FORECASTING WITH LARGE LANGUAGE MODELS

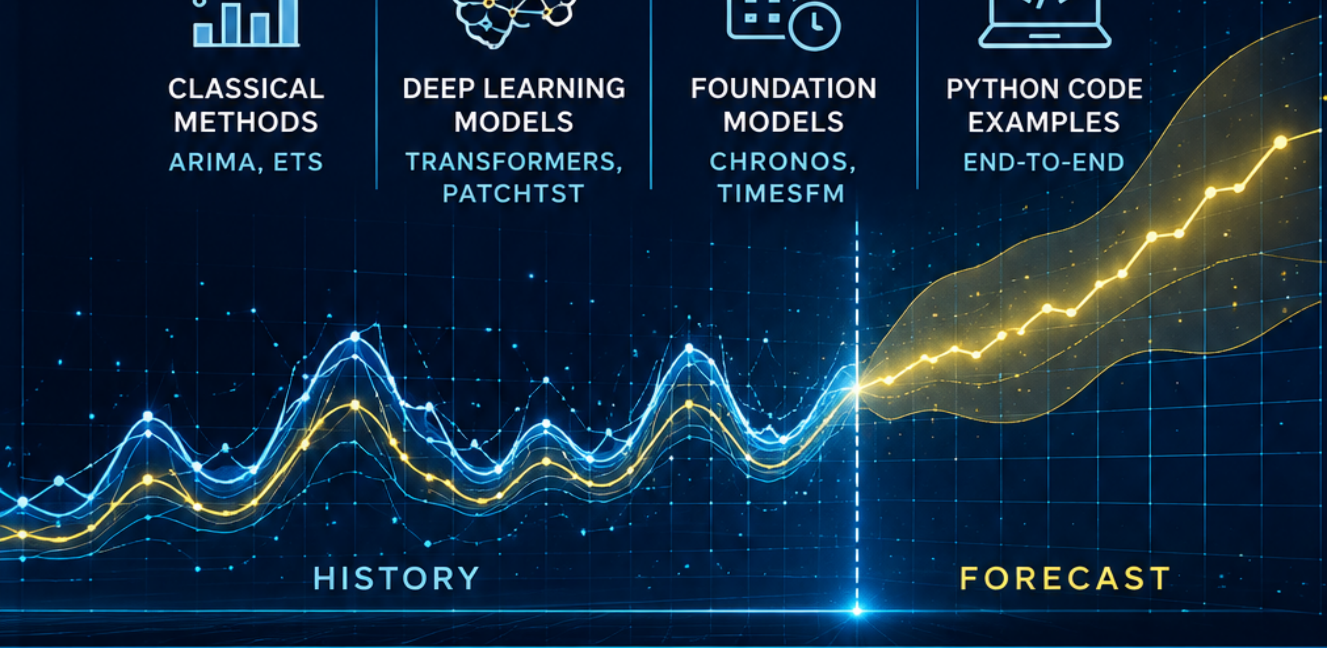
A COMPLETE GUIDE TO TIME SERIES PREDICTION FROM CLASSICAL METHODS TO LLMS

CLASSICAL METHODS
ARIMA, ETS

DEEP LEARNING MODELS
TRANSFORMERS, PATCHTST

FOUNDATION MODELS
CHRONOS, TIMESFM

PYTHON CODE EXAMPLES
END-TO-END



- PRACTICAL AND APPLICABLE
- CLEAR MATH MADE ACCESSIBLE
- HONEST INSIGHTS ON TRADE-OFFS AND FAILURE MODES

JESUS HOFFMAN

Forecasting with Large Language Models

A Complete Guide to Time Series Prediction from Classical Methods to LLMs

Steve Publications

This book is available at
https://leanpub.com/forecastingwithlargelanguagemodels

This version was published on 2026-09-01

Contents

A Complete Guide to Time Series Prediction from Classical Methods to LLMs

Time series forecasting has evolved through statistical models, machine learning, deep learning, and now foundation models built on large language model architectures. This book takes you from the fundamentals of time series analysis to the cutting edge of LLM-based forecasting. You will learn classical methods like ARIMA and exponential smoothing, modern neural architectures like Transformers and PatchTST, and pretrained foundation models like Chronos and TimesFM. Through complete Python code examples, mathematical intuition explained accessibly, and honest discussion of trade-offs and failure modes, this book equips you to design, evaluate, and deploy forecasting systems that actually work in production.

Chapter 1: What Is Time Series Forecasting?

The Value of Prediction

A retailer is deciding how much inventory to stock for the holiday season. Too little and customers leave empty-handed, taking their spending elsewhere. Too much and capital sits on shelves, eventually discounted or written off. A power grid operator must balance electricity generation against demand hour by hour. Shortages cause blackouts; excess capacity wastes money and carbon. A hospital administrator needs to anticipate patient admissions to staff wards appropriately. Understaffing endangers lives while overstaffing strains budgets already stretched thin.

These are not hypothetical scenarios. They are the daily reality of organizations that depend on time series forecasting. The numbers involved are substantial. Research from McKinsey found that improving forecast accuracy by just one percentage point could save a large retailer approximately 0.5 percent of its total cost of goods sold, which translates to tens or hundreds of millions of dollars annually for major chains [1]. In the energy sector, accurate load forecasting directly affects billions in generation and procurement decisions each year. Supply chain disruptions during and after the pandemic made these stakes even more visible when companies with better forecasting capabilities maintained delivery schedules while competitors struggled.

Forecasting also enables proactive rather than reactive decision-making. Instead of responding to stock-outs after they happen, a well-tuned demand forecast triggers reorder points before shelves go bare. Instead of discovering equipment failure through downtime, predictive maintenance models flag anomalies in sensor readings days or weeks ahead. The value extends beyond cost savings into customer satisfaction, operational resilience, and competitive advantage.

Yet forecasting is also notoriously difficult. Real-world time series are noisy, nonstationary, and influenced by factors that shift without warning.

A model that worked flawlessly last quarter may degrade after a product launch, a competitor's price change, or an entirely unforeseen event. These challenges have driven decades of methodological progress, from the statistical techniques introduced in the mid-twentieth century through modern deep learning and foundation models. Understanding this landscape is essential for choosing the right tool for your problem.

What Makes Time Series Different

Time series data has a property that separates it fundamentally from most other data you will encounter in machine learning: temporal ordering matters, and observations are not exchangeable. In standard supervised learning, shuffling training examples does not change what the model learns because each example is assumed to be independently drawn from the same distribution. Time series violates both assumptions.

Consider daily sales of a coffee shop. The value on any given day depends systematically on what happened yesterday, last week at this time, and whether it is summer or winter. Shuffling these observations destroys the very structure you want to model. Moreover, the underlying process generating the data can change over time. A new competitor opening across the street shifts demand permanently. A marketing campaign creates a temporary spike. These shifts mean that past data may not be representative of future conditions, a phenomenon called nonstationarity that complicates every forecasting method.

Temporal dependence also creates unique opportunities for leakage. If you accidentally let information from the future influence your model during training, you will get deceptively good results on validation data that completely fail in production. Splitting time series into training and test sets requires care: all training data must precede all test data in time, with no overlap in features derived from future values. This constraint eliminates many standard machine learning practices like random cross-validation folds.

Another distinctive feature of time series is the presence of multiple time scales. Retail sales exhibit daily patterns (higher on weekends), weekly cycles (payday effects), monthly rhythms (beginning-of-month spending), and annual seasonality (holidays). Energy demand shows intra-hour fluctuations, daily peaks, seasonal weather effects, and longer-term trends driven by efficiency

improvements or economic growth. A good forecasting method must handle all relevant time scales simultaneously without confusing short-term noise for long-term signals.

Core Terminology

Before we dive into methods, let us establish precise definitions for the terms used throughout this book. Clear terminology prevents confusion when comparing approaches and reading research papers.

A **time series** is a sequence of observations recorded at successive points in time, typically equally spaced. The spacing defines the **frequency**: hourly data has sixty-minute intervals, daily data has twenty-four-hour intervals, weekly data spans seven days, monthly data covers calendar months, and so on. Frequency choice often depends on the decision horizon: if you need to make decisions every day, daily forecasts are appropriate; if you manage inventory by the week, weekly aggregation may be more useful and less noisy.

The **forecast horizon** is how far into the future you want to predict. A one-step-ahead forecast predicts the next observation given all available history. Multi-step or multi-horizon forecasts predict several future values simultaneously. Longer horizons are inherently more uncertain because errors compound and unforeseen events become more likely. The relationship between horizon length and achievable accuracy shapes method selection: some approaches excel at short-term prediction while others handle long horizons better.

Univariate forecasting uses only the historical values of a single series to predict its future. If you are forecasting electricity consumption using only past consumption data, that is univariate. **Multivariate forecasting** incorporates multiple related series simultaneously. Forecasting electricity demand using both historical consumption and temperature forecasts is multivariate because it leverages cross-variable relationships.

Covariates (also called exogenous variables or features) are additional inputs available alongside the target series. They fall into categories based on when their values become known:

- **Past covariates** are only known up to the current time, like historical sales of related products.

- **Future-known covariates** have values that can be predicted or planned ahead of time, such as calendar features (day of week, holidays), promotional calendars, or weather forecasts.
- **Static covariates** do not change over time, like product category, store location, or equipment type.

The distinction between these categories matters for model design because some architectures explicitly handle each type differently. For example, the Temporal Fusion Transformer has dedicated processing pathways for static, past, and future-known inputs [2].

A **point forecast** is a single predicted value for a future time step: "next month's sales will be 10,000 units." Point forecasts are simple to communicate but incomplete because they hide uncertainty. A **probabilistic forecast** provides a full probability distribution over possible future values or at least quantiles defining prediction intervals: "next month's sales have a 90 percent chance of falling between 8,500 and 11,500 units." Probabilistic forecasts are essential for risk-aware decision-making in inventory management, capacity planning, and financial applications.

The Forecasting Landscape

Time series forecasting has evolved through several distinct eras, each introducing new capabilities while retaining useful ideas from predecessors. Understanding this progression helps you evaluate claims about new methods and avoid the trap of assuming that newer always means better.

Classical statistical methods dominated forecasting from the 1950s through the 1990s. ARIMA (Autoregressive Integrated Moving Average), introduced by Box and Jenkins in their seminal 1976 book [3], remains one of the most widely used approaches despite its age. Exponential smoothing methods, formalized by Holt and Winters for handling trend and seasonality, were later recast in a rigorous state-space framework by Hyndman and colleagues [4]. These methods are interpretable, computationally cheap, and surprisingly competitive on many real-world series. They require relatively little data and provide natural prediction intervals derived from their statistical assumptions.

Machine learning approaches entered forecasting as computational power grew and datasets expanded. Gradient boosting algorithms like XGBoost,

LightGBM, and CatBoost became popular for transforming time series into supervised learning problems by creating lag features and rolling statistics. The M5 forecasting competition, based on Walmart retail data with over 42,000 hierarchical product-store series, demonstrated the strength of this approach: the top submissions relied heavily on ensemble methods built around gradient boosted trees [5]. Tree-based models handle heterogeneous features well, require less manual tuning than deep learning, and provide feature importance scores for interpretability.

Deep learning brought sequence modeling capabilities to forecasting. Recurrent neural networks, particularly Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRU), were designed specifically for sequential data and could theoretically capture long-range temporal dependencies [6]. Probabilistic models like DeepAR from Amazon trained recurrent architectures on collections of related series to share statistical strength across products or locations [7]. More recently, Transformer architectures borrowed from natural language processing have been adapted for time series through models like the Temporal Fusion Transformer, PatchTST, and TimesNet [2, 8, 9]. These approaches excel with large datasets containing many related series but demand substantial computational resources and careful tuning.

Foundation models represent the current frontier. Inspired by the success of large language models in text, researchers have trained massive models on enormous time series corpora that can forecast unseen series without task-specific training. Chronos from AWS tokenizes numerical values like words and trains T5-based transformers to predict future tokens [10]. TimesFM from Google Research uses a patched-decoder architecture trained on 100 billion time points from Google Trends, Wikipedia pageviews, and public datasets [11]. These models offer zero-shot forecasting capabilities that rival or exceed specialized models trained from scratch on individual tasks.

The relationship between these eras is not purely competitive. Modern practice often combines approaches: using statistical methods for baseline comparisons, machine learning for feature engineering, deep learning for complex pattern recognition, and foundation models for rapid prototyping or data-scarce scenarios. The best forecasting systems are pragmatic about tool selection rather than ideologically committed to any single paradigm.

How to Use This Book

This book is organized as a progressive learning journey from fundamentals to advanced practice. You do not need prior expertise in time series analysis, but basic Python programming skills will help you follow the code examples. Here is how to navigate it:

Chapters 2 through 5 build the foundation. Chapter 2 teaches you how to examine time series data visually and statistically, identifying components like trend, seasonality, and noise. Chapter 3 covers data preparation: handling missing values, engineering features, scaling, and most importantly, proper temporal splitting that avoids leakage. Chapter 4 establishes rigorous evaluation practices including metrics, validation strategies, and baseline comparisons. Chapter 5 presents classical statistical methods with complete implementations in Python.

Chapters 6 through 8 bridge to modern approaches. Chapter 6 shows how machine learning models like gradient boosting can be applied to time series through careful feature design. Chapter 7 introduces recurrent neural networks for sequence modeling. Chapter 8 covers Transformer-based architectures that have reshaped the field.

Chapters 9 through 11 focus on large language models and foundation models, the book's central theme. Chapter 9 examines how text-trained LLMs can be prompted to forecast numbers, along with their limitations and failure modes. Chapter 10 surveys pretrained time series foundation models like Chronos and TimesFM. Chapter 11 covers adaptation techniques: fine-tuning, parameter-efficient methods, retrieval augmentation, and hybrid architectures.

Chapters 12 through 15 address advanced topics and production concerns. Chapter 12 tackles complex scenarios: multivariate forecasting, long horizons, hierarchical structures, and irregular data. Chapter 13 covers probabilistic forecasting and uncertainty quantification. Chapter 14 discusses deploying forecasting systems in production, including monitoring, drift detection, and retraining. Chapter 15 surveys current research directions and open problems.

Throughout the book, code examples use Python with widely adopted libraries: pandas for data manipulation, statsmodels for classical methods, scikit-learn for machine learning, PyTorch for deep learning, and specialized packages like darts, gluonts, and neuralforecast where appropriate. Examples

are designed to be complete and runnable rather than illustrative fragments.

Some readers may want to jump directly to LLM-based methods in Chapters 9 through 11. That is reasonable if you already have forecasting experience. However, the foundational chapters contain practical insights about evaluation, baselines, and data preparation that apply regardless of which models you choose. Even when using a foundation model as a black box, understanding what makes time series difficult helps you interpret results critically and recognize when something has gone wrong.

The book takes an evidence-based perspective. Where research consensus exists, I state it clearly. Where experts disagree or evidence is mixed, I present competing views with their supporting arguments. Where methods are promising but under-studied, I flag the uncertainty explicitly. The goal is not to advocate for any particular approach but to give you the knowledge and critical thinking skills to make informed decisions for your specific forecasting problems.

Chapter 2: Understanding Time Series Data

Components of a Time Series

Every time series can be decomposed into interpretable components that reveal its underlying structure. Recognizing these components guides model selection, feature engineering, and diagnostic analysis. The four standard components are trend, seasonality, cyclical patterns, and noise (also called residuals or irregular component).

Trend is the long-term directional movement in the data. A series with positive trend grows over time; negative trend indicates decline. Trends can be linear, exponential, or follow more complex shapes. Monthly revenue for a growing startup typically shows exponential trend as customer base compounds. Temperature data might exhibit linear trend due to climate change. Trend complicates forecasting because it implies the future differs systematically from the past, violating stationarity assumptions that many statistical methods require.

Seasonality refers to regular, repeating patterns at fixed intervals. Retail sales peak every December due to holidays. Electricity demand rises each afternoon as businesses operate and falls overnight. Web traffic shows daily seasonality (higher during business hours) and weekly seasonality (lower on weekends). Seasonality is predictable in principle because it recurs, but the exact amplitude and phase can shift over time. Some series exhibit multiple seasonalities simultaneously: hourly energy data has both twenty-four-hour daily patterns and seven-day weekly cycles.

Cyclical patterns resemble seasonality but operate at irregular intervals without fixed periods. Economic expansions and contractions produce business cycles lasting several years with variable timing. Unlike true seasonality, cyclical behavior cannot be captured by simply repeating a known pattern because the cycle length itself varies. Distinguishing cycles from trend can

be subtle: what looks like a long-term trend might actually be one phase of a multi-year cycle.

Noise (residuals) captures everything unexplained by the other components: measurement error, random fluctuations, and idiosyncratic events. After removing trend, seasonality, and identifiable cycles, well-specified decomposition leaves residuals that look approximately random with constant variance. Systematic patterns remaining in residuals indicate missed structure: perhaps an additional seasonal component was overlooked or a regime change was not accounted for.

Decomposition models combine these components additively or multiplicatively. An additive model assumes the series equals trend plus seasonality plus noise. A multiplicative model assumes the series equals trend times seasonality times noise, which is appropriate when seasonal amplitude scales with the level of the series (for example, holiday sales spikes are larger in absolute terms as total revenue grows). Real-world series often require hybrid approaches or transformations like logarithms to stabilize variance before additive decomposition works well.

Visualizing Time Series

Visualization is the first and most important step in understanding any time series. No statistical test replaces the insight gained from looking directly at the data. A systematic visual examination reveals issues that automated methods might miss: structural breaks, outliers, regime changes, and data quality problems.

The simplest visualization is a line plot of observations over time. This single chart shows trend direction, seasonal patterns, level shifts, and anomalies. When plotting, always label axes clearly with units and time scale. For long series, consider faceting by year or rolling windows to examine how patterns evolve.

Beyond basic line plots, several specialized visualizations extract specific insights:

Rolling statistics plots compute moving averages or standard deviations over sliding windows to reveal trend and volatility changes. A rolling mean smooths short-term fluctuations to expose underlying direction. A rolling standard deviation highlights periods of increased uncertainty. For example,

plotting a six-month rolling average of daily sales alongside raw data separates the long-term trajectory from day-to-day noise.

Seasonal subseries plots align observations by their position within each seasonal cycle. For monthly data with annual seasonality, this means plotting all January values together, all February values together, and so on, showing how each month behaves across years. This visualization exposes whether seasonality is stable or evolving: if January sales are consistently declining year over year while the overall trend rises, the seasonal pattern itself is changing.

Lag plots scatter each observation against its value at a previous time step (the lag). A clear linear relationship in a lag-1 plot indicates strong short-term autocorrelation: today's value depends heavily on yesterday's. Curved patterns suggest nonlinear dependence. Random scatter suggests the series resembles white noise with little predictable structure. Lag plots help diagnose whether autoregressive modeling is appropriate and at what lag orders.

Histograms and density plots of the data reveal distributional properties. Time series often exhibit skewness (asymmetric distributions), heavy tails (more extreme values than a normal distribution would predict), or multiple modes (suggesting mixture of different regimes). These characteristics affect model choice: methods assuming Gaussian errors may produce poor prediction intervals for heavy-tailed series.

Let us see these techniques in action with a complete Python example using real data from the Monash Time Series Forecasting Repository. We will load monthly international airline passenger data, a classic benchmark series that exhibits strong trend and annual seasonality.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.tsa.seasonal import STL
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.stattools import adfuller, kpss

# Load airline passengers data (classic benchmark series)
url = "
    https://raw.githubusercontent.com/jbrownlee/Datasets/master/airline-passengers.csv
    "
df = pd.read_csv(url, parse_dates=["Month"], index_col="Month")
df.columns = ["Passengers"]

```

```
print(f"Series length: {len(df)} observations")
print(f"Frequency: Monthly from {df.index.min()} to {df.index.max()}")
print(df.describe())

# Set plotting style
plt.style.use("seaborn-v0_8-whitegrid")
sns.set_context("talk")

# 1. Basic line plot
fig, axes = plt.subplots(2, 3, figsize=(15, 10))

df["Passengers"].plot(ax=axes[0, 0], title="Airline Passengers (Monthly)",
↪  color="steelblue")
axes[0, 0].set_ylabel("Thousands of passengers")

# 2. Rolling statistics (12-month window for annual smoothing)
rolling_mean = df["Passengers"].rolling(window=12, center=True).mean()
rolling_std = df["Passengers"].rolling(window=12, center=True).std()

axes[0, 1].plot(df.index, df["Passengers"], alpha=0.3, label="Raw",
↪  color="gray")
axes[0, 1].plot(df.index, rolling_mean, linewidth=2, label="12-month rolling
↪  mean", color="steelblue")
axes[0, 1].set_title("Rolling Mean and Standard Deviation")
axes[0, 1].axhspan(
    rolling_mean - rolling_std,
    rolling_mean + rolling_std,
    alpha=0.2,
    color="steelblue",
    label="±1 rolling std"
)
axes[0, 1].legend()

# 3. Seasonal subseries plot (by month of year)
monthly_data = df.copy()
monthly_data["Month_of_Year"] = monthly_data.index.month
monthly_data["Year"] = monthly_data.index.year

months_order = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
month_names = ["Jan", "Feb", "Mar", "Apr", "May", "Jun",
               "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"]

for i, month in enumerate(months_order):
    subset = monthly_data[monthly_data["Month_of_Year"] == month]
    axes[0, 2].scatter(
        subset["Year"], subset["Passengers"],
        label=month_names[i], s=30, alpha=0.8
    )
axes[0, 2].set_title("Seasonal Subseries Plot")
axes[0, 2].set_xlabel("Year")
```

```
# Only show first and last labels to avoid clutter
handles, labels = axes[0, 2].get_legend_handles_labels()
axes[0, 2].legend(handles[::3], labels[::3], fontsize=8)

# 4. Lag plot (lag-1)
from pandas.plotting import lag_plot
lag_plot(df["Passengers"], lag=1, ax=axes[1, 0])
axes[1, 0].set_title("Lag Plot (Lag 1)")
axes[1, 0].set_xlabel("Passengers at t-1")
axes[1, 0].set_ylabel("Passengers at t")

# 5. Histogram and density
axes[1, 1].hist(df["Passengers"], bins=20, density=True, alpha=0.6,
↪  color="steelblue", edgecolor="black")
df["Passengers"].plot(kind="kde", ax=axes[1, 1], color="red", linewidth=2)
axes[1, 1].set_title("Distribution of Observations")
axes[1, 1].set_xlabel("Thousands of passengers")

# 6. Log transform for variance stabilization
df["Log_Passengers"] = np.log(df["Passengers"])
df["Log_Passengers"].plot(ax=axes[1, 2], title="Log-Transformed Series",
↪  color="steelblue")
axes[1, 2].set_ylabel("Log(passengers)")

plt.tight_layout()
plt.savefig("chapter2_exploratory_analysis.png", dpi=150, bbox_inches="tight")
plt.show()
```

Running this code produces six plots revealing the series structure. The raw data shows clear upward trend and annual seasonality with peaks in summer months. The rolling mean smooths out seasonal fluctuations to expose the underlying growth trajectory. The seasonal subseries plot confirms that each month follows a roughly parallel upward path, indicating stable seasonal patterns riding on an increasing trend. The lag-1 plot shows strong positive correlation: higher passenger counts tend to be followed by higher counts. The histogram reveals right-skewness consistent with exponential growth. Finally, the log-transformed series stabilizes variance, making seasonal amplitude more constant over time.

Stationarity and Why It Matters

Stationarity is one of the most important concepts in time series analysis, yet it is frequently misunderstood. A stationary time series has statistical properties

(mean, variance, autocorrelation structure) that do not change over time. More precisely:

- **Strict stationarity** requires that the joint probability distribution of any collection of observations is invariant to time shifts. This is a strong condition rarely satisfied in practice.
- **Weak (covariance) stationarity** requires only that the mean is constant, the variance is finite and constant, and autocovariance depends only on the lag between observations, not on absolute time. Most practical tests and methods target weak stationarity.

Why does stationarity matter? Many classical forecasting models assume stationarity because it makes mathematical analysis tractable and statistical inference valid. ARIMA models explicitly require differencing to achieve stationarity before estimating parameters. Nonstationary data can produce spurious regression results where unrelated series appear correlated simply because they share common trends [12]. Prediction intervals derived under stationarity assumptions become unreliable when applied to nonstationary series.

However, stationarity should not be treated as an absolute requirement that all data must satisfy. Real-world processes are rarely stationary over long horizons: economies grow, populations change, technologies evolve. The practical question is whether a series can be made approximately stationary through transformation (differencing, detrending, log transforms) over the forecast horizon of interest.

Two standard tests assess stationarity empirically:

The **Augmented Dickey-Fuller (ADF) test** tests the null hypothesis that a unit root exists (the series is nonstationary) against the alternative that the series is stationary. A small p-value (typically below 0.05) rejects the null, suggesting stationarity. The ADF test has good power for detecting trending nonstationarity but can be conservative with structural breaks.

The **KPSS test** takes the opposite approach: its null hypothesis is stationarity against the alternative of a unit root. Here, a small p-value suggests nonstationarity. Using both tests together provides more robust conclusions because they have different failure modes. If ADF rejects nonstationarity and KPSS does not reject stationarity, evidence strongly supports stationarity.

Let us apply these tests to our airline data, both in raw form and after transformations commonly used to achieve stationarity.

```python
# Stationarity tests on original and transformed series

def test_stationarity(series, name):
    """Run ADF and KPSS tests and print results."""
    # ADF test: null hypothesis is non-stationary (unit root present)
    adf_result = adfuller(series.dropna())
    adf_stat = adf_result[0]
    adf_pvalue = adf_result[1]

    # KPSS test: null hypothesis is stationary
    kpss_result = kpss(series.dropna(), regression="c", nlags="auto")
    kpss_stat = kpss_result[0]
    kpss_pvalue = kpss_result[1]

    print(f"\n{name}:")
    print(f"  ADF statistic: {adf_stat:.4f}, p-value: {adf_pvalue:.4f}")
    print(f"  {'Stationary' if adf_pvalue < 0.05 else 'Non-stationary'} (ADF)")
    print(f"  KPSS statistic: {kpss_stat:.4f}, p-value: {kpss_pvalue:.4f}")
    print(f"  {'Stationary' if kpss_pvalue >= 0.05 else 'Non-stationary'}
    ↪  (KPSS)")

# Test original series
test_stationarity(df["Passengers"], "Original series")

# Test log-transformed series
test_stationarity(np.log(df["Passengers"]), "Log-transformed series")

# Test first difference of log series (log returns)
log_diff = np.log(df["Passengers"]).diff().dropna()
test_stationarity(log_diff, "First difference of log series")

# Test seasonal difference (12-month lag for monthly data)
seasonal_diff = df["Passengers"].diff(12).dropna()
test_stationarity(seasonal_diff, "Seasonal difference (lag 12)")

# Combined: first difference of seasonal difference of log
log_seasonal_diff = np.log(df["Passengers"]).diff(12).dropna()
test_stationarity(log_seasonal_diff, "Log + seasonal difference")
```

Typical output shows that the raw series fails both tests (ADF cannot reject nonstationarity, KPSS rejects stationarity), confirming strong nonstationarity from trend and growing seasonal amplitude. The log transformation alone does not achieve stationarity because trend remains. First differencing removes trend but leaves annual seasonality. Seasonal differencing at lag 12 removes

the annual cycle. Combining log transform with seasonal differencing typically produces a series that passes both tests, indicating approximate stationarity suitable for ARIMA modeling.

Autocorrelation and Partial Autocorrelation

Autocorrelation measures how a time series correlates with its own past values at different lags. Understanding autocorrelation patterns is essential for model identification, particularly for ARIMA models where the orders of autoregressive (AR) and moving average (MA) components are chosen based on autocorrelation behavior.

The **autocorrelation function (ACF)** computes the correlation between observations separated by lag k for each k from 1 up to some maximum. If ACF(k) is large and positive, values k periods apart tend to move together. The ACF plot shows these correlations as bars with confidence intervals: bars extending beyond the interval indicate statistically significant autocorrelation at that lag.

The **partial autocorrelation function (PACF)** measures the direct correlation between observations at lag k after removing the effects of intermediate lags. While ACF includes both direct and indirect relationships, PACF isolates only the direct link. For example, if today's value correlates with yesterday's and yesterday's with the day before, ACF at lag 2 captures both the direct two-day relationship and the indirect path through yesterday. PACF at lag 2 removes the yesterday effect to show only the pure two-day dependency.

ACF and PACF patterns provide diagnostic signatures for different model types:

- **Pure AR(p) process**: ACF tails off gradually (exponential decay or damped sine wave), while PACF cuts off sharply after lag p. The cutoff point in PACF suggests the AR order.
- **Pure MA(q) process**: ACF cuts off after lag q, while PACF tails off gradually. The ACF cutoff suggests the MA order.
- **ARMA(p,q) or ARIMA processes**: Both ACF and PACF tail off without sharp cutoffs. Model selection requires trying combinations of p and q orders and comparing information criteria.

Seasonal patterns produce characteristic spikes at seasonal lags. Monthly data with annual seasonality shows significant ACF values at lags 12, 24, 36, and so on. These seasonal autocorrelations guide the choice of seasonal ARIMA parameters.

Let us compute and visualize ACF and PACF for our airline series after transformation to stationarity.

```
# Create stationary series via log + seasonal differencing
stationary = np.log(df["Passengers"]).diff(12).dropna()

fig, axes = plt.subplots(1, 2, figsize=(14, 5))

# ACF plot
plot_acf(
    stationary,
    lags=48,
    ax=axes[0],
    title="ACF of Log-Seasonally Differenced Series",
    alpha=0.05,
    linewidth=2
)

# PACF plot
plot_pacf(
    stationary,
    lags=48,
    ax=axes[1],
    title="PACF of Log-Seasonally Differenced Series",
    alpha=0.05,
    linewidth=2
)

plt.tight_layout()
plt.savefig("chapter2_acf_pacf.png", dpi=150, bbox_inches="tight")
plt.show()

# Print key autocorrelation values
from statsmodels.tsa.stattools import acf, pacf
acf_values = acf(stationary, nlags=24)
pacf_values = pacf(stationary, nlags=24, method="ywfast")

print("\nKey ACF values:")
for lag in [1, 2, 3, 6, 11, 12]:
    if lag < len(acf_values):
        print(f"  Lag {lag:2d}: ACF = {acf_values[lag]:.4f}")

print("\nKey PACF values:")
for lag in [1, 2, 3, 6, 11, 12]:
```

```
    if lag < len(pacf_values):
        print(f"  Lag {lag:2d}: PACF = {pacf_values[lag]:.4f}")
```

For the differenced airline series, the ACF typically shows a significant spike at lag 1 (short-term dependence) and possibly at seasonal lags if residual seasonality remains. The PACF helps distinguish whether these patterns suggest AR or MA components. If PACF cuts off after lag 1 while ACF tails off gradually, an AR(1) component is suggested. These patterns directly inform the model specification we will explore in Chapter 5.

Seasonality Detection and Characterization

Seasonality is both a blessing and a curse for forecasting. It represents predictable structure that good models exploit to improve accuracy. At the same time, it complicates analysis because seasonal patterns can mask underlying trends, interact with other components in complex ways, and shift over time as conditions change.

Detecting seasonality begins visually: does the series show repeating patterns at regular intervals? Line plots and seasonal subseries plots (shown earlier) make this assessment straightforward for strong seasonality. For subtler patterns or automated detection on large collections of series, statistical methods help.

The **seasonal decomposition of time series by LOESS (STL)** is a robust method that separates any series into trend, seasonal, and remainder components without assuming a specific parametric form [13]. STL uses local regression (LOESS) to estimate the trend flexibly, then extracts seasonality as the average pattern at each position within the cycle. It handles changing seasonal amplitude and works well with outliers because LOESS is locally weighted.

Python's statsmodels library implements STL efficiently:

```python
# STL decomposition on log-transformed airline data
stl = STL(np.log(df["Passengers"]), period=12, robust=True)
result = stl.fit()

fig, axes = plt.subplots(4, 1, figsize=(12, 10), sharex=True)

result.observed.plot(ax=axes[0], title="Observed (Log Passengers)",
    color="steelblue")
result.trend.plot(ax=axes[1], title="Trend Component", color="darkorange")
result.seasonal.plot(ax=axes[2], title="Seasonal Component",
    color="forestgreen")
result.resid.plot(ax=axes[3], title="Remainder (Noise)", color="crimson")

for ax in axes:
    ax.set_ylabel("Value")

plt.tight_layout()
plt.savefig("chapter2_stl_decomposition.png", dpi=150, bbox_inches="tight")
plt.show()

# Analyze seasonal pattern
print("\nAverage seasonal effect by month (on log scale):")
monthly_effects = result.seasonal.groupby(result.seasonal.index.month).mean()
monthly_effects.index = month_names
print(monthly_effects.round(4))

# Check if remainder looks random (white noise)
test_stationarity(result.resid, "STL Remainder")
plot_acf(result.resid, lags=24, title="ACF of STL Remainder")
plt.show()
```

The STL decomposition reveals clear structure. The trend component shows steady upward growth in airline travel from 1949 to 1960. The seasonal component exhibits a repeating annual pattern with peaks around June through August (summer travel season) and troughs in winter months. The remainder should resemble random noise if the decomposition captured all systematic patterns; checking its ACF confirms whether significant autocorrelation remains unexplained.

For series with multiple seasonalities, STL can be applied iteratively or specialized methods like TBATS (Trigonometric, Box-Cox transformation, ARIMA errors, Trend, and Seasonal components) handle them directly [14]. Hourly electricity data, for example, requires modeling both daily (24-hour) and weekly (168-hour) seasonalities simultaneously.

Seasonality characterization also involves assessing stability: does the seasonal pattern remain constant over time, or does it evolve? Examining

STL seasonal components across different time windows reveals whether amplitude changes (seasonal effects growing or shrinking) or phase shifts occur (peak months moving). These insights guide model choice: stable seasonality suits classical methods like SARIMA, while evolving patterns may require more flexible approaches like dynamic harmonic regression or deep learning models that adapt to changing structure.

Understanding your data through visualization, decomposition, stationarity testing, and autocorrelation analysis is not optional preliminary work. It is the foundation upon which all forecasting decisions rest. Models applied blindly without this diagnostic step frequently fail in production because they miss structural features, overfit noise mistaken for signal, or violate assumptions that invalidate their predictions. Take time to explore your series thoroughly before selecting a method.

Chapter 3: Preparing Time Series Data

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Data Quality and Cleaning

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Handling Missing Values

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Temporal Train/Validation/Test Splitting

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Feature Engineering for Time Series

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Scaling and Transformations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Chapter 4: Evaluating Forecasts

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Forecast Error Metrics

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Probabilistic Forecast Metrics

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Validation Strategies

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

The Importance of Baselines

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Avoiding Evaluation Pitfalls

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Chapter 5: Classical Statistical Methods

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Naive and Simple Models

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Exponential Smoothing Family

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

ARIMA and SARIMA

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Automatic Model Selection

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

When Classical Methods Win

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Chapter 6: Machine Learning for Time Series

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Framing Time Series as Supervised Learning

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Tree-Based Models for Forecasting

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Feature Engineering at Scale

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Handling Multiple Series

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Practical Strengths and Limitations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Chapter 7: Deep Learning Foundations for Time Series

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Why Neural Networks for Sequences

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Recurrent Neural Networks

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

LSTM and GRU Architectures

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Practical Deep Learning for Time Series

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Chapter 8: Transformers and Modern Deep Learning Architectures

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Attention Mechanisms for Time Series

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Positional Encodings and Sequence Order

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Temporal Fusion Transformer

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Patch-Based and Frequency-Domain Models

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Benchmark Performance on M4/M5/GluonTS

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Chapter 9: Large Language Models Enter Forecasting

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

How Can Text Models Forecast Numbers?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Prompting Strategies for Time Series

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Zero-Shot and Few-Shot Forecasting

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Numerical Reasoning Limitations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Prompt-Based vs Model-Based Approaches

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

When LLMs Are the Wrong Tool

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Chapter 10: Pretrained Time-Series Foundation Models

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

The Foundation Model Paradigm for Time Series

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Chronos: Probabilistic Forecasting with Autoregressive Models

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

TimesFM and Google's Approach

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Time-LLM and Temporal Adaptation Layers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Tokenization and Representation of Numerical Sequences

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Chapter 11: Adapting LLMs for Forecasting

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Fine-Tuning Strategies

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Parameter-Efficient Adaptation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Retrieval-Augmented Forecasting

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Hybrid Architectures

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Multimodal Context Integration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Chapter 12: Advanced Forecasting Scenarios

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Multivariate Time Series

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Long-Horizon Forecasting

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Covariates and Exogenous Variables

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Hierarchical and Grouped Time Series

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Irregularly Sampled and Missing Data

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Chapter 13: Probabilistic Forecasting and Uncertainty

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Why Uncertainty Matters

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Classical Prediction Intervals

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Quantile Regression and Pinball Loss

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Deep Probabilistic Methods

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Conformal Prediction for Time Series

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Chapter 14: Production Systems and MLOps for Forecasting

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

From Prototype to Production

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Data Pipelines for Forecasting

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Model Serving Architectures

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Monitoring and Alerting

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Model Retraining and Lifecycle Management

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Cost and Efficiency Considerations

Chapter 15: Future Directions and Advanced Topics

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Where the Field Is Heading

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Emerging Architectures and Techniques

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Benchmarking and Evaluation Challenges

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

LLMs and the Human Forecaster

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Ethical and Societal Considerations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Preparing for What Comes Next

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

Conclusion: The Forecasting Journey Ahead

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.

References

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/forecastingwithlargelanguagemodels.