m} = (j, i_0, i_1, \ldots, i_{m-1}) \mid \mathbf{Z}_{k,m} = \mathbf{z}_{k,m}\right) \\ &\qquad \times \mathbb{P}\left(\mathbf{Z}_{k,m} = \mathbf{z}_{k,m} \mid \mathcal{F}_0\right). \end{aligned} \tag{3}$$

Note that the term $\mathbb{P}\left(\mathbf{Z}_{k+1,m} = (j, i_0, i_1, \ldots, i_{m-1}) \mid \mathbf{Z}_{k,m} = \mathbf{z}_{k,m}, \mathcal{F}_0\right)$ is equal to zero in most cases; in particular, that transition can be non-zero when $\mathbf{z}_{k,m} = (i_0, i_1, \ldots, i_m)$. Therefore, all but the last entry of $\mathbf{z}_{k,m}$ take predefined values. The summation in (3) is thus reduced to a summation over the last entry of $\mathbf{z}_{k,m}$, as follows:

$$\begin{aligned}\boldsymbol{\pi}_{k+1,j\,i_0 i_1\ldots i_{m-1}} = \sum_{i_m\in\mathcal{S}} &\underbrace{\mathbb{P}\left(\mathbf{Z}_{k+1,m} = (j, i_0, i_1, \ldots, i_{m-1}) \mid \mathbf{Z}_{k,m} = (i_0, i_1, \ldots, i_m)\right)}_{\mathcal{P}_{i_0 i_1\ldots i_m,j}} \\ &\times \underbrace{\mathbb{P}\left(\mathbf{Z}_{k,m} = (i_0, i_1, \ldots, i_m) \mid \mathcal{F}_0\right)}_{\boldsymbol{\pi}_{k,i_0 i_1\ldots i_m}}.\end{aligned}$$

In conclusion, we can propagate the **state distribution tensor** forward in time using the following expression:

$$\boxed{\boldsymbol{\pi}_{k+1,j\,i_0 i_1\ldots i_{m-1}} = \sum_{i_m\in\mathcal{S}} \mathcal{P}_{i_0 i_1\ldots i_m,j}\boldsymbol{\pi}_{k+1,i_0 i_1\ldots i_m}}$$

Notice how this expression is a natural extension of expression (2) obtained for the memoryless (i.e., $m = 0$) Markov chain.

Given the exponential increase in dimensionality as m grows, storing the transition tensor can become computationally expensive, particularly if the state space $\mathcal{S}$ is large. However, in many practical situations, the tensor may be sparse (i.e., many transitions have zero probability), and sparse storage techniques can be used to store the tensor more efficiently. While this expanded state space $\mathcal{S}^m$ grows exponentially fast, transforming the higher-order Markov chain into a first-order chain by expanding the state space allows us to apply the standard tools and methods from first-order Markov chain theory. The trade-off is that we now work with a larger state space, but the benefit is that the system retains the simplicity of the Markov property, where the future state depends solely on the current composite state, encapsulating all the necessary history in one step.

3 Multivariate Time Series

In many real-world applications, it is common to handle multiple time-dependent variables that evolve together, with each potentially influencing the others. A **multivariate time series** involves observing and analyzing multiple interrelated time-dependent variables simultaneously. Unlike a univariate time series, which focuses on a single variable, multivariate time series account for several variables together, enabling the study of potential interactions and dynamic dependencies among them. The strength of multivariate analysis lies in its ability to capture these interdependencies over time. This is especially valuable in fields such as economics, where inflation, interest rates, and exchange rates interact; in finance, where stock prices, trading volume, and volatility influence each other; and in environmental science, where temperature, humidity, and atmospheric pressure are interrelated. For instance, consider three interdependent time series representing the evolution of energy demand, temperature, and humidity in a particular region (see Fig. 6), where changes in one variable influence the behavior of the others, capturing dynamic interdependencies that a univariate analysis would miss.

Let us now formalize the concept of a multivariate time series. Consider a collection of C time series, each of length L, denoted by $\mathcal{Y}^1, \mathcal{Y}^2, \ldots, \mathcal{Y}^C$, co-evolving over discrete time[8]. Each of these time series can influence or be influenced by the others, capturing the underlying structure of their interaction. We define a single vector-valued time series as $\boldsymbol{\mathcal{Y}} = (\mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_L)$, where each vector $\mathbf{Y}_k = [Y_k^1, Y_k^2, \ldots, Y_k^C]^\intercal$ contains the values of the C time series at a given time k. Specifically, Y_k^c denotes the value of the c-th time series (also referred to as the **component**) at time step k, where $1 \leq c \leq C$ and $1 \leq k \leq L$. This vector-valued representation allows us to capture the joint behavior of all variables at each time step, providing a comprehensive framework for analyzing their interactions.

Multivariate time series analysis provides enhanced predictive power compared to univariate methods, as it leverages the correlations and interactions between

[8]In our exposition, we assume that all $\mathcal{Y}^1, \ldots, \mathcal{Y}^C$ are random processes evolving on the same probability space, and that their means, variances, and covariances exist and are finite. These assumptions are crucial to ensure the proper definition of second-order statistics, such as covariance matrices, that are widely used in multivariate analysis.

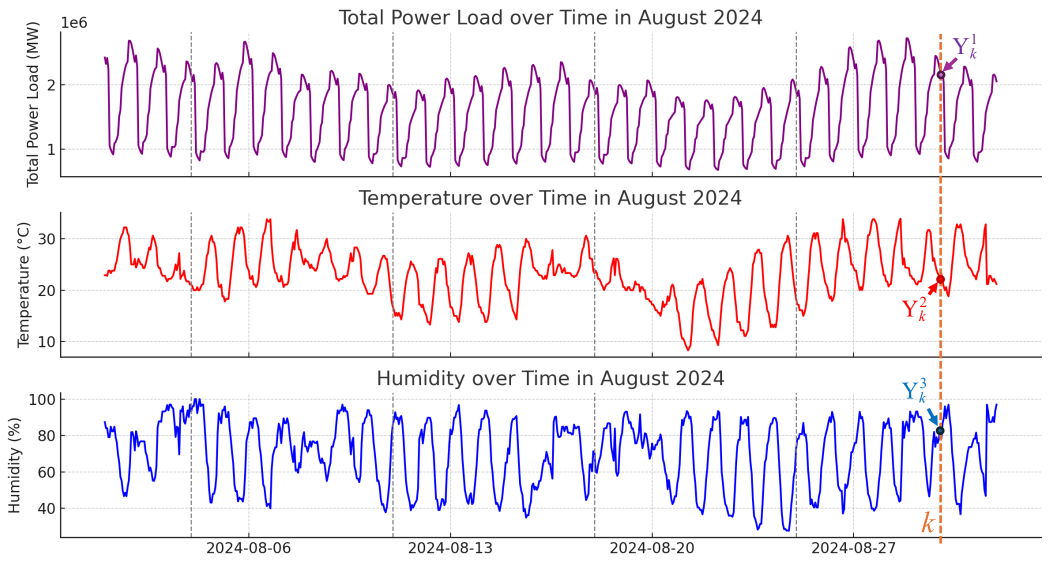

Figure 6: Hourly evolution of a multivariate time series with $C = 3$ components: Power Load (top, in MW), Temperature (middle, in Celsius), and Humidity (bottom, in %) in the Mid-Atlantic Region during August 2024. These three variables are known to be interrelated, as energy demand can be affected by both temperature and humidity. (This multivariate time series is available in GitHub)

different variables. This ability to model dynamic dependencies between variables over time is crucial in understanding complex systems, leading to more accurate forecasts and a better grasp of the underlying processes.

3.1 Statistical Properties of Multivariate Time Series

A complete statistical description of a multivariate random process can be expressed in terms of its joint cumulative distribution function (CDF), which provides the probability that all components of the time series are less than or equal to certain values over time. Consider a multivariate process $\boldsymbol{\mathcal{Y}}$ of length L with C components. The full statistical behavior of the process is captured in the following joint CDF:

$$F_{\boldsymbol{\mathcal{Y}}}(\mathbf{y}_1, \ldots, \mathbf{y}_L) = \mathbb{P}(\mathbf{Y}_1 \leq \mathbf{y}_1, \ldots, \mathbf{Y}_L \leq \mathbf{y}_L).$$

The arguments of this CDF are C-dimensional vectors and the inequalities in the right-hand side are interpreted component-wise. This joint distribution encompasses the entire set of variables across all time steps and dimensions, thus characterizing both temporal and cross-variable dependencies within the system. It offers a complete picture of the stochastic behavior and interdependencies of the process.

In practice, specifying the full joint distribution for a multivariate process becomes intractable due to its high dimensionality and complexity, especially as both the number of time steps L and components C increase. Therefore, simplified statistical tools, such as means and covariances, are typically employed to capture essential aspects of the relationships between components and their temporal dynamics. Below, we outline the most commonly used statistical measures in the analysis of multivariate stochastic processes:

- **Mean Vector:** The mean vector $\boldsymbol{\mu}_{\boldsymbol{\mathcal{Y}}}(k)$ represents the expected values for each component of the multivariate time series at a given time k. It is expressed as:
$$\boldsymbol{\mu}_{\boldsymbol{\mathcal{Y}}}(k) = \mathbb{E}[\mathbf{Y}_k] = \left[\mathbb{E}[Y_k^1], \mathbb{E}[Y_k^2], \ldots, \mathbb{E}[Y_k^C]\right]^\intercal \in \mathbb{R}^C.$$
This vector encapsulates the average behavior of each component over time, providing a foundational perspective on trends in the data. For notational convenience, we define $\mu_k^i = \mathbb{E}[Y_k^i]$.

- **Autocovariance Matrix:** The autocovariance matrix $C_{\boldsymbol{\mathcal{Y}}}(k,h)$ of a multivariate time series measures the co-variation of the components across two time points, k and $k-h$. Formally, it is defined as:
$$C_{\boldsymbol{\mathcal{Y}}}(k,h) = \mathbb{E}\left[(\mathbf{Y}_k - \boldsymbol{\mu}_{\boldsymbol{\mathcal{Y}}}(k))(\mathbf{Y}_{k-h} - \boldsymbol{\mu}_{\boldsymbol{\mathcal{Y}}}(k-h))^\intercal\right].$$
Notice that this is a matrix whose entries are functions of k and h. The autocovariance matrix captures both the variability within individual components (through the diagonal elements) and the interactions between different components (through the off-diagonal elements). The diagonal elements of the matrix satisfy:
$$[C_{\boldsymbol{\mathcal{Y}}}(k,h)]_{ii} = \mathbb{E}\left[(Y_k^i - \mu_k^i)(Y_{k-h}^i - \mu_{k-h}^i)\right].$$
Notice that this is exactly the expression for the autocovariance of the i-th component of the time series at time k and time $k-h$. For $h=0$, this expression reduces to:
$$[C_{\boldsymbol{\mathcal{Y}}}(k,0)]_{ii} = \mathbb{E}[(Y_k^i - \mu_k^i)^2] = \mathrm{Var}[Y_k^i],$$
i.e., the variance of the i-th component at time k. In summary, the diagonal elements of the autocovariance matrix describe the variability of each individual time series component and its temporal autocorrelation.

  On the other hand, the off-diagonal elements satisfy:
$$[C_{\boldsymbol{\mathcal{Y}}}(h)]_{ij} = \mathbb{E}\left[(Y_k^i - \mu_k^i)(Y_{k-h}^j - \mu_{k-h}^j)\right].$$
This term is called the **cross-covariance** between the i-th and j-th components of the time series at time k and $k-h$. The off-diagonal elements quantify the degree to which two different components of the multivariate time series

move together over time. A positive off-diagonal value indicates that the components tend to increase or decrease together, while a negative value suggests that when one component increases, the other decreases. In summary, the off-diagonal elements reveal the extent of interaction or co-movement between different components of the multivariate process at different lags.

- **Autocorrelation Matrix:** While cross-covariance provides valuable information about the relationship between different components of a multivariate time series, it can be challenging to interpret due to its dependence on the scale of the variables. To address this, we introduce the concept of **cross-correlation**, which is a normalized version of cross-covariance. The cross-correlation between the i-th and j-th components at lag h is defined as:

  $$[R_{\boldsymbol{\mathcal{Y}}}(k,h)]_{ij} = \frac{[C_{\boldsymbol{\mathcal{Y}}}(k,h)]_{ij}}{\sqrt{[C_{\boldsymbol{\mathcal{Y}}}(k,0)]_{ii}[C_{\boldsymbol{\mathcal{Y}}}(k-h,0)]_{jj}}} \in [-1,1].$$

  The cross-correlation function provides insights into the strength and direction of the linear relationship between different components of the multivariate time series at various time lags, facilitating the analysis of lead-lag relationships and interdependencies within the system.

In conclusion, statistical measures such as the autocovariance and the cross-covariance provide valuable insights into both the internal variability of each component and the interactions between different components. The diagonal elements of the autocovariance matrix capture the variability of individual series, while the off-diagonal elements represent the co-movement between different series, highlighting their interdependencies. These measures allow researchers to gain a deeper understanding of complex multivariate systems, uncover hidden dependencies, and improve forecasting accuracy across time.

3.2 Stationarity in Multivariate Processes

We can extend the definition of stationarity from univariate to multivariate stochastic processes, as follows. A multivariate stochastic process $\boldsymbol{\mathcal{Y}} = \{\mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_L\}$ is **strong-sense stationary** if, for any time indices $k_1, k_2, \ldots, k_n$ and for any integer h, the joint CDF satisfies:

$$F_{\mathcal{Y}}(\mathbf{y}_{k_1}, \mathbf{y}_{k_2}, \ldots, \mathbf{y}_{k_n}) = F_{\mathcal{Y}}(\mathbf{y}_{k_1+h}, \mathbf{y}_{k_2+h}, \ldots, \mathbf{y}_{k_n+h}),$$

for all $n \geq 1$ and for all possible time shifts h. This condition implies that the statistical properties of the process, as captured by the joint CDF, are invariant to shifts in time.

A multivariate stochastic process $\mathcal{Y} = \{\mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_n\}$ is said to be **weak-sense stationary** (or wide-sense stationary) if the mean vector is a time-invariant vector and the autocovariance matrix depend only on the time difference between the samples, not on absolute time. Formally, the process is weak-sense stationary if:

1. The mean vector $\mathbb{E}[\mathbf{Y}(k)] = \boldsymbol{\mu}_{\boldsymbol{\mathcal{Y}}} \in \mathbb{R}^C$ does not depend on k.
2. The autocovariance matrix, defined as:

$$C_{\boldsymbol{\mathcal{Y}}}(k,h) = \mathbb{E}[(\mathbf{Y}(k) - \boldsymbol{\mu}_{\boldsymbol{\mathcal{Y}}})(\mathbf{Y}(k-h) - \boldsymbol{\mu}_{\mathcal{Y}})^\intercal],$$

   depends only on the lag h. Whenever the multivariate process is WSS, we will simplify the notation of the covariance matrix to $C_{\boldsymbol{\mathcal{Y}}}(h)$, i.e., we remove the argument k. Therefore, each entry of the autocovariance matrix is a function of h. In Fig. 9, we plot these functions for a 2×2 autocovariance matrix.

Thus, in a WSS process, the mean and variance are time-independent, while the autocorrelation (and cross-correlation) are shift-invariant. This weaker form of stationarity is often sufficient for many practical applications, particularly in the context of linear time series modeling. To numerically test for stationarity in an empirical multivariate time series, one would typically check the stationarity of each component individually (e.g., using the Augmented Dickey-Fuller test) and examine the time-invariance of the cross-covariances.

Example 12: Mean and Autocovariance of a VAR(1) Process

Consider a **Vector Autoregressive process of order 1**, denoted as VAR(1), with C components, defined as:

$$\mathbf{Y}_k = A\mathbf{Y}_{k-1} + \boldsymbol{\epsilon}_k,$$

where $\mathbf{Y}_k = [Y_k^1, \ldots, Y_k^C]^\intercal \in \mathbb{R}^C$, $A \in \mathbb{R}^{C\times C}$ is the **autoregressive coefficient matrix**, and $\boldsymbol{\epsilon}_k \sim_{iid} \mathcal{N}(\mathbf{0}_C, \Sigma_\epsilon)$ is a *vector-valued* white noise whose vector mean is the C-dimensional vector of all zeros, denote by $\mathbf{0}_C$, and covariance matrix $\Sigma_\epsilon \in \mathbb{R}^{C\times C}$. Assume that the initial condition $\mathbf{Y}_0$ is a random variable with mean $\boldsymbol{\mu}_0$ and covariance matrix $C_{\mathbf{Y}_0}$. In order to ensure that the process converges to a stationary process, the eigenvalues of A must have magnitude less than one; otherwise, the random process grows unbounded. The mean and autocovariance of the VAR(1) process are derived as follows:

- **Mean:** The mean of the process at time k, denoted as $\boldsymbol{\mu}_k = \mathbb{E}[\mathbf{Y}_k]$, follows the recursion $\boldsymbol{\mu}_k = A\boldsymbol{\mu}_{k-1}$. Assuming stationarity, the mean $\boldsymbol{\mu}_\infty = \lim_{k\to\infty} \mathbb{E}[\mathbf{Y}_k]$ satisfies the vector equation $\boldsymbol{\mu}_\infty = A\boldsymbol{\mu}_\infty$, which implies that the stationary mean is $\boldsymbol{\mu}_\infty = \mathbf{0}_C$ (i.e., the vector of all zeros).
- **Autocovariance Matrix:** The autocovariance matrix of the VAR(1)

process, much like in the AR(1) process, experiences a transient phase before reaching stationarity exponentially fast. In the stationary phase, the autocovariance matrix at lag h is defined as:

$$C_{\boldsymbol{\mathcal{Y}}}(h) = \lim_{k\to\infty} \mathbb{E}\left[\mathbf{Y}_k \mathbf{Y}_{k-h}^\intercal\right].$$

The *stationary* covariance matrix at lag 0, $C_{\boldsymbol{\mathcal{Y}}}(0)$, satisfies the following equation:

$$\begin{aligned} C_{\boldsymbol{\mathcal{Y}}}(0) &= \mathbb{E}\left[(A\mathbf{Y}_{k-1} + \boldsymbol{\epsilon}_k)(A\mathbf{Y}_{k-1} + \boldsymbol{\epsilon}_k)^\intercal\right] \\ &= A\mathbb{E}\left[\mathbf{Y}_{k-1}\mathbf{Y}_{k-1}^\intercal\right] A^\intercal + \mathbb{E}\left[\boldsymbol{\epsilon}_k\boldsymbol{\epsilon}_k^\intercal\right] \\ &= AC_{\boldsymbol{\mathcal{Y}}}(0)A^\intercal + \Sigma_\epsilon. \end{aligned}$$

This matrix equation, called a **Lyapunov** equation, can be solved numerically to determine $C_{\boldsymbol{\mathcal{Y}}}(0)$.

To derive an expression for the stationary autocovariance matrix, $C_{\boldsymbol{\mathcal{Y}}}(h)$, at any lag $h \neq 0$, we can use the following expression for $\mathbf{Y}_k$ as a function of $\mathbf{Y}_{k-h}$ (left as an exercise for the reader):

$$\mathbf{Y}_k = A^h\mathbf{Y}_{k-h} + \sum_{i=1}^{h} A^{h-i}\boldsymbol{\epsilon}_{k-h+i}.$$

Therefore, the autocovariance matrix for any $h \neq 0$ satisfies:

$$\begin{aligned} C_{\boldsymbol{\mathcal{Y}}}(h) &= \mathbb{E}\left[\mathbf{Y}_k\mathbf{Y}_{k-h}^\intercal\right] \\ &= \mathbb{E}\left[\left(A^h\mathbf{Y}_{k-h} + \sum_{i=1}^{h} A^{h-i}\boldsymbol{\epsilon}_{k-h+i}\right)\mathbf{Y}_{k-h}^\intercal\right] \\ &= A^h\mathbb{E}\left[\mathbf{Y}_{k-h}\mathbf{Y}_{k-h}^\intercal\right] + \sum_{i=1}^{h} A^{h-i}\mathbb{E}\left[\boldsymbol{\epsilon}_{k-h+i}\mathbf{Y}_{k-h}^\intercal\right] \\ &= A^hC_{\boldsymbol{\mathcal{Y}}}(0), \end{aligned}$$

where, in the last identity, the summation disappears because $\boldsymbol{\epsilon}_{k-h+i}$ and $\mathbf{Y}_{k-h}^\intercal$ are uncorrelated for all $i \geq 1$.

In conclusion, the autocovariance matrix for any $h \neq 0$ can be computed as:

$$C_{\boldsymbol{\mathcal{Y}}}(h) = A^hC_{\boldsymbol{\mathcal{Y}}}(0),$$

where $C_{\boldsymbol{\mathcal{Y}}}(0)$ is the solution to the Lyapunov equation: $C_{\boldsymbol{\mathcal{Y}}}(0) = AC_{\boldsymbol{\mathcal{Y}}}(0)A^\intercal + \Sigma_\epsilon$.

Python Lab: VAR(1) Process Analysis

In this example, a synthetic multivariate time series with $C = 2$ components is generated using a VAR(1) process in Python. In particular, the VAR(1) process is defined as the following vector-valued recursion:

$$\mathbf{Y}_k = A\mathbf{Y}_{k-1} + \boldsymbol{\epsilon}_k, \quad \text{with } \mathbf{Y}_0 = \mathbf{0}_2 = \begin{bmatrix} 0 & 0 \end{bmatrix}^\intercal .$$

where:

$$\mathbf{Y}_k = \begin{bmatrix} Y_k^1 \\ Y_k^2 \end{bmatrix}, \quad A = \begin{bmatrix} 0.7 & 0.1 \\ 0.3 & 0.5 \end{bmatrix}, \quad \boldsymbol{\epsilon}_k = \begin{bmatrix} \epsilon_k^1 \\ \epsilon_k^2 \end{bmatrix} \text{ with } \boldsymbol{\epsilon}_k \overset{\text{iid}}{\sim} \mathcal{N}(\mathbf{0}, \mathbb{I}_2).$$

The vector recursion can be expanded into the following two scalar recursions:

$$Y_k^1 = 0.7Y_{k-1}^1 + 0.1Y_{k-1}^2 + \epsilon_k^1, \quad \epsilon_k^1 \overset{\text{iid}}{\sim} \mathcal{N}(0,1), \tag{4}$$

$$Y_k^2 = 0.3Y_{k-1}^1 + 0.5Y_{k-1}^2 + \epsilon_k^2, \quad \epsilon_k^2 \overset{\text{iid}}{\sim} \mathcal{N}(0,1). \tag{5}$$

In this system, both $\mathcal{Y}^1$ and $\mathcal{Y}^2$ evolve over time, each influenced by their own past values and by the past values of the other component. This reflects the interdependent nature of the components in a VAR(1) model, where both processes share information and influence each other across time steps, albeit to different degrees. For example, in the first equation, Y_k^1 depends strongly on its own past value Y_{k-1}^1, as indicated by the coefficient 0.7, and weakly on the past value of Y_{k-1}^2, with a smaller coefficient of 0.1. The random noise term ϵ_k^1 introduces some level of unpredictability at each time step.

Furthermore, based on the derivations in Example 12, we can compute the autocovariance matrix of the VAR(1) process at lag 0 by solving the following Lyapunov equation:

$$C_{\boldsymbol{\mathcal{Y}}}(0) = AC_{\boldsymbol{\mathcal{Y}}}(0)A^\intercal + \mathbb{I}_2,$$

This matrix equation expands into four scalar equations by performing matrix multiplications, as follows:

$$\begin{aligned}
C_{11}(0) &= 0.7^2 C_{11}(0) + 0.7 \cdot 0.1 \cdot C_{12}(0) + 0.7 \cdot 0.1 \cdot C_{21}(0) + 0.1^2 C_{22}(0) + 1, \\
C_{12}(0) &= 0.7 \cdot 0.3 \cdot C_{11}(0) + 0.7 \cdot 0.5 \cdot C_{12}(0) + 0.1 \cdot 0.3 \cdot C_{21}(0) + 0.1 \cdot 0.5 \cdot C_{22}(0), \\
C_{21}(0) &= 0.3 \cdot 0.7 \cdot C_{11}(0) + 0.3 \cdot 0.1 \cdot C_{12}(0) + 0.5 \cdot 0.7 \cdot C_{21}(0) + 0.5 \cdot 0.1 \cdot C_{22}(0), \\
C_{22}(0) &= 0.3^2 C_{11}(0) + 0.3 \cdot 0.5 \cdot C_{12}(0) + 0.5 \cdot 0.3 \cdot C_{21}(0) + 0.5^2 C_{22}(0) + 1,
\end{aligned}$$

where, $C_{11}(0)$, $C_{12}(0)$, $C_{21}(0)$, and $C_{22}(0)$ represent the elements of the autocovariance matrix $C_{\boldsymbol{\mathcal{Y}}}(0)$ at lag 0. The four equations above form a system of four *linear equations* with four unknowns. This system can be solved to determine[9] the elements of the autocovariance matrix $C_{\boldsymbol{\mathcal{Y}}}(0)$ at lag 0.

[9]This system of linear equations will have a unique solution if and only if all the eigenvalues of the matrix A have magnitudes strictly less than 1. This condition also ensures that the VAR(1) process converges to a *stationary* process.

To solve the Lyapunov equation for the autocovariance matrix, we can use the `solve_discrete_lyapunov` function from the `scipy.linalg` module in Python. Below is the Python code to do so:

```python
import numpy as np
from scipy.linalg import solve_discrete_lyapunov

# Define the matrix A and the noise covariance matrix
A = np.array([[0.7, 0.1], [0.3, 0.5]])
Sigma_eps = np.eye(2)

# Solve the Lyapunov equation
C_0 = solve_discrete_lyapunov(A, Sigma_eps)

print("Autocovariance matrix at lag 0:")
print(C_0)
```

The output of this code provides the following autocovariance matrix at lag $h = 0$:

$$C_{\boldsymbol{\mathcal{Y}}}(0) = \begin{bmatrix} 2.25256769 & 0.92203548 \\ 0.92203548 & 1.97245565 \end{bmatrix}.$$

The diagonal elements $C_{\mathcal{Y}^1}(0) = 2.25$ and $C_{\mathcal{Y}^2}(0) = 1.97$ represent the variances of the components $\mathcal{Y}^1$ and $\mathcal{Y}^2$, respectively. The off-diagonal element $C_{\mathcal{Y}^1\mathcal{Y}^2}(0) = 0.92203548$ reflects the covariance between the two components at lag $h = 0$. The cross-correlation between $\mathcal{Y}^1$ and $\mathcal{Y}^2$ at lag $h = 0$ can be computed by normalizing the covariance by the standard deviations of each component:

$$R_{\mathcal{Y}^1\mathcal{Y}^2}(0) = \frac{C_{\mathcal{Y}^1\mathcal{Y}^2}(0)}{\sigma_{\mathcal{Y}^1}\sigma_{\mathcal{Y}^2}} = \frac{0.92}{\sqrt{2.25} \cdot \sqrt{1.97}} \approx 0.43. \tag{6}$$

This cross-correlation value of approximately 0.43 indicates a moderate positive relationship between the two components at lag 0.

We will now generate a sample path of this VAR(1) process with $C = 2$ components, compute the sample mean vector, the sample covariance matrix, and visualize the relationships using scatter plots and cross-correlation plots. (The Python code can be found online in GitHub.)

1. **Simulating VAR(1) Process:** In this first code cell, we generate a multivariate time series using the VAR(1) model with two components. This step simulates two variables with a predefined autoregressive coefficient matrix A and identity covariance matrix for the noise $\Sigma_\epsilon = \mathbb{I}_2$.

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.api import VAR

```

```python
# Simulate VAR(1) time series
np.random.seed(42)
L = 1000  # length of time series
C = 2     # number of variables

# Autoregressive coefficient matrix for VAR(1) process
A = np.array([[0.7, 0.1],
              [0.3, 0.5]])

# Innovation (noise) covariance (identity matrix)
cov = np.eye(C)

# Generate random noise
noise = np.random.multivariate_normal(np.zeros(C), cov, L)

# Initialize the time-series data
data = np.zeros((L, C))

# Simulate the VAR(1) process
for t in range(1, L):
    data[t] = A @ data[t-1] + noise[t]

# Create DataFrame
df = pd.DataFrame(data, columns=['Y1', 'Y2'])

# Display the first few rows
df.head()
```

In this cell, we simulate a VAR(1) process for $L = 1000$ time steps and $C = 2$ components, stored in the DataFrame `df`. A sample path of this multivariate process is visualized in Fig. 7.

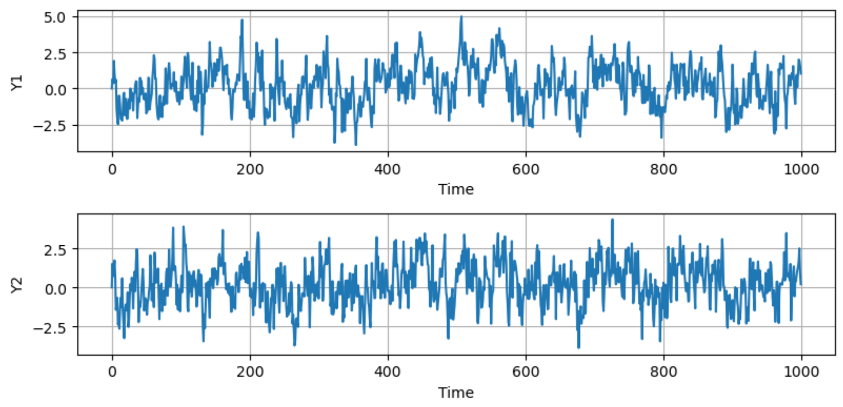

Figure 7: Sample path of a VAR(1) process with $C = 2$ and $L = 1000$.

2. **Scatter Plots:** Scatter plots are used to visualize the pairwise relationships

between the two variables. The correlation coefficients are computed and displayed in the titles of the plots.

```python
# Scatter plots between Y1 and Y2
plt.figure(figsize=(5, 4))

# Compute the Pearson correlation coefficients
corr_Y1_Y2 = df['Y1'].corr(df['Y2'])

# Y1 vs Y2
plt.scatter(df['Y1'], df['Y2'], alpha=0.5)
plt.title(f'Y1 vs Y2\nCorr: {corr_Y1_Y2:.2f}')
plt.xlabel('Y1')
plt.ylabel('Y2')

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

The output of this cell is shown in Fig. 8, which illustrates the positive correlation between $\mathcal{Y}^1$ and $\mathcal{Y}^2$. Notice that the Pearson correlation coefficient, $\rho = 0.43$, is exactly the same as the theoretical value obtained in (6).

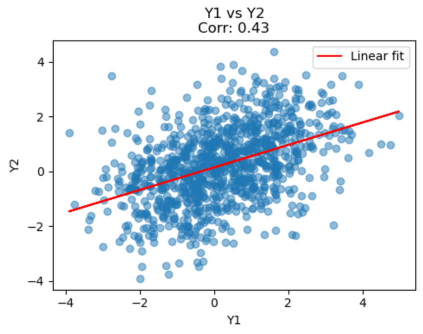

Figure 8: Scatter plot showing the correlation between $\mathcal{Y}^1$ and $\mathcal{Y}^2$.

3. **Cross-Correlation Functions (CCF):** Finally, we compute and plot the cross-correlation functions for the two variables across different time lags using a matrix plot.

```python
from scipy.stats import zscore

# Standardizing the series (z-scores)
df['Y1_z'] = zscore(df['Y1'])
df['Y2_z'] = zscore(df['Y2'])

# Create subplots for cross-correlation
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(8, 8))
max_lags = 20  # Maximum number of lags
```

```python

# Plot cross-correlations
axes[0, 0].xcorr(df['Y1_z'], df['Y1_z'], maxlags=max_lags)
axes[0, 0].set_title(f'Autocorrelation: Y1')

axes[1, 1].xcorr(df['Y2_z'], df['Y2_z'], maxlags=max_lags)
axes[1, 1].set_title(f'Autocorrelation: Y2')

axes[0, 1].xcorr(df['Y1_z'], df['Y2_z'], maxlags=max_lags)
axes[0, 1].set_title(f'Cross-correlation: Y1 & Y2')

axes[1, 0].xcorr(df['Y2_z'], df['Y1_z'], maxlags=max_lags)
axes[1, 0].set_title(f'Cross-correlation: Y2 & Y1')

for ax in axes.flat:
    ax.set_xlabel('Lag')
    ax.set_ylabel('Cross-correlation')

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

This cell produces the matrix of subplots in Fig. 9, where the diagonal elements are the autocorrelation functions of each component and the off-diagonal elements are the cross-correlation functions between the two components for different time lags.

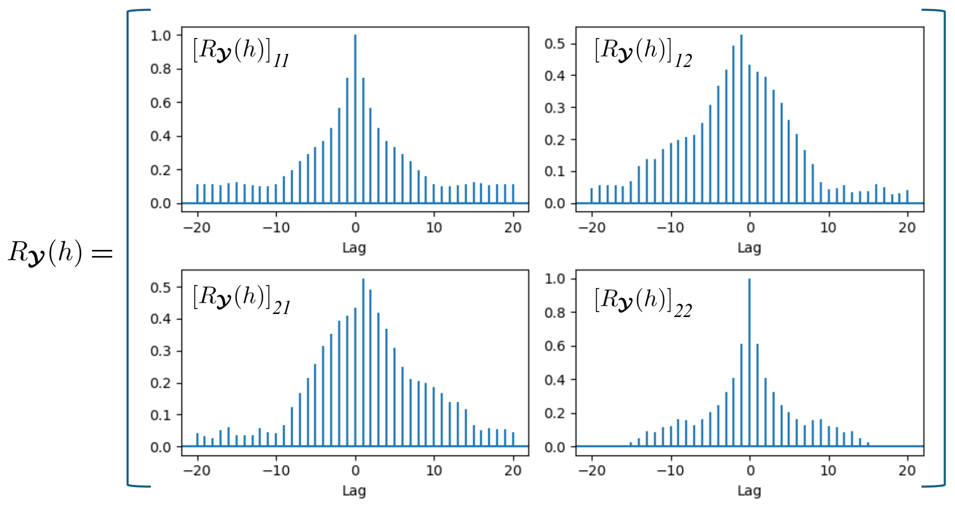

Figure 9: Matrix of subplots representing the entries of the cross-correlation matrix $R_{\mathcal{Y}}(h)$. We assume stationarity; hence, each entry $[R_{\mathcal{Y}}(h)]_{ij}$ is a function of the lag $h \in \mathbb{Z}$, for all $i, j \in \{1, 2\}$.

3.3 Multivariate Markov Processes

A **multivariate Markov process** is a generalization of the classical Markov process to systems involving multiple interrelated stochastic variables evolving over time. In this framework, the future evolution of the system depends only on the current values of the variables, not on the past history. Let $\boldsymbol{\mathcal{Y}} = \{\mathbf{Y}_k \colon k \in \mathbb{N}\}$ be a vector-valued stochastic process, where each $\mathbf{Y}_k \in \mathbb{R}^C$ is a C-dimensional vector, with C being the number of interrelated components of the multivariate process. The process $\mathbf{Y}_k$ is said to be a multivariate Markov process if it satisfies the **Markov property**, defined as:

$$\mathbb{P}(\mathbf{Y}_{k+1} \leq \mathbf{y}_{k+1} \mid \boldsymbol{\mathcal{F}}_k) = \mathbb{P}(\mathbf{Y}_{k+1} \leq \mathbf{y}_{k+1} \mid \mathbf{Y}_k = \mathbf{y}_k) \text{ for all } k \in \mathbb{N},$$

where the vector inequality $\mathbf{Y}_{k+1} \leq \mathbf{y}_{k+1}$ is understood component-wise. This means that the conditional distribution of future observations $\mathbf{Y}_{k+1}$ depends only on the current observations $\mathbf{Y}_k$ and not on the entire past history $\mathbf{Y}_{k-1}, \mathbf{Y}_{k-2}, \ldots, \mathbf{Y}_1$.

A fundamental property of vector-valued Markov processes is their capacity to encapsulate higher-order dependencies. Specifically, any *scalar-valued Markov process of order* m—where the future state depends on the previous m states—can be reformulated as an equivalent *first-order vector-valued Markov process*. This transformation preserves the essential dynamics of the original process while simplifying its mathematical representation. To illustrate this concept, let us consider a simple example.

Example 13: Transformation of AR(2) into a VAR(1) Process

Consider a (scalar-valued) AR(2) process, defined as:

$$Y_{k+1} = \phi_0 Y_k + \phi_1 Y_{k-1} + \epsilon_k, \quad \text{with } \epsilon_k \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2), Y_0 = y_0 \text{ and } Y_{-1} = y_{-1}. \tag{7}$$

This is a second-order Markov process, since the distribution of the future value Y_{k+1} depends on the values of the previous two observations, Y_k and Y_{k-1}; hence, the order parameter is $m = 2$.
We can transform this AR(2) process into a vector-valued first-order Markov process by defining a vector-valued process $\boldsymbol{\mathcal{X}} = \{\mathbf{X}_k \colon k \in \mathbb{N}\}$, with $\mathbf{X}_k = [X_{k,1}, X_{k,2}]^\intercal$. This transformation can be done by defining a VAR(1) process, $\mathbf{X}_{k+1} = \mathbf{A}\mathbf{X}_k + \boldsymbol{\epsilon}_k$ with:

$$\mathbf{A} = \begin{bmatrix} \phi_0 & \phi_1 \\ 1 & 0 \end{bmatrix}, \boldsymbol{\epsilon}_k = \begin{bmatrix} \epsilon_k \\ 0 \end{bmatrix} \text{ and } \mathbf{X}_k = \begin{bmatrix} y_0 & y_{-1} \end{bmatrix}^\intercal.$$

The VAR(1) process is a first-order (multivariate) Markov process, since the distribution of $\mathbf{X}_{k+1}$ depends solely on the value of $\mathbf{X}_k$.
We now demonstrate the equivalence between AR(2) and VAR(1). To achieve

this, we expand the VAR(1) recursion into its scalar components, as::

$$X_{k+1,1} = \phi_0 X_{k,1} + \phi_1 X_{k,2} + \epsilon_k,$$
$$X_{k+1,2} = X_{k,1}.$$

Substituting the second equation into the first one, one obtains the following recursion:

$$X_{k+1,1} = \phi_0 X_{k,1} + \phi_1 X_{k-1,1} + \epsilon_k$$

Notice that we can recover the AR(2) recursion in (7) by making the change of variables $X_{k,1} = Y_k$. Thus, we have shown that the scalar AR(2) process can be equivalently represented as a VAR(1) process. This transformation is helpful in analyzing and simulating the AR(2) dynamics within a first-order Markov framework.

In this example, we we have shown that the scalar AR(2) process can be equivalently represented as a VAR(1) process. More generally, any higher-order scalar process can be transformed into an equivalent first-order vector-valued Markov process by appropriately choosing the dimension of the vectors. By recasting a higher-order dependence structure in terms of a first-order vector representation, we capture the same dynamics within a multivariate framework, providing a more flexible and comprehensive set of tools for studying interactions, forecasting, and understanding the behavior of complex time-dependent systems.

3.4 Causality in Multivariate Time Series

In multivariate time series analysis, the concept of **causality** is fundamental in understanding the dynamic interactions between multiple components. Specifically, we aim to determine whether the past values of certain components can help predict the future values of other components. This is particularly important in systems where the components exhibit complex interdependencies over time.

Establishing causality with conventional statistical methods poses significant challenges. Statistical methods typically excel at identifying correlations between variables but fall short in discerning the direction or nature of influence. Correlation merely captures co-movement between variables, without revealing whether one variable drives changes in another. To establish a causal relationship, we must demonstrate that changes in one variable lead to changes in another. The gold standard for such an analysis is the **randomized controlled trial (RCT)**, where an intervention is deliberately introduced to isolate the causal effect of one variable on another. However, RCTs are often costly, difficult to implement, or even infeasible in certain domains, both practically and ethically.

3.4.1 Granger Causality

When an RCT is not a feasible option, **Granger causality** offers a tractable alternative framework for assessing *predictive relationships* between time series variables. The central idea behind Granger causality is *predictive relevance*: if the past values of one time series $\mathcal{Y}^i$ enhance the prediction of another time series $\mathcal{Y}^j$, beyond what can be achieved using only the past values of $\mathcal{Y}^j$ itself, then $\mathcal{Y}^i$ is said to **Granger-cause** $\mathcal{Y}^j$.

To formalize the concept of causality, consider a multivariate time series $\boldsymbol{\mathcal{Y}}$ as a collection of C scalar-valued time series of equal length L, denoted as $\{\mathcal{Y}^1, \mathcal{Y}^2, \ldots, \mathcal{Y}^C\}$. Let Y_k^i represent the k-th sample of the time series $\mathcal{Y}^i$, and define the **information set** $\mathcal{F}_k^i = \{Y_k^i = y_k^i, Y_{k-1}^i = y_{k-1}^i, \ldots\}$. The information set indicates what values the random variables in the component $\mathcal{Y}^i$ have taken up to time k. Granger causality between two stochastic processes $\mathcal{Y}^i$ and $\mathcal{Y}^j$ is formally defined as follows: $\mathcal{Y}^i$ Granger-causes $\mathcal{Y}^j$ if the conditional probability distribution of Y_{k+1}^j, given the information sets $\mathcal{F}_k^i$ and $\mathcal{F}_k^j$, differs from the conditional distribution of Y_{k+1}^j given only $\mathcal{F}_k^j$. In mathematical terms, this implies that there exists some $y \in \mathbb{R}$ such that:

$$\mathbb{P}\left(Y_{k+1}^j \leq y \mid \mathcal{F}_k^j, \mathcal{F}_k^i\right) \neq \mathbb{P}\left(Y_{k+1}^j \leq y \mid \mathcal{F}_k^j\right).$$

In simpler terms, $\mathcal{Y}^i$ *Granger-causes* $\mathcal{Y}^j$ if incorporating the information set $\mathcal{F}_k^i$ (i.e., the past values of $\mathcal{Y}^i$) changes the conditional distribution of future values of $\mathcal{Y}^j$.

It is crucial to note that Granger causality does not imply *true* causality in the scientific sense. Rather, it reflects the predictive power of one time series over another, indicating temporal precedence and predictability without accounting for possible confounding variables or underlying mechanisms that may affect both series. Establishing true causality often requires experimental intervention or deep insights into the system's underlying dynamics, where direct manipulation of variables is feasible. Granger causality, in contrast, is a statistical construct that captures the structure of the data without necessarily implying a direct cause-and-effect relationship.

3.4.2 Detecting Granger Causality

Detecting Granger causality is crucial for understanding the temporal dependencies between variables in multivariate time series, particularly when experimental intervention is not possible. In the context of wide-sense stationary multivariate time series, Granger causality is typically detected using **Vector Autoregressive (VAR)** models. A VAR model captures the dependencies of each variable on both its own past values and the past values of other components in the multivariate system. Formally, a VAR(m) model for a multivariate time series is represented by the following recursion:

$$\mathbf{Y}_{k+1} = A_0 \mathbf{Y}_k + A_1 \mathbf{Y}_{k-1} + \cdots + A_{m-1} \mathbf{Y}_{k-m+1} + \boldsymbol{\epsilon}_k,$$

where $\boldsymbol{\epsilon}_k$ is a C-dimensional white noise with $\boldsymbol{\epsilon}_k \overset{\text{iid}}{\sim} \mathcal{N}(\mathbf{0}, \Sigma_\epsilon)$, m is the *order*[10] of the VAR process, and $A_0, A_1, \ldots, A_{m-1}$ are $C \times C$ coefficient matrices, where C is the number of variables in the system.

To determine whether Granger causality is present, a hypothesis test is performed, where the null hypothesis H_0 posits that the inclusion of past values from Y^i does not improve the prediction of Y^j beyond what is already predicted by the past values of Y^j alone. Formally, this can be stated as:

$$H_0 : \alpha^j_{hi} = 0 \text{ for all } h = 0, 1, \ldots, m-1 \quad \text{(in the unrestricted model).}$$

In testing whether the time series component $\mathcal{Y}^i$ Granger-causes $\mathcal{Y}^j$, two models are compared:

1. The **unrestricted model**, which predicts the values of Y^j_{k+1} using a linear combination of all available information, represented by the union of information sets $\mathcal{F}^1_k \cup \mathcal{F}^2_k \cup \cdots \cup \mathcal{F}^C_k$, i.e., the information set that includes the past values of all components up to time k:

$$Y^j_{k+1} = \alpha_0 + \sum_{h=0}^{m-1} \sum_{c=1}^{C} \alpha^j_{hc} Y^c_{k-h} + \epsilon^j_k.$$

2. The **restricted model**, which excludes the information available in $\mathcal{F}^i_k$ from the model:

$$Y^j_{k+1} = \alpha_0 + \sum_{h=0}^{m-1} \sum_{c \neq i} \alpha^j_{hc} Y^c_{k-h} + \epsilon^j_k.$$

To test for Granger causality, we use a standard F-test[11], which compares the prediction performance of the restricted and unrestricted models using their residual sum of squares (RSS), denoted by RSS_{res} and $\text{RSS}_{\text{unres}}$, respectively. The F-test evaluates whether including past values from other components significantly reduces the RSS of the restricted model. Mathematically, the F-statistic is defined as:

$$F = \frac{(\text{RSS}_{\text{res}} - \text{RSS}_{\text{unres}})/m}{\text{RSS}_{\text{unres}}/(L - m - mC)}.$$

A significant reduction in the prediction error of the unrestricted model suggests that the additional predictors meaningfully contribute to predicting the dependent

[10]The order m represents the number of past time steps that influence the present in the VAR process.

[11]For a detailed explanation of F-tests, see [4].

variable, leading to the rejection of the null hypothesis. If the F-statistic is significant, we reject H_0, concluding that $\mathcal{Y}^i$ Granger-causes $\mathcal{Y}^j$. In technical terms, under the null hypothesis H_0, the F-statistic follows an $F(m, L-m-mC)$ distribution. The p-value is computed by comparing the observed value of the F-statistic to this distribution. If the p-value is sufficiently small (typically less than 0.05), we reject the null hypothesis, concluding that the additional predictors significantly enhance the model, and thus, $\mathcal{Y}^i$ Granger-causes $\mathcal{Y}^j$.

It is important to note that rejecting H_0 indicates that the past values of $\mathcal{Y}^i$ provide significant additional predictive information for $\mathcal{Y}^j$, beyond what can be predicted using the past values of $\mathcal{Y}^j$ alone. However, this result does not imply that $\mathcal{Y}^i$ directly causes $\mathcal{Y}^j$ in a causal or mechanistic sense. Granger causality identifies a temporal predictive relationship, but it is possible that other unobserved variables or confounding factors influence both series. Therefore, Granger causality should be understood as a statistical association reflecting predictive power, rather than definitive proof of a direct cause-and-effect relationship.

Python Lab: Granger Causality

This example illustrates how to test Granger causality between two time series: $\mathcal{Y}^1$ and $\mathcal{Y}^2$. We explain the steps of our analysis below:

1. **Multivariate Model:** We generate two synthetic time series using the following VAR(1) model:

$$Y_k^1 = 0.5Y_{k-1}^1 + \epsilon_k^1, \quad \epsilon_k^1 \overset{\text{iid}}{\sim} \mathcal{N}(0,1) \tag{8}$$

$$Y_k^2 = 0.3Y_{k-1}^2 + 0.7Y_{k-1}^1 + \epsilon_k^2, \quad \epsilon_k^2 \overset{\text{iid}}{\sim} \mathcal{N}(0,1). \tag{9}$$

   From our knowledge of the model, we can observe how $\mathcal{Y}^1$ evolves independently of $\mathcal{Y}^2$, while $\mathcal{Y}^2$ depends on the past values of $\mathcal{Y}^1$; therefore, $\mathcal{Y}^1$ Granger-causes $\mathcal{Y}^2$, establishing a unidirectional predictive relationship between the two series. In practical applications, we typically lack prior knowledge of the underlying model structure. Granger causality tests are used to infer predictive relationships using solely time series (without prior knowledge about the model). In other words, we only have access to one sample path of the multivariate time series to test for Granger causality. In the following steps, we will demonstrate how to establish Granger causality within the synthetic model solely by analyzing the time-series data, without relying on prior knowledge of the true underlying model that generated the data.

2. **Simulated Example in Python:** The following Python script performs a Granger causality test using the method `grangercausalitytests` in the `statsmodels` library. In the last line of code, the `maxlag` variable is set to

be equal to 1. This indicates to the method to use a lag equal to 1 in the regressions. Although the `grangercausalitytests` method shows the result of several tests, the key test statistic to focus on is **RSS based F-test**, which compares the RSS between models with and without the lagged values of the potential Granger-causing variable. If the F-statistic of this test is large and the p-value is small, it suggests that the inclusion of these lagged values leads to a significant improvement in the predictive power of the model. This code can be found in GitHub.

```python
import numpy as np
import pandas as pd
from statsmodels.tsa.stattools import grangercausalitytests

# Simulating time-series data
np.random.seed(42)
n = 100
epsilon1 = np.random.normal(0, 1, n)
epsilon2 = np.random.normal(0, 1, n)

Y1 = np.zeros(n)
Y2 = np.zeros(n)

# Generating Y1 and Y2 based on the equations
for k in range(1, n):
    Y1[k] = 0.5 * Y1[k-1] + epsilon1[k]
    Y2[k] = 0.3 * Y2[k-1] + 0.7 * Y1[k-1] + epsilon2[k]

# Combine into a DataFrame
data = pd.DataFrame({'Y1': Y1, 'Y2': Y2})

# Performing Granger causality test
grangercausalitytests(data[['Y2', 'Y1']], maxlag=1)
```

3. **Interpretation of Results:** For lag 1, the **RSS-based test** produces an F-statistic of **60.08** with a p-value of $\mathbf{9.5 \times 10^{-12}}$, providing compelling evidence that the past values of the potential Granger-causing variable at lag 1 significantly enhance the predictive power of the model. This result aligns with the structure of the underlying VAR(1) process, where the dependent time series is directly influenced by its own past values as well as those of the Granger-causing variable.

In this section, we are only scratching the surface of the vast field of causality, which remains an area of active research across multiple disciplines. Causal inference involves understanding the true cause-and-effect relationships in data, particularly in settings where controlled experiments are not feasible. Modern approaches to causal inference often revolve around the **potential outcomes framework** [2], which provides a systematic way to evaluate treatment effects under various assumptions. An authoritative reference on this topic is [5].

3.5 Information Theory for Time Series

Complement...

- **Granger Causality/Transfer Entropy**: Used to infer causal relationships between time-series.
- **Entropy**: Quantifies uncertainty or complexity in time-series.
- **Mutual Information**: Measures dependency between time-series or between different time points.
- **Transfer Entropy**: Captures the directed information flow between time-series.
- **KL Divergence**: Measures the difference between distributions in time-series data.
- **Conditional Entropy**: Measures remaining uncertainty in one series given another.
- **Renyi Entropy, Approximate Entropy, and Sample Entropy**: Specialized measures of complexity and regularity.
- **Predictive Information**: Measures the predictability of a time-series.

Appendix. Use of the Python Software

The Python software for general-purpose computing and data analysis has become a popular choice for time series analysis, statistical computing, and the development of new algorithms. Python is available as free and open-source software under the terms of the Python Software Foundation License. It runs on all major operating systems including Windows, macOS, and Linux. The main website for the Python project is `https://www.python.org`.

The Python environment consists of a base system, which includes the Python language interpreter, and a large ecosystem of user-contributed libraries. The base system provides the core functionality of Python, including built-in types and functions. Additionally, there are thousands of libraries available for download that extend the functionality of Python. These libraries cover various fields, such as numerical computing, data visualization, and machine learning.

For time series modeling and forecasting, some of the most useful Python libraries include:

- `NumPy`: A library for numerical computing, providing support for arrays, matrices, and a large collection of mathematical functions. `https://numpy.org/`
- `Pandas`: A powerful data manipulation and analysis library that includes support for time-series data. `https://pandas.pydata.org/`
- `Statsmodels`: A library that provides classes and functions for the estimation of many different statistical models, including time series models like ARIMA. `https://www.statsmodels.org/`
- `SciPy`: A library used for scientific computing, providing tools for optimization, integration, interpolation, and statistics. `https://scipy.org/`
- `Matplotlib`: A plotting library used for creating static, interactive, and animated visualizations in Python. `https://matplotlib.org/`

Running Python Code in the Cloud (Google Colab)

Google Colab is a free cloud service that allows you to write and execute Python code in a web-based Jupyter notebook environment. One of the main benefits of Colab is that it requires no setup, and it provides free access to GPUs, making it an excellent option for heavy computations such as machine learning and time series forecasting.

To start using Google Colab:

1. Navigate to `https://colab.research.google.com`.

2. Sign in with your Google account (if not already signed in).
3. Create a new notebook by clicking `File > New Notebook`.
4. In the notebook, you can write Python code directly in cells and run it by clicking the "Run" button or using the keyboard shortcut `Shift + Enter`.

To install and import libraries in Colab, use the following commands. For example, to install `pandas`, run:

```
!pip install pandas
```

Once installed, the library can be imported as usual:

```
import pandas as pd
import numpy as np
```

Colab also allows you to upload files directly into the environment or access them from Google Drive, making it highly flexible for data science tasks.

Running Python Code Locally on Your Computer

To run Python locally, you first need to install the Python interpreter and a suitable Integrated Development Environment (IDE), such as `Jupyter Notebook`, `VSCode`, or `PyCharm`.

1. Installing Python

The easiest way to install Python and manage packages is by using the `Anaconda` distribution, which comes with Python, `pip`, and most of the essential libraries for data analysis and scientific computing. To install Anaconda:

1. Go to `https://www.anaconda.com/products/individual`.
2. Download the installer for your operating system (Windows, macOS, or Linux).
3. Follow the installation instructions on the Anaconda website.

After installation, you can launch the `Anaconda Navigator` to open Jupyter Notebooks or use `Anaconda Prompt` to run Python code from the command line.

2. Installing Libraries

Once Python is installed, you can install additional libraries using `pip`, Python's package manager. For example, to install `pandas` and `statsmodels`, open a terminal or command prompt and run the following commands:

```
pip install pandas
pip install statsmodels
```

To verify that Python and the necessary libraries have been installed correctly, you can start a Python session and import the libraries:

```python
import pandas as pd
import statsmodels.api as sm
```

3. Running Python Code in a Jupyter Notebook

Once Anaconda is installed, you can launch a `Jupyter Notebook` by opening the `Anaconda Navigator` and selecting `Jupyter Notebook`. This will open a web browser where you can create new notebooks and execute Python code in cells. Here's an example of loading a dataset in a notebook:

```python
import pandas as pd

# Load data from CSV
data = pd.read_csv('yourfile.csv')

# Display the first few rows
print(data.head())
```

For statistical analysis, the `statsmodels` library provides various time series models, including ARIMA, and functions to evaluate model performance. To fit an ARIMA model to a time series, you can use the following code:

```python
from statsmodels.tsa.arima.model import ARIMA

# Fit an ARIMA model
model = ARIMA(data, order=(1,1,1))
fit_model = model.fit()

# Output model summary
print(fit_model.summary())
```

Documentation and Learning Resources

The documentation for Python and its libraries is extensive and available online. The documentation for major libraries such as `NumPy`, `Pandas`, and `Statsmodels` is also available on their respective websites.

Appendix: Spectral Analysis of Stochastic Processes

TBD: Fourier and Wavelets.

Exercises

1. (4 points) Consider a fair coin toss game where you start with \$10. Every time you flip the coin:

   - If it lands heads, you win \$1.
   - If it lands tails, you lose \$1.

   Let Y_k represent the amount of money you have after k coin flips, with $Y_0 = 10$. For each coin flip, the outcome is independent of previous flips. Answer the following questions assuming that you can get into debt (i.e., Y_k can be negative):

   (a) Find the expected value of Y_k, i.e., the amount of money you will have after k coin flips.

   (b) Compute the variance of Y_k after k coin flips.

   (c) Program a piece of Python code to generate and plot 10 sample paths of length 100 of the random process. Include a shaded area indicating the 95% confidence interval of the process (assume that the distribution of Y_k is approximately normal).

2. (3 points) Modify the proof in Example 5 to consider the correlation between Y_k and Y_{k+h} (with a positive lag h, rather than a negative lag). Follow these steps:

   (a) Start by expressing Y_{k+h} as a function of Y_k, ϕ, and the noise terms ϵ at different time steps.

   (b) Compute the expectation $\mathbb{E}[Y_k Y_{k+h}]$ and expand the equation into a sum of expectations.

   (c) Apply the statistical properties of the noise terms (i.e., zero mean and lack of correlation) to simplify the expression.

   (d) Compare your expression for the autocovariance to the one obtained in Example 5.

3. (5 points) Consider a random process Y_k defined by the recursion:

$$Y_k = (1 - \phi^k) + \theta_0 \epsilon_k + \theta_1 \epsilon_{k-1}, \quad \text{with } \epsilon_k \overset{\text{iid}}{\sim} \mathcal{N}(0,1) \text{ and } |\phi| < 1. \tag{10}$$

   Answer the following questions:

   (a) Compute the theoretical mean $\mathbb{E}[Y_k]$ and its limit for $k \to \infty$.

   (b) Compute the theoretical variance $\text{Var}[Y_k]$.

   (c) Compute the theoretical autocovariance $\text{Cov}[Y_k, Y_{k-h}]$.

(d) Is the random process strong-sense stationary? Explain your answer.

(e) Is it strong-sense stationary in the limit $k \to \infty$.

(f) Program a piece of Python code to generate and plot 10 sample paths of length 100 of the random process. Include a shaded area indicating the 95% confidence interval of the process, assuming that the process is normally distributed. Use the value $\theta_1 = 0.5$ in your simulations.

4. (3 points) Let $\alpha \sim \mathcal{N}(0,1)$ and $\beta \sim \mathcal{N}(0,2)$ be two independent random variables. Consider the random process $\mathcal{Y}$ defined by the recursion:

$$Y_k = \alpha + \beta k + k^2 + \epsilon_k, \quad \text{with } Y_0 = 0 \text{ and } \epsilon_k \overset{\text{iid}}{\sim} \mathcal{N}(0,1).$$

Answer the following questions:

(a) Compute the theoretical mean $\mathbb{E}[Y_k]$.

(b) Compute the theoretical variance $\mathrm{Var}[Y_k]$.

(c) Compute the theoretical autocovariance $\mathrm{Cov}[Y_k, Y_{k-h}]$.

(d) Is the random process wide-sense stationary? Explain your answer.

5. (3 points) Consider the process defined by the recursion:

$$Y_{k+1} = \phi_0 Y_k + \phi_1 Y_{k-1} + \phi_2 Y_{k-2} + \epsilon_k,$$

where $\epsilon_k \overset{\text{iid}}{\sim} \mathcal{N}(0,\sigma^2)$ is a white noise. Answer the questions below:

(a) What is the order of this higher-order Markov process? Justify your answer.

(b) What is the conditional distribution of Y_{k+1} given all the previous observations?

(c) Rewrite the process as a vector-valued first-order Markov process. Define the AR(3) process as a vector-valued recursion using a transition matrix and provide an explicit expression for the covariance matrix of the noise vector.

6. (3 points) Consider a Markov chain $\mathcal{X} = \{X_1, X_2, \ldots\}$ with state space $\mathcal{S} = (1,2,3)$ and transition matrix:

$$P = \begin{bmatrix} 0 & 0.8 & * \\ 0.1 & * & 0.3 \\ 0.2 & 0.8 & * \end{bmatrix}.$$

(a) Fill in the unknown entries, denoted by $*$, in the transition matrix.

(b) Find the probability that $X_2 = s_3$ given that $X_0 = s_1$.

(c) Given that $X_0 = s_1$, compute the distribution vector $\boldsymbol{\pi}_2$, representing the state probabilities at time step 2.

7. (3 points) What is the maximum number of non-zero entries in the transition matrix P of a higher-order Markov chain with states $\mathcal{S} = \{s_1, s_2, \ldots, s_H\}$ and memory m (i.e., order $m-1$? Your answer should be a function of H and m.

8. (3 points) Consider the random process $\mathcal{Y}$ defined by the recursion:

$$Y_k = f(k) + \epsilon_k, \quad \text{with } Y_0 = 0 \text{ and } \epsilon_k \overset{\text{iid}}{\sim} \mathcal{N}(0,1),$$

where $f(\cdot)$ is a generic function. Answer the following questions:

(a) Compute the theoretical mean $\mathbb{E}[Y_k]$. Does it depend on the particular form of $f(\cdot)$?

(b) Compute the theoretical variance $\mathrm{Var}[Y_k]$. Does it depend on the particular form of $f(\cdot)$?

(c) Compute the theoretical autocovariance $\mathrm{Cov}[Y_k, Y_{k-h}]$. Does it depend on the particular form of $f(\cdot)$?

(d) Under what assumption on $f(\cdot)$ is the random process wide-sense stationary? Explain your answer.

9. (6 points) Consider a two-dimensional VAR(1) process defined by the recursion:

$$\mathbf{Y}_k = A\mathbf{Y}_{k-1} + \boldsymbol{\epsilon}_k,$$

where:

$$\mathbf{Y}_k = \begin{bmatrix} Y_k^1 \\ Y_k^2 \end{bmatrix}, \quad A = \begin{bmatrix} \phi_{11} & 0 \\ \phi_{21} & \phi_{22} \end{bmatrix}, \quad \boldsymbol{\epsilon}_k = \begin{bmatrix} \epsilon_k^1 \\ \epsilon_k^2 \end{bmatrix} \sim \mathcal{N}(\mathbf{0}_2, \mathbb{I}_2),$$

For stability reasons, we also assume $|\phi_{11}| < 1$ and $|\phi_{22}| < 1$. Answer the following questions:

(a) Derive an expression for the autocorrelation function for $\mathcal{Y}^1$.

(b) What is the cross-correlation between $\mathcal{Y}^1$ and $\mathcal{Y}^2$ at lag 0 and lag 1?

(c) Is the conditional distribution $F_{\mathbf{Y}_k|\boldsymbol{\mathcal{Y}}_{\leq k-1}}$ a Gaussian distribution? If so, what are the mean and the covariance matrix of this conditional distribution?

(d) Is VAR(1) a Markov process? Justify your answer based on the definition of a vector-valued Markov process.

(e) If $\phi_{21} \neq 0$, does $\mathcal{Y}^1$ Granger-cause $\mathcal{Y}^2$? Does $\mathcal{Y}^2$ Granger-cause $\mathcal{Y}^1$?

(f) If $\phi_{21} = 0$, how would this impact Granger causality between the two processes?

10. (3 points) We can observe in Fig. 9, that the entries of the matrix plot of cross-correlations satisfy certain symmetries. Assuming that we have a WSS multivariate process, prove the following statements:

    (a) The subplots in the diagonal, representing the autocorrelation functions, are even functions, i.e, $C_{\mathcal{Y}^i}(h) = C_{\mathcal{Y}^i}(-h)$ for all $h \in \mathbb{N}$ and all $i \in [C]$.

    (b) The subplots in the off-diagonal, representing the cross-correlation functions, follow a lag-reversal symmetry, i.e., $\mathcal{R}_{\mathcal{Y}^i,\mathcal{Y}^j}(h) = \mathcal{R}_{\mathcal{Y}^j,\mathcal{Y}^i}(-h)$ for all $h \in \mathbb{N}$ when $i \neq j$.

11. (4 points) Below is a matrix representing the autocovariance matrix of a random process with two components: $\mathcal{Y}^1$ and $\mathcal{Y}^2$. Information about the autocorrelation functions and cross-correlation functions between the processes are shown for lag values less than 3 in the following table:

Autocorrelation: $R_{\mathcal{Y}^1}(h)$		Cross-correlation: $R_{\mathcal{Y}^1\mathcal{Y}^2}(h)$	
Lag	Value	Lag	Value
0	1.0	0	0.8
1	0.5	1	0.4
2	0.2	2	0.2
3	0.1	3	0.1
Cross-correlation: $R_{\mathcal{Y}^2\mathcal{Y}^1}(h)$		**Autocorrelation: $R_{\mathcal{Y}^2}(h)$**	
Lag	Value	Lag	Value
0	0.8	0	1.0
1	0.3	1	0.4
2	0.2	2	0.1
3	0.05	3	0.05

Assuming that the variance of $\mathcal{Y}^1$ and $\mathcal{Y}^2$ are 1 and 2, respectively, answer the following questions:

(a) Calculate the covariance between $\mathcal{Y}^1$ and $\mathcal{Y}^2$ at lag 0.

(b) Compute $R_{\mathcal{Y}^1}(-2)$.

(c) Calculate $R_{\mathcal{Y}^1\mathcal{Y}^2}(-1)$?

References

[1] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung. *Time series analysis: forecasting and control.* John Wiley & Sons, 2015.

[2] S. Cunningham. *Causal inference: The mixtape.* Yale university press, 2021.

[3] R. Durrett. *Probability: theory and examples*, volume 49. Cambridge university press, 2019.

[4] D. C. Montgomery, E. A. Peck, and G. G. Vining. *Introduction to linear regression analysis.* John Wiley & Sons, 2021.

[5] J. Pearl, M. Glymour, and N. P. Jewell. *Causal inference in statistics: A primer.* John Wiley & Sons, 2016.