LUISS

Corso di laurea in Economia e Finanza
Percorso: Banche e Intermediari Finanziari

Cattedra Machine Learning

Dynamic Linear Models: a Bayesian computational approach

Prof.ssa Marta Catalano

RELATORE

Prof.ssa Megha Patnaik

CORRELATORE

Viviana Luisa Palade 779871

CANDIDATO

Contents

Tables and Figures

List of Tables

List of Figures

Abstract

This thesis poses a central question in empirical macroeconomics: *which inferential paradigm—frequentist or Bayesian—is more suitable for forecasting and for communicating uncertainty in economic time series?* We frame the question within a single, coherent modeling scaffold: Dynamic Linear Models (state–space representations). DLMs separate signal from noise in a transparent way and allow both paradigms to be placed on exactly the same structural footing, keeping the focus on interpretation rather than implementation.

The empirical setting is real GDP growth for Italy over a long historical sample, used as a representative macroeconomic series where uncertainty quantification matters for policy, risk management, and business planning. Parallel DLM specifications with identical state and observation components provide a neutral arena to contrast the two readings of uncertainty: the frequentist perspective, anchored in long-run sampling behavior and confidence statements, and the Bayesian perspective, centered on probability statements that update beliefs and deliver predictive credible sets.

The thesis sets out the conceptual and practical criteria that guide the choice between paradigms—how each frames uncertainty, supports decisions across tranquil and volatile regimes, and communicates risk to stakeholders—using DLMs as a common language.

Chapter 1

Dynamic Linear Models

Introduction

Time is the medium through which economies expand, cells divide, machines wear out, rivers swell and retreat with the seasons, and galaxies rotate. Observing such phenomena at regular intervals yields a *time series*: an ordered collection $\{Y_t\}_{t=1}^T$ indexed by time. Unlike a cross-section—where observations can be permuted without consequence—a time series must be read in order: earlier values shape expectations about later ones, and cause and effect flow forward. Turning this ordered evidence into guidance for policy, risk management, and planning requires statistical models that translate a noisy past into a plausible future.

A univariate time series is called stationary when its probabilistic structure is time-invariant (precise definitions in Section 1.2.1); in applied work, a common weaker requirement is that the mean, variance, and autocovariance depend only on the lag h, not on calendar time. When variability and dependence are stable across lags, the sample may be treated as a miniature of the long-run process—an idealization underpinning traditional time-series modeling.

The idea of stable statistical structure emerged in the early twentieth century. Studying sunspot numbers, *Yule* (1927) articulated an early notion of stationarity. *Slutsky* (1937) and *Wold* (1938) then showed that every covariance-stationary process admits a linear representation in past random shocks. These insights produced two complementary building blocks: autoregression, where the present depends on its own past values, and moving averages, where it depends on past shocks. During the 1930s these elements were unified in the autoregressive–moving-average family, *ARMA*(p, q). *Box and Jenkins* (1970) later supplied a comprehensive program for identification, estimation, and diagnosis. When the data are (approximately) stationary, ARMA models compress dependence into a handful of coefficients, admit exact likelihood evaluation, and deliver low-variance forecasts.

Stationarity, however, is often an illusion. Many economic, environmental, and biological indicators drift over time or change their variability. To address such patterns, *Box and Jenkins* extended ARMA to the autoregressive–integrated moving-average family, *ARIMA*. The "I" (integrated) indicates that differencing removes persistent levels to render fluctuations approximately stationary, after which ARMA machinery applies. Yet differencing is a blunt instrument: it erases long-run levels, constrains relationships to be time-invariant, and offers limited scope for incorporating covariates, seasonality, or regime shifts within a single coherent structure.

A more flexible representation introduces a latent *state* that carries forward the system's unobserved condition. In a state–space (Dynamic Linear) framework, an observable series arises from a latent state through linked observation and transition mechanisms. With suitable choices of design and evolution objects and with static states, one recovers ARMA and linear-regression models as special cases; allowing the state to evolve yields local-level smoothers, trend–seasonal decompositions, time-varying-parameter regressions, and stochastic-volatility filters. The recursive estimation machinery traces back to the *Kalman filter* (*Kalman*, 1960), adapted for statistical modeling by *Harrison and Stevens* (1976) and consolidated by *West and Harrison* (1997). In this sense, Dynamic Linear Models (DLMs) *generalize*—rather than replace—ARMA/ARIMA, recovering them as time-invariant cases while permitting structure to evolve. A formal state–space specification is given in Section 1.1.2.

Two complementary inferential traditions apply to these models. The *frequentist* perspective treats unknown quantities as fixed but unknown and frames uncertainty through long-run sampling behavior and confidence statements. The *Bayesian* perspective treats unknowns as random, encodes prior information, and expresses uncertainty through posterior and predictive probability statements. Each provides a distinct language for quantifying and communicating risk.

This chapter sets the conceptual stage. The thesis subsequently uses DLMs to examine a central question: which inferential paradigm—frequentist or Bayesian—better serves forecasting and uncertainty communication in economic time series?

1.1 State-Space Models and Dynamic Linear Models

ARIMA's success highlights the value of linear dynamics, yet its difference–stationary shell can be rigid when levels drift, variances shift, or data arrive at uneven intervals. To retain parsimony while allowing such features to evolve, we move to the broader class of *state–space systems*, which treat the observable series as a noisy projection of an underlying process that changes over time.

1.1.1 State-space models: a general framework

Modern time series often display evolving dynamics, structural breaks, or irregular sampling that strain classical ARIMA analysis. State–space models address this by positing that observable data are generated from latent processes whose behavior may itself vary over time—an essential flexibility for financial volatility, macroeconomic trends, and other non-stationary phenomena.

State–space modeling sits at the crossroads of control engineering and statistics. Building on Wiener–Kolmogorov filtering, *Kalman* (1960) introduced a fully recursive solution for linear–Gaussian systems; the framework rapidly entered economics and demography, where it underpins trend–cycle decomposition, measures of unobserved inflation expectations, stochastic volatility for returns, and yield-curve forecasting. Monographs by *Harvey* (1989) and *Durbin and Koopman* (2001) turned these ideas into standard empirical tools.

At its core, a state–space model views the observable series y_t as a noisy projection of an unobserved state θ_t. Two linked mechanisms define the system: an *observation equation* mapping state to data and a *state* (or *evolution*) equation governing how θ_t changes over time. Special choices recover static regression, ARMA processes, or hidden Markov chains; richer choices allow time-varying parameters, hierarchical layers, missing data, or mixed frequencies within one coherent structure.

Formally, a general state–space model is

$$y_t = h_t(\theta_t, \varepsilon_t), \qquad \theta_t = g_t(\theta_{t-1}, \omega_t), \tag{1.1}$$

where $h_t(\cdot)$ and $g_t(\cdot)$ are known mappings, and the mutually independent noise terms ε_t and ω_t may be Gaussian or non-Gaussian.

A central assumption is that the latent process is *first-order Markov*, for which each state depends only on its immediate predecessor, so that the random vector θ_t condenses the entire past into a single probabilistic summary that mediates between history and future. This relationship can be formalized as:

$$p(\theta_t \mid \theta_{t-1}, \theta_{t-2}, \ldots) = p(\theta_t \mid \theta_{t-1}), \tag{1.2}$$

and that observations are conditionally independent given the states. It follows that the likelihood factorizes as

$$p(y_{1:n} \mid \theta_{1:n}) = \prod_{t=1}^{n} f(y_t \mid \theta_t), \tag{1.3}$$

and the joint law separates measurement noise from state dynamics,

$$p(\theta_0, \ldots, \theta_n, y_1, \ldots, y_n) = p(\theta_0) \prod_{t=1}^{n} f(y_t \mid \theta_t)\, p(\theta_t \mid \theta_{t-1}), \tag{1.4}$$

which paves the way for recursive filters. Marginal likelihoods for $y_{1:n}$ are then obtained by integrating out the latent states from (1.4). The information flow in a state–space model is summarized in Figure 1.1, with the latent chain mediating between past and future observations.

$$\begin{tikzcd}
\theta_0 \arrow[r] & \theta_1 \arrow[r] \arrow[d] & \theta_2 \arrow[r] \arrow[d] & \cdots \arrow[r] & \theta_{t-1} \arrow[r] \arrow[d] & \theta_t \arrow[r] \arrow[d] & \theta_{t+1} \arrow[r] \arrow[d] & \cdots \\
 & Y_1 & Y_2 & & Y_{t-1} & Y_t & Y_{t+1} &
\end{tikzcd}$$

Figure 1.1: Information flow in a state–space model

If the state variables are discrete, the specification becomes a *hidden Markov model* (HMM); otherwise it is a continuous state–space system. In both cases, the latent chain mediates between past and future without requiring a high-order autoregression in the measurement equation.

Within this general scheme, the linear–Gaussian specialization delivers fully recursive and closed-form inference. The next subsection adopts precisely this specialization—Dynamic Linear Models—by specifying linear observation and evolution mappings with Gaussian disturbances, so that filtering, smoothing, and forecasting follow from a single set of matrix recursions.

1.1.2 Formal structure of DLMs

Empirical series such as financial volatility, macroeconomic indicators, and other non-stationary signals often require models that can adapt in real time while retaining the interpretability of linear dynamics. Dynamic Linear Models (DLMs) meet that need: they embed classical regression and ARMA structure within a state-space framework whose parameters evolve between observations, so forecasts respond smoothly to structural change without repeated differencing or ad-hoc regime switches.

The idea goes back to Harrison and Stevens's article *Bayesian Forecasting* (1976), which translated Kalman-filter recursions into statistical language and emphasized sequential updating. West and Harrison's monograph *Bayesian Forecasting and Dynamic Models* (1997) then extended the framework to multilayer systems and heavy-tailed errors, securing adoption in economic, engineering, and environmental forecasting.

Conceptually, a DLM represents a flexible class of parametric models designed to analyze univariate and multivariate time series. Structurally, it maintains the usual state-space split into an *observation equation* and a *state equation*. This structure lets an analyst decompose complex time-dependent phenomena into their underlying elements—such as level, slope, seasonal variation, and exogenous effects—while keeping a coherent probabilistic description of the data-generating process.

In the DLM specification corresponding to the template in (1.1), both mappings are linear and both noises are Gaussian. The assumption of normality simplifies inference by enabling closed-form Bayesian updating via the Kalman filter, facilitating the computation of posterior moments and a coherent interpretation of uncertainty through credible intervals and predictive distributions. That recursive convenience makes DLMs ideal for continuous forecasting and real-time monitoring.

The Gaussian assumption is often justified by the Central Limit Theorem (CLT), which states that, under appropriate conditions, the average of a large number of independent and identically distributed (i.i.d.) random variables $X_1, \ldots, X_n$, with finite mean μ and variance σ^2, converges in distribution to a standard normal, even when the original variables are not themselves normal. Formally:

$$\frac{1}{\sqrt{n}} \sum_{i=1}^{n} \left(\frac{X_i - \mu}{\sigma} \right) \xrightarrow{d} \mathcal{N}(0, 1).$$

The DLM framework can still accommodate non-Gaussian choices—e.g., Poisson or Student's *t*—when heavy tails, asymmetry, or discreteness are empirically required, using simulation-based filters and smoothers.

Exploiting the state-space template in (1.1), a linear–Gaussian DLM is written as:

$$Y_t = F_t^\top \theta_t + \varepsilon_t, \qquad \varepsilon_t \sim \mathcal{N}(0, V_t), \tag{1.5}$$

$$\theta_t = G_t \theta_{t-1} + \omega_t, \qquad \omega_t \sim \mathcal{N}(0, W_t). \tag{1.6}$$

Here Y_t denotes the observed datum at time t—a scalar in the univariate case or an $m \times 1$ vector in the multivariate case—assumed conditionally normal given the latent state θ_t; F_t is the design object (a $p \times 1$ vector or an $m \times p$ matrix) that maps the latent state to the observation space; $\theta_t \in \mathbb{R}^p$ is the latent state vector encoding the underlying structure (e.g., level, slope, trend, seasonality, regression effects); $G_t \in \mathbb{R}^{p \times p}$ governs the temporal evolution of θ_t; V_t is the observational variance/covariance capturing measurement uncertainty; and $W_t \in \mathbb{R}^{p \times p}$ is the state-noise covariance controlling adaptability (smoothness vs responsiveness). Because only selected entries of F_t, G_t, V_t, W_t need vary with t, one can tune how quickly the system reacts to shocks.

If the state were held fixed, the specification would reduce to ordinary regression or finite-order ARMA. Allowing the state to drift transforms the same linear skeleton into a local-level smoother, a trend–seasonal extractor, a time-varying-parameter regression, or a stochastic-volatility filter. In that sense a DLM is not a competitor to ARMA but its natural generalization, retaining linear parsimony while replacing rigid difference-stationarity with an adaptive stochastic structure. This balance of flexibility, interpretability, and tractability explains why DLMs underpin applications ranging from intraday volatility tracking and GDP nowcasting to adaptive demand forecasts in energy markets, yield-curve estimation, inflation-expectation extraction, and algorithmic trading.

Multivariate block form. Extending (1.5)–(1.6) to m observed series and a p-dimensional state yields

$$\begin{bmatrix} y_t^{(1)} \\ y_t^{(2)} \\ \vdots \\ y_t^{(m)} \end{bmatrix} = \begin{bmatrix} F_t^{(1,1)} & F_t^{(1,2)} & \cdots & F_t^{(1,p)} \\ F_t^{(2,1)} & F_t^{(2,2)} & \cdots & F_t^{(2,p)} \\ \vdots & \vdots & \ddots & \vdots \\ F_t^{(m,1)} & F_t^{(m,2)} & \cdots & F_t^{(m,p)} \end{bmatrix} \begin{bmatrix} \theta_t^{(1)} \\ \theta_t^{(2)} \\ \vdots \\ \theta_t^{(p)} \end{bmatrix} + \begin{bmatrix} \varepsilon_t^{(1)} \\ \varepsilon_t^{(2)} \\ \vdots \\ \varepsilon_t^{(m)} \end{bmatrix}$$

$$\begin{bmatrix} \theta_t^{(1)} \\ \theta_t^{(2)} \\ \vdots \\ \theta_t^{(p)} \end{bmatrix} = \begin{bmatrix} G_t^{(1,1)} & G_t^{(1,2)} & \cdots & G_t^{(1,p)} \\ G_t^{(2,1)} & G_t^{(2,2)} & \cdots & G_t^{(2,p)} \\ \vdots & \vdots & \ddots & \vdots \\ G_t^{(p,1)} & G_t^{(p,2)} & \cdots & G_t^{(p,p)} \end{bmatrix} \begin{bmatrix} \theta_{t-1}^{(1)} \\ \theta_{t-1}^{(2)} \\ \vdots \\ \theta_{t-1}^{(p)} \end{bmatrix} + \begin{bmatrix} \omega_t^{(1)} \\ \omega_t^{(2)} \\ \vdots \\ \omega_t^{(p)} \end{bmatrix}$$

with $V_t \in \mathbb{R}^{m\times m}$ capturing cross-sectional observation covariance. No further changes to (1.5)–(1.6) are required beyond these dimensional adjustments.

1.2 Modular Components of Dynamic Linear Models

A key strength of Dynamic Linear Models lies in their modular specification, which enables distinct structural elements—trend, seasonality, and regression effects—to be combined to represent complex temporal dynamics. Before constructing comprehensive forecasting systems, it is crucial to identify and understand these fundamental components.

Each component is presented in state–space form, with emphasis on how the variance ratio W/V (see Section 1.3.2) mediates the trade-off between smoothness and responsiveness. Tuning these quantities controls sensitivity to short-term fluctuations versus the ability to capture long-run movements.

Moreover, these stochastic specifications admit translations into classical time-series processes (AR, MA, and differenced ARIMA), establishing a bridge to the Box–Jenkins methodology and clarifying how DLMs generalize traditional approaches.

Altogether, this modular perspective supports the construction of interpretable and flexible systems capable of capturing structural change, seasonal patterns, and the influence of exogenous variables, while retaining the recursive estimation benefits intrinsic to DLMs. The next subsections introduce seasonal blocks and dynamic regression components that enhance the model's capacity to represent real-world time-series behavior.

1.2.1 Classical Time Series Models

Classical time-series models, notably those in the ARMA and ARIMA families, have long provided essential tools for analysis and forecasting. Their appeal lies in conceptual and mathematical clarity: by representing a process as a combination of its past values and random shocks, these models yield interpretable and tractable descriptions of time dependence under stationarity.

The idea that a time series can be modeled as a function of its past values and stochastic innovations emerged in the early 20th century. Pioneering work by *Yule* (1927), *Slutsky* (1937), and *Wold* (1938) laid the foundation for autoregressive and moving-average structures; their integration led to autoregressive–moving-average (ARMA) models. *Box and Jenkins* (1970) then systematized practice through a full modeling cycle—identification, estimation, and diagnostics—making ARMA and its extension ARIMA widely accessible. These developments revolutionized short-term forecasting and remain central in teaching and practice.

A key property underlying these models is *stationarity*. Roughly, a stationary series has a time-invariant probabilistic structure. More precisely, two notions are distinguished (for $t, h \in \mathbb{Z}$). A series $\{Y_t\}$ is *strong* (or *strict*) stationary if the joint distribution of any finite collection is invariant under time shifts: for any integers $t_1, \dots, t_k$ and any shift h, the distribution of $(Y_{t_1}, \dots, Y_{t_k})$ equals that of $(Y_{t_1+h}, \dots, Y_{t_k+h})$. By contrast, *weak* (or *covariance*) stationarity requires only constant first and second moments and an autocovariance that depends solely on the lag:

$$\mathbb{E}[Y_t] = \mu < \infty, \qquad \mathrm{Var}(Y_t) = \sigma^2 < \infty, \qquad \gamma(h) := \mathrm{Cov}(Y_t, Y_{t+h}) \text{ depends only on } h.$$

Under ARMA-type specifications, weak stationarity is generally sufficient and more tractable; it ensures mean reversion, whereby shocks have temporary effects and the series fluctuates around a stable level.

This stability notion is crucial for identification, estimation, and forecasting. The subsections that follow examine how autoregressive and moving-average components underpin ARMA and ARIMA models and how these structures embed naturally within the Dynamic Linear Model framework.

Autoregressive (AR) Model

An autoregressive (AR) model captures persistence by expressing the current value of a time series as a linear combination of its own past values plus a stochastic innovation. Formally, an AR(p) process is

$$Y_t = \sum_{i=1}^{p} \phi_i Y_{t-i} + \varepsilon_t,$$

where $\{\varepsilon_t\}$ is white noise[1] (i.e., uncorrelated, zero mean, constant variance) and $\phi_1, \ldots, \phi_p$ are autoregressive coefficients. The order p indicates how many past observations directly influence Y_t.

Using the lag (backshift) operator L ($L^h Y_t = Y_{t-h}$), the model can be written compactly as

$$\Phi(L) Y_t = \varepsilon_t, \qquad \Phi(L) = 1 - \phi_1 L - \cdots - \phi_p L^p.$$

This notation facilitates theoretical analysis, particularly when examining stationarity properties. Stationarity in AR models is guaranteed when all roots of the characteristic polynomial $\Phi(z) = 1 - \phi_1 z - \cdots - \phi_p z^p$ lie outside the unit circle in the complex plane; that is, $|\lambda_i| > 1$ for all i. This condition ensures that the process is mean-reverting, and that shocks to the system have only temporary effects.

AR models are well suited to series with momentum or persistent autocorrelation and form the backbone of more complex specifications in econometrics and machine learning.

Moving Average (MA) Model

A moving-average (MA) model describes a time series as a linear function of current and past shocks. Unlike autoregressive models, which feed back past values of the series, MA models capture short-term dependence by smoothing innovations. Formally, an MA(q) process is

$$Y_t = \varepsilon_t + \sum_{j=1}^{q} \theta_j \, \varepsilon_{t-j},$$

where $\{\varepsilon_t\}$ is white noise (zero mean, constant variance, no autocorrelation). The order q specifies how many past shocks directly affect Y_t. Using the lag operator L,

$$Y_t = \Theta(L) \, \varepsilon_t, \qquad \Theta(L) = 1 + \theta_1 L + \cdots + \theta_q L^q.$$

[1] A white-noise process (WN) satisfies $\mathbb{E}[\varepsilon_t] = 0$, $\mathrm{Var}(\varepsilon_t) = \sigma^2$, and $\mathrm{Cov}(\varepsilon_t, \varepsilon_s) = 0$ for all $t \neq s$. If $\varepsilon_t \sim \mathcal{N}(0, \sigma^2)$ the process is Gaussian white noise.

MA(q) processes are (covariance) stationary for any parameter vector $\theta = (\theta_1, \dots, \theta_q)$ provided ε_t is white noise. Their autocorrelation function (ACF) cuts off after lag q, reflecting finite memory of shocks.

Invertibility

Invertibility ensures that an MA(q) process can be written as an infinite-order AR process. This is necessary for identifiability: otherwise, multiple parameterizations could yield the same autocovariance structure. A MA(q) process is invertible if all the roots of $\Theta(z) = 1 + \theta_1 z + \cdots + \theta_q z^q$ lie outside the unit circle, i.e., $|\lambda_i| > 1$. In this case, the process can be written as:

$$\varepsilon_t = \sum_{j=0}^{\infty} \psi_j Y_{t-j}, \quad \text{with } \psi_0 = 1,$$

where $\Psi(L) = \sum_{j=0}^{\infty} \psi_j L^j$ satisfies $\Theta(L)\Psi(L) = 1$. The ψ_j coefficients are computed recursively:

$$\begin{aligned}
\psi_0 &= 1, \\
\psi_1 &= -\theta_1, \\
\psi_2 &= -\theta_2 - \theta_1\psi_1, \\
\psi_3 &= -\theta_3 - \theta_1\psi_2 - \theta_2\psi_1, \\
&\vdots \\
\psi_j &= -\theta_j - \sum_{k=1}^{j-1} \theta_k \psi_{j-k}, \quad \text{with } \theta_j = 0 \text{ for } j > q.
\end{aligned}$$

This recursive structure defines the autoregressive coefficients of the AR(∞) representation. Importantly, two MA processes with different parameterizations but the same autocovariance function are said to be *observationally equivalent in the second-order sense*. To avoid this ambiguity, estimation procedures typically constrain the parameter space to the invertible region, ensuring a unique and stable model specification.

Model Identification Tools: ACF and PACF

An essential step in building time series models is to identify their structure from the data, and two fundamental diagnostics for this purpose are the autocorrelation function (ACF) and the partial autocorrelation function (PACF). The ACF, denoted $\rho(h)$, measures the linear dependence between Y_t and its lagged value Y_{t-h} and is defined as

$$\rho(h) = \frac{\operatorname{Cov}(Y_t, Y_{t-h})}{\sqrt{\operatorname{Var}(Y_t)\operatorname{Var}(Y_{t-h})}}.$$

The PACF at lag h captures the direct linear influence of Y_{t-h} on Y_t once the effects of the intermediate lags $(Y_{t-1}, \ldots, Y_{t-h+1})$ have been removed. Patterns in ACF and PACF plots guide model selection: for an AR(p) process the PACF displays an abrupt cutoff after lag p while the ACF decays gradually; for an MA(q) process the ACF cuts off after lag q while the PACF decays gradually; and for an ARMA(p, q) process both the ACF and the PACF typically show exponential or damped sinusoidal decay without a clear cutoff. These decay behaviors reflect core properties of the underlying process—stationarity in autoregressive models, invertibility in moving-average models, and the more intricate dynamics of combined ARMA specifications—and their proper interpretation is a cornerstone of the Box–Jenkins methodology, informing the preliminary specification of candidate models.

AutoRegressive Moving Average (ARMA) Model

ARMA models combine autoregressive (AR) and moving-average (MA) components to describe stationary time series, capturing both persistence from past values and the impact of past innovations. Formally, an ARMA(p, q) process is

$$Y_t = \phi_1 Y_{t-1} + \cdots + \phi_p Y_{t-p} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q},$$

where $\{\varepsilon_t\}$ is white noise, and $\phi_1, \ldots, \phi_p$ and $\theta_1, \ldots, \theta_q$ are the AR and MA coefficients, respectively. Using the lag operator L,

$$\Phi(L) Y_t = \Theta(L)\, \varepsilon_t, \qquad \Phi(L) = 1 - \phi_1 L - \cdots - \phi_p L^p, \;\; \Theta(L) = 1 + \theta_1 L + \cdots + \theta_q L^q.$$

Stationarity requires that all zeros of $\Phi(z) = 0$ lie outside the unit circle ($|z| > 1$), while invertibility requires the zeros of $\Theta(z) = 0$ to lie outside the unit circle. In practice one also rules out common factors between $\Phi(\cdot)$ and $\Theta(\cdot)$ to ensure identifiability.

ARMA models admit a state–space representation (not unique). One convenient innovations form for ARMA(1,1),

$$Y_t = \phi Y_{t-1} + \varepsilon_t + \theta\, \varepsilon_{t-1},$$

uses the state vector $\theta_t = \big(Y_t,\ \varepsilon_t\big)^\top$:

$$\theta_t = \begin{bmatrix} \phi & \theta \\ 0 & 0 \end{bmatrix} \theta_{t-1} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} \varepsilon_t, \qquad Y_t = \begin{bmatrix} 1 & 0 \end{bmatrix} \theta_t.$$

Here the observation equation carries no additional measurement noise and uncertainty enters via the state disturbance ε_t; this innovations form is likelihood-equivalent to the standard ARMA specification.

ARMA models are widely applied in economics, engineering, and environmental sciences for capturing short-run dynamics. However, their stationarity assumption can be restrictive when structural shifts or long-term trends are present; ARIMA or time-varying models provide more flexibility.

AutoRegressive Integrated Moving Average (ARIMA) Model

ARIMA models generalize ARMA processes to non-stationary series by differencing. An ARIMA(p, d, q) is defined by

$$(1-L)^d Y_t = \phi_1 Y_{t-1} + \cdots + \phi_p Y_{t-p} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q},$$

where L is the lag operator ($L^k Y_t = Y_{t-k}$) and d is the number of differences needed to achieve stationarity. The differencing operator $(1-L)^d$ removes persistent levels and trends; e.g.,

$$(1-L)Y_t = Y_t - Y_{t-1}, \qquad (1-L)^2 Y_t = Y_t - 2Y_{t-1} + Y_{t-2}.$$

After differencing, the process is modeled using ARMA techniques:

$$\Phi(L)(1-L)^d Y_t = \Theta(L)\,\varepsilon_t.$$

ARIMA models also have state–space representations, with differencing embedded in the transition dynamics. For example, one admissible representation of ARIMA(1,1,1) can be written as

$$\theta_t = \begin{bmatrix} 1+\phi & \theta \\ 1 & 0 \end{bmatrix} \theta_{t-1} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \varepsilon_t, \qquad Y_t = \begin{bmatrix} 1 & 0 \end{bmatrix} \theta_t,$$

for a suitably defined state vector θ_t; other equivalent formulations exist. This illustrates that ARMA/ARIMA models are encompassed within the DLM/state–space framework as time-invariant special cases.

While ARMA and ARIMA models remain foundational in time-series analysis, their stationarity assumption—whether imposed directly or achieved through differencing—can be restrictive for complex, real-world data. Empirical series often feature abrupt level shifts, heteroskedasticity, and structural breaks that fixed-parameter, static dynamics may capture only imperfectly. In such contexts, extensions of classical models or alternative formulations are required.

A natural resolution is to embed ARMA/ARIMA within the state–space (DLM) framework. Classical ARMA and ARIMA models arise as special cases with time-invariant

system matrices and deterministic observation noise, which in turn permits recursive filtering via the Kalman filter (see Section 1.3.2). Moreover, specific DLMs map to familiar ARIMA processes—e.g., the *local-level* model to $\mathrm{ARIMA}(0,1,1)$ and the *local-linear-trend* model to $\mathrm{ARIMA}(0,2,2)$—making explicit how DLMs extend and generalize classical approaches while better accommodating nonstationarity, time-varying parameters, and richer error structures.
Operationally, the match arises because a first-order stochastic trend (random-walk level) implies stationarity after one difference, while a second-order stochastic trend (random-walk slope) implies stationarity after two differences; the remaining short-run dynamics are captured by the MA terms.

1.2.2 Local Level and Local Trend DLMs

Economic and financial time series rarely follow rigid deterministic trends; instead they evolve gradually and occasionally shift regime, yet typically display *temporal continuity* (current values close to recent ones). Capturing this smooth evolution is essential for accurate modeling and forecasting.

A flexible way to do so is the family of *polynomial trend models*, in which latent components evolve according to polynomial functions of time whose coefficients follow stochastic processes. This framework includes the widely used *Local Level Model* and *Local Linear Trend Model*, which serve as building blocks for richer DLM specifications. Historically, these models arose as dynamic, probabilistic extensions of classical ARIMA methods, addressing their limitations with non-stationary behavior while enabling sequential updating via the Kalman filter—crucial for real-time applications where latent estimates are revised as new data arrive.

Formally, a polynomial trend model of order n has state–space form

$$\begin{aligned} y_t &= F^\top \theta_t + \varepsilon_t, \qquad && \varepsilon_t \sim \mathcal{N}(0, V), \\ \theta_t &= G\,\theta_{t-1} + \omega_t, \qquad && \omega_t \sim \mathcal{N}(0, W), \end{aligned}$$

where $\theta_t \in \mathbb{R}^n$ collects the latent trend components, $F^\top = (1, 0, \ldots, 0)$ selects the level, and G is the companion matrix

$$G = \begin{pmatrix} 1 & 1 & 0 & \ldots & 0 \\ 0 & 1 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & 1 \end{pmatrix}.$$

The two most common instances are summarized in Table 1.1:

Order n	**State Vector θ_t**	**Name**	**ARIMA Equivalent**
1	(μ_t)	**Local level** (random walk + noise)	ARIMA(0,1,1)
2	$(\mu_t,\, \beta_t)^\top$	**Local linear trend**	ARIMA(0,2,2)

Table 1.1: State–space specifications and their ARIMA equivalents.

Why these equivalences? In the local level model the state evolves as a random walk, so first differences of the observation follow a stationary ARMA law; this corresponds to an $\mathrm{ARIMA}(0,1,1)$ under standard conditions. In the local linear trend, the level's first difference (the slope) is itself a random walk, so second differences of the observation are approximately stationary with an $\mathrm{ARMA}(0,2,2)$ representation. In both cases, the order of differencing in ARIMA mirrors the polynomial order of the stochastic trend in the state-space form.

Local Level Model (Random Walk Plus Noise)

The Local Level Model is the simplest member of the polynomial-trend family. It assumes that the observed series y_t fluctuates around a latent level μ_t that evolves as a random walk:

$$\begin{aligned} y_t &= \mu_t + \varepsilon_t, \qquad & \varepsilon_t &\sim \mathcal{N}(0, V), \\ \mu_t &= \mu_{t-1} + \omega_t, \qquad & \omega_t &\sim \mathcal{N}(0, W), \end{aligned}$$

where y_t is the observation, μ_t the (unobserved) level, and ε_t and ω_t are independent Gaussian noises for observation and system, with variances V and W, respectively. This specification captures stochastic changes in the level without imposing a deterministic trend.

The system noise ω_t drives the level's evolution: when $W = 0$ the level is constant, while for $W > 0$ the process is non-stationary in levels. Importantly, the first differences $\Delta y_t = y_t - y_{t-1}$ can be stationary even if y_t is not, placing the model in the neighborhood of an $\mathrm{ARIMA}(0,1,1)$ under appropriate conditions. In state–space terms,

$$F^\top = (1), \qquad G = (1).$$

The label *Random Walk Plus Noise* (RWPN) is often used interchangeably with Local Level, emphasizing the random walk for the latent level and the measurement noise on observations. In time-series terminology, the *level* is the baseline value around which observations fluctuate, net of trend/seasonal components.

A convenient time-varying form (allowing for time-indexed variances) is

$$y_t = \mu_t + \varepsilon_t, \qquad \varepsilon_t \sim \mathcal{N}(0, V_t), \tag{1.7}$$

$$\mu_t = \mu_{t-1} + \omega_t, \qquad \omega_t \sim \mathcal{N}(0, W_t), \tag{1.8}$$

with latent state $\theta_t = \mu_t$, fixed scalar system/design $F_t = G_t = 1$, and variances (V_t, W_t) controlling measurement noise and adaptability.

The parameters W_t and V_t critically shape behavior. Smaller W_t yields smoother level paths; larger W_t allows rapid adjustments (the limit $W_t = 0$ freezes μ_t). Smaller V_t makes observations track the level closely. A common summary is the signal-to-noise ratio W/V (see Section 1.3.2): larger W/V enables faster adaptation; smaller W/V enforces conservative updates.

Example. Consider a simulated series with $m_0 = 0$, $C_0 = 10$, $V = 0.5$, $W = 1.0$, balancing moderate measurement noise with flexible level evolution. In DLM notation, $\theta_t = \mu_t$, $F_t = G_t = 1$, $V_t = V$, $W_t = W$. Figure 1.2 displays the trajectories of y_t and μ_t.

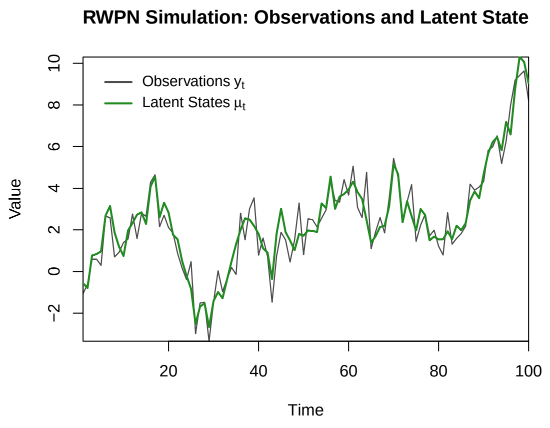
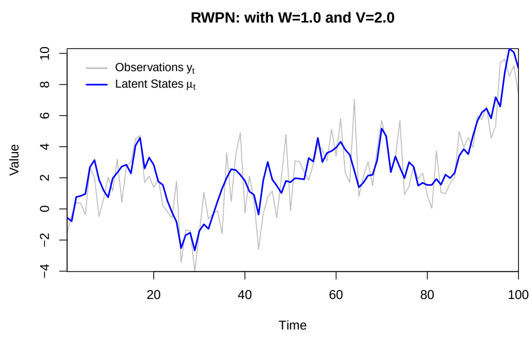
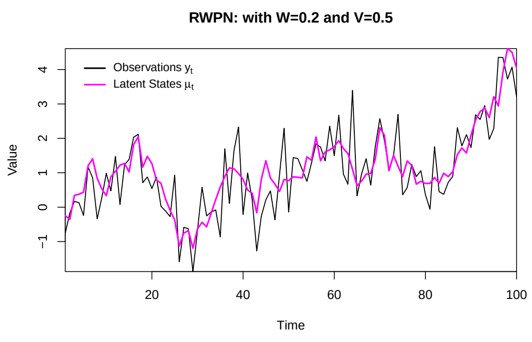


Figure 1.2: Simulation from a Random Walk Plus Noise (RWPN) model.

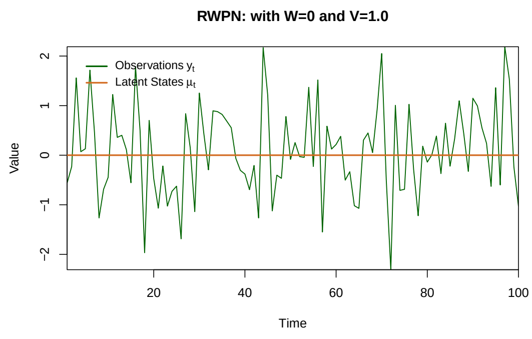
The comparative plot in Figure 1.3 clarifies the distinct roles of W and V:



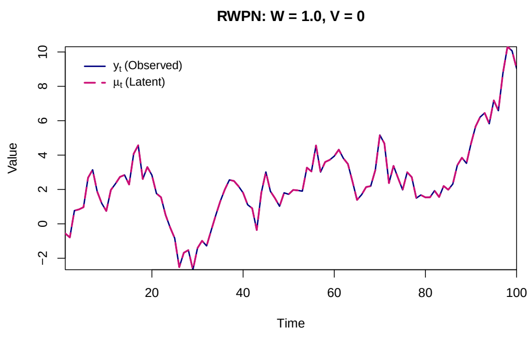
(a) V large



(b) W small



(c) $W = 0$



(d) $V = 0$

Figure 1.3: RWPN simulations under different W and V settings.

Local Linear Trend Model

In many economic and financial series, persistent upward or downward movements cannot be captured by a constant level. The *Linear Growth Model* (or *Local Linear Trend (LLT)*) extends the Local Level specification by adding a time–varying slope, so the trend itself can evolve smoothly over time.

This model is especially relevant for non-stationary patterns that are smoother than a pure random walk—e.g., GDP growth, inflation, or other macro indicators with evolving trends. Within the DLM framework, it balances interpretability and adaptability, supporting forecasting and sequential updating via the Kalman filter.

Formally, the model is the order-$n = 2$ polynomial trend:

$$\begin{aligned}
y_t &= \mu_t + \varepsilon_t, \qquad \varepsilon_t \sim \mathcal{N}(0, V),\\
\mu_t &= \mu_{t-1} + \beta_{t-1} + \omega_{1t}, \qquad \omega_{1t} \sim \mathcal{N}(0, W_1),\\
\beta_t &= \beta_{t-1} + \omega_{2t}, \qquad \omega_{2t} \sim \mathcal{N}(0, W_2),
\end{aligned}$$

where μ_t is the local level and β_t the time–varying slope. The k-step forecast mean is linear in the horizon,

$$f_t(k) = \hat{\mu}_t + k\,\hat{\beta}_t,$$

with $(\hat{\mu}_t, \hat{\beta}_t)$ the current posterior means of the state. In state–space notation,

$$F^\top = (1,\ 0), \qquad G = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \qquad W = \mathrm{diag}(W_1,\ W_2).$$

Setting $W_2 = 0$ freezes the slope (constant trend rate), while $W_2 > 0$ lets it adapt to structural changes.

Example: Impact of Variance Settings on Model Fit Two concepts from Section 1.3.2 guide behavior. First, the signal-to-noise ratio ($\mathrm{SNR} = W/V$) governs responsiveness: higher W/V yields a more reactive state (less smoothing), whereas lower W/V enforces smoother trajectories dominated by observation-noise filtering. Second, forecast errors $e_t = y_t - f_t$ permit accuracy assessment via three standard criteria,

$$\mathrm{MAD} = \frac{1}{n}\sum_{t=1}^{n}|e_t|, \qquad \mathrm{MSE} = \frac{1}{n}\sum_{t=1}^{n}e_t^2, \qquad \mathrm{MAPE} = \frac{1}{n}\sum_{t=1}^{n}\frac{|e_t|}{|y_t|}.$$

MAD summarizes typical absolute error (robust to sign); MSE emphasizes large misses; MAPE scales errors by the observation level (useful across units but unstable near $y_t \approx 0$).

To illustrate the trade-off, consider two specifications: a *flexible* model with `dW` = $(1000,\ 300)$ (high variability in level and slope) and a *rigid* model with `dW` = $(0.1,\ 0.1)$ (strong smoothing on both components). Figure 1.4 shows estimated level paths and residuals.

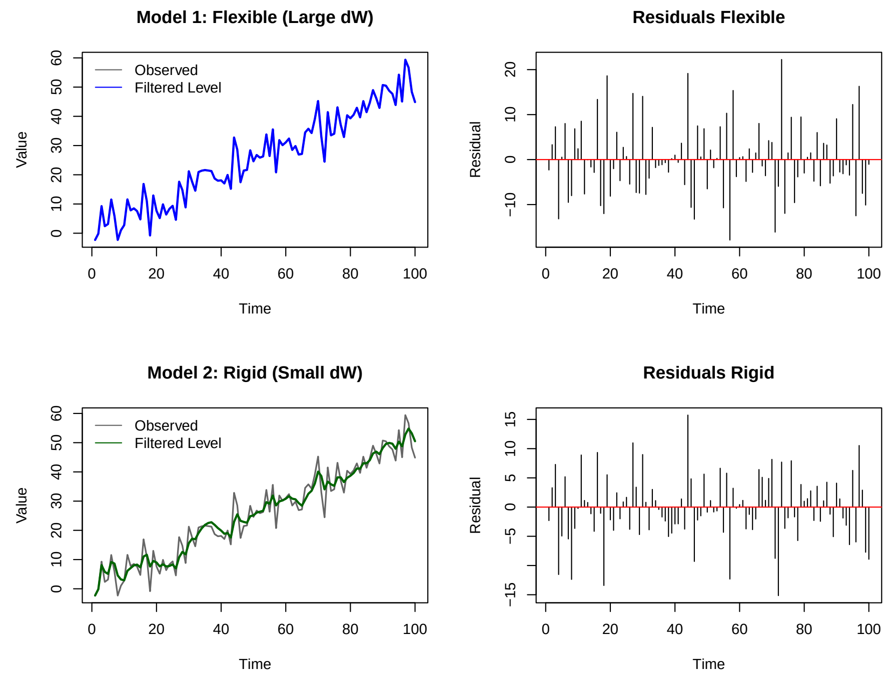


Figure 1.4: Comparison of Linear Growth Models under different system-noise variances.

Table 1.2 reports one-step-ahead accuracy (MAD, MSE, MAPE) for both settings.

Table 1.2: Forecast accuracy measures for flexible and rigid Linear Growth Models.

Model	**MAD**	**MSE**	**MAPE**
Flexible (large `dW`)	6.21	63.43	0.0148
Rigid (small `dW`)	4.34	30.80	0.1512

Lower values indicate better accuracy. The flexible model's higher system variance lets the level track data more closely (lower bias), at the cost of higher short-term variability in residuals; the rigid model enforces a smoother latent path, pushing short-run fluctuations into the residuals. This trade-off appears in the metrics: the rigid model attains lower MSE (more stable forecasts), while the flexible model achieves lower MAPE due to closer level alignment, albeit with larger MAD/MSE when noise is amplified.

Dynamic Regression with Covariates

Classical DLMs naturally accommodate external explanatory variables (*covariates*) that affect the series of interest—e.g., macro indicators or related-market prices. The resulting **Dynamic Regression Models** generalize the Normal DLM by allowing regression coefficients to evolve alongside the latent trend.

Let $\mathbf{x}_t \in \mathbb{R}^p$ denote the covariates at time t. A dynamic regression with stochastic level and time-varying coefficients is

$$\begin{aligned} y_t &= \mu_t + \mathbf{x}_t^\top \boldsymbol{\beta}_t + v_t, & v_t &\sim \mathcal{N}(0, V), \\ \mu_t &= \mu_{t-1} + w_{1t}, & w_{1t} &\sim \mathcal{N}(0, W_1), \\ \boldsymbol{\beta}_t &= \boldsymbol{\beta}_{t-1} + \mathbf{w}_{2t}, & \mathbf{w}_{2t} &\sim \mathcal{N}(\mathbf{0}, W_2), \end{aligned}$$

where μ_t is the local level, $\boldsymbol{\beta}_t \in \mathbb{R}^p$ collects the time-varying coefficients, and v_t, w_{1t}, $\mathbf{w}_{2t}$ are mutually independent Gaussian disturbances. This specification lets both the level and the covariate effects adapt over time. When $W_2 = 0$, coefficients are constant (static regression with stochastic trend); when $W_2 > 0$, coefficients adjust to structural changes in the predictor–outcome relationship.

Dynamic Regression (Normal DLM). In standard DLM notation,

$$y_t = F_t^\top \theta_t + \varepsilon_t, \quad \varepsilon_t \sim \mathcal{N}(0, V_t), \qquad \theta_t = G_t \theta_{t-1} + \omega_t, \quad \omega_t \sim \mathcal{N}(0, W_t),$$

with state vector $\theta_t = \left(\mu_t,\ \boldsymbol{\beta}_t^\top\right)^\top$, design vector $F_t = \left(1,\ \mathbf{x}_t^\top\right)^\top$ (intercept plus time-varying covariates), evolution matrix $G_t = I_{p+1}$ (random-walk evolution for each state component), and $W_t = \operatorname{diag}(W_1, W_2)$. This linear–Gaussian specification defines the *Normal DLM*.

To handle time-varying regressors in a modular way, introduce a selector (mapping) matrix JFF with the same dimension as F_t and link it to an external covariate array X. An entry 0 in JFF fixes the corresponding element of F_t over time (e.g., the intercept), whereas a positive integer k instructs that, at each t, the corresponding element of F_t takes the value from column k of X.

For example, with an intercept and one time-varying regressor x_t,

$$F_t = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \qquad JFF = \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$

so the second entry of F_t is populated over time by the series $\{x_t\}$ stored in X. This mechanism lets the Normal DLM incorporate external time-varying predictors without redefining the model structure at each step.

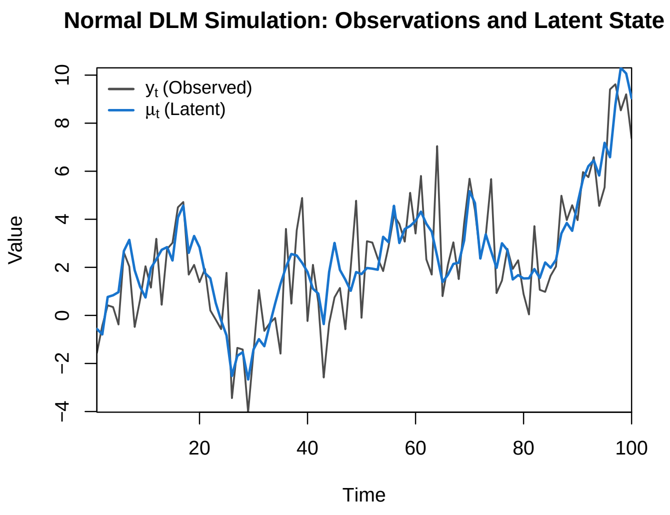


Figure 1.5: Simulation from a Normal DLM.

These simple examples serve as a practical foundation for understanding DLMs, upon which more complex dynamic structures can be built.

This modeling approach is particularly useful in financial applications where returns are influenced by time-varying market factors or economic fundamentals. The flexibility of this formulation enables the model to capture shifts in sensitivity to different drivers across time, which is crucial in policy evaluation, forecasting, and risk management.

1.2.3 Seasonality

Seasonality denotes systematic, calendar-related fluctuations that recur at fixed intervals (e.g., daily, monthly, quarterly), driven by climate, institutional cycles, or behavioral habits. Examples in finance include the "January effect" (unusually high January returns) and the "weekend effect" (return patterns across trading days). Ignoring seasonal structure biases estimates, inflates residual variance, and degrades forecasts; modeling it separates long-run trends from short-run cyclical movements.

Formally, a seasonal component has fixed periodicity S. In the deterministic case $s_t = s_{t+S}$ for all t; in a stochastic formulation the pattern evolves gradually while retaining its S-cycle nature.

Dynamic Linear Models accommodate seasonality in several ways—deterministic or stochastic, with fixed or time-varying amplitudes—which we detail in the following subsections.

Alternative Approaches

The trigonometric representation provides a smooth cyclical pattern. However, it can be more difficult to interpret and is omitted here for simplicity. See Harvey (1989), Durbin and Koopman (2001) for details.

Dummy Variable Representation of Seasonality

Recall that the seasonal component at time t can be expressed as

$$s_t = \mathbf{D}_t^\top \boldsymbol{\gamma},$$

where s_t denotes the seasonal effect at time t, $\mathbf{D}_t$ is a vector of dummy (indicator) variables, and $\boldsymbol{\gamma}$ collects the seasonal coefficients. For monthly data with annual seasonality ($S = 12$), $\mathbf{D}_t \in \mathbb{R}^{12}$ contains a single entry equal to 1 (the current month) and zeros elsewhere—for instance, if t corresponds to March,

$$\mathbf{D}_t = \big(0,\, 0,\, 1,\, 0,\, 0,\, 0,\, 0,\, 0,\, 0,\, 0,\, 0,\, 0\big)^\top,$$

and the coefficient vector is

$$\boldsymbol{\gamma} = \big(\gamma_1,\, \gamma_2,\, \ldots,\, \gamma_{12}\big)^\top.$$

Hence s_t simply picks the coefficient of the current season; for March, $s_t = \gamma_3$. This formulation presumes the seasonal pattern repeats identically each year and therefore treats seasonality as fixed effects—appropriate when the pattern is discrete and non-evolving.

State-Space Formulation In the Dynamic Linear Model (DLM) framework, seasonality is often modeled as a stochastic component that evolves over time. A common specification is the seasonal model of order S:

$$\begin{aligned} y_t &= \mu_t + s_t + v_t, & v_t &\sim \mathcal{N}(0, V), \\ s_t &= -\sum_{j=1}^{S-1} s_{t-j} + w_t, & w_t &\sim \mathcal{N}(0, W). \end{aligned}$$

This imposes the constraint

$$\sum_{j=0}^{S-1} s_{t-j} = 0,$$

which ensures identifiability by forcing seasonal effects to sum to zero over each cycle. Although the model appears to require S state variables, only $S - 1$ are free to vary, yielding a reduced, non-redundant representation that is computationally efficient while allowing the seasonal pattern to drift gradually.

Alternative State-Space Formulations An equivalent representation embeds the seasonal factors in the state and advances them by rotation:

$$\theta_t = G\,\theta_{t-1} + w_t, \qquad y_t = F\,\theta_t + v_t,$$

where θ_t is the state vector containing the seasonal coefficients, G is a permutation matrix that shifts the seasonal effects forward one period at each t, and $F = (1, 0, \ldots, 0)$ selects the season relevant at time t. This rotating design yields the same S-cycle structure while keeping the observation equation simple.

Illustration of Seasonality To illustrate how the dummy-variable representation captures seasonality, we simulated a time series with and without a seasonal component over five years of monthly data. The upper panels display a process with a strong positive January effect, whereas the lower panels show data generated without seasonality. The estimated seasonal coefficients reveal systematic calendar patterns when present—most notably the elevated January effect in the upper-right panel—while remaining centered around zero in the non-seasonal case, reflecting mere random fluctuations.

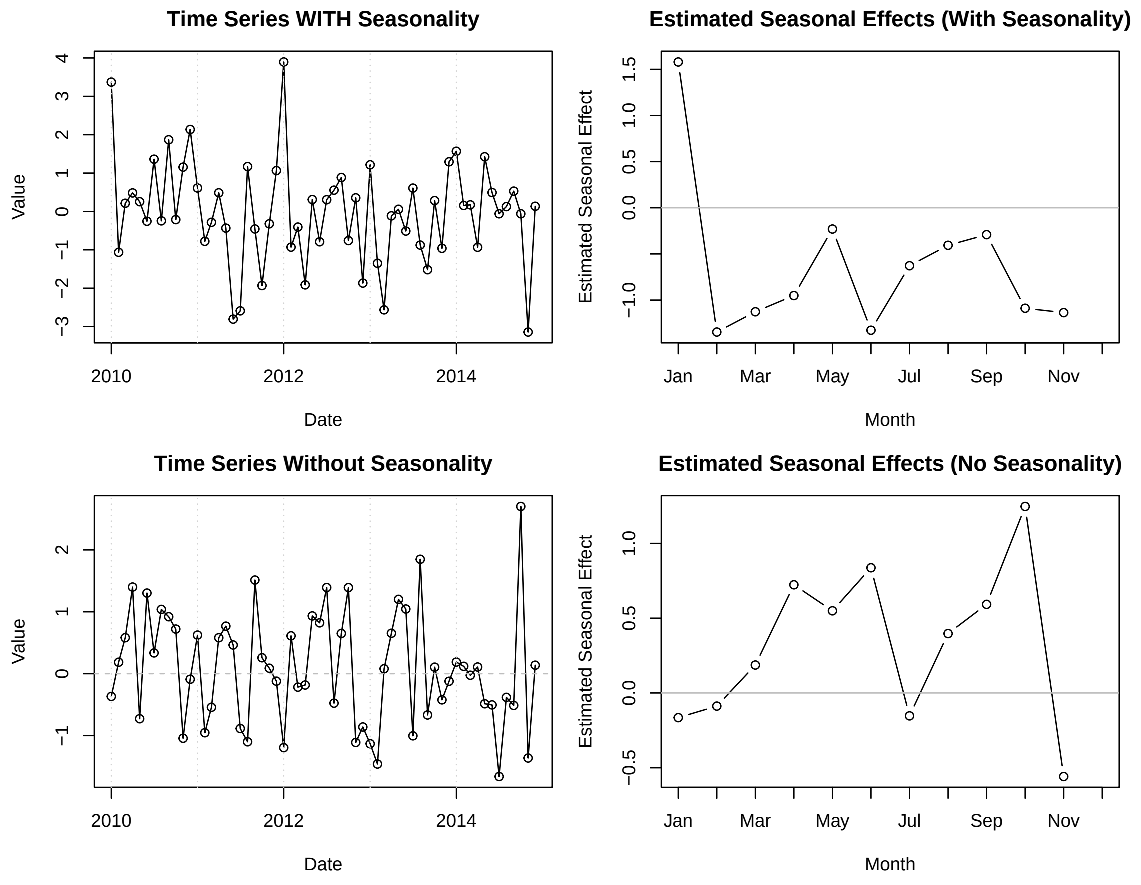


Figure 1.6: Simulation of seasonality

The simulated January effect is a stylized example of higher values occurring systematically at the beginning of each year, consistent with the tendency in financial markets for increased trading activity in January. In contrast, when no seasonality is present, the estimated effects oscillate around zero without persistent structure.

1.2.4 Extensions and Alternatives

Beyond the standard specifications discussed so far, Dynamic Linear Models (DLMs) can be extended in several directions to address more complex time–series structures.

Higher-Order Polynomial Trends The local level and local linear trend correspond to first- and second-order polynomial trends. More generally, an order-k polynomial trend is obtained by enlarging the state to include the level and its first $(k-1)$ derivatives, e.g. $\theta_t = \big(\mu_t, \mu_t^{(1)}, \ldots, \mu_t^{(k-1)}\big)^\top$, with a companion-form evolution. In practice, small values of k are typically sufficient to capture smooth trend dynamics and both deterministic and stochastic trends.

Non-Gaussian and Non-Linear Models When Gaussian observation errors or linear state evolution are inadequate, one may adopt non-Gaussian observation models and/or non-linear dynamics. A general observation equation is

$$y_t \mid \eta_t \sim p(\cdot \mid \eta_t), \qquad \eta_t = F_t^\top \theta_t,$$

covering, for instance, Poisson or binomial outcomes via appropriate likelihoods (and links). Non-linear state evolution can be written as

$$\theta_t = g(\theta_{t-1}) + \omega_t,$$

with suitable process noise. Estimation then typically relies on simulation-based or approximate methods (e.g., particle filters / sequential Monte Carlo, extended or unscented Kalman filtering).

Unknown Variances and Non-Informative Priors In practice, the observation variance V and system variance W (possibly time-varying as V_t, W_t) are rarely known. A frequentist analysis estimates them by maximizing the likelihood, whereas a Bayesian analysis places priors on these quantities. Non-informative or weakly informative priors are natural when prior knowledge is limited, though they may increase sensitivity to the data and computational demands.

Implications for Inference These extensions often require advanced computation to obtain parameter estimates and posterior summaries. Common tools include maximum likelihood (or EM/MAP) optimization, Gibbs sampling and more general MCMC, as well as sequential Monte Carlo for online inference. The next chapter discusses these estimation strategies in detail.

Such extensions enable modeling of non-linearity and non-Gaussian features, thereby increasing the flexibility of DLMs relative to classical approaches.

1.3 Statistical Inference

Statistical inference provides a formal mechanism for learning from data by quantifying uncertainty and updating beliefs in light of new information. In dynamic systems, where observations evolve over time and are influenced by latent factors, inference enables the recovery of hidden structures and the estimation of governing parameters.

The core task is to determine the probabilistic relationship between observed outputs and unobserved states, allowing for signal extraction, forecasting, and uncertainty quantification. In state–space settings, three complementary operations cover the temporal relations between states and data: *filtering*, which estimates the current latent state given information up to time t; *smoothing*, which refines past state estimates using the full sample; and *prediction/forecasting*, which projects the system forward beyond t.

When the system admits a state–space representation—particularly a linear–Gaussian one—these tasks become computationally tractable via recursive algorithms such as the Kalman filter, delivering exact Gaussian conditional distributions for the quantities of interest. By formalizing how evidence accumulates over time, filtering, smoothing, and prediction form the backbone of both real-time monitoring and retrospective analysis.

The same building blocks support both inferential paradigms. In the *Bayesian* approach, sequential filtering updates the posterior $p(\theta_t \mid y_{1:t})$ as new data arrive, smoothing recovers $p(\theta_{1:n} \mid y_{1:n})$, and combining these with priors on static parameters ψ yields the joint posterior $p(\theta_{1:n}, \psi \mid y_{1:n})$ (closed-form for linear–Gaussian DLMs, simulation-based otherwise). In the *frequentist* approach, the same prediction–error decomposition generated by filtering defines the likelihood $p(y_{1:n} \mid \psi)$; maximizing it delivers $\widehat{\psi}$, while filtered and smoothed states provide point estimates and standard errors. Thus, filtering, smoothing, and prediction are the common computational core underpinning both Bayesian updating and frequentist likelihood-based inference.

1.3.1 Inference and Forecasting in State-Space Models

In state–space models, inference concentrates on the latent state by evaluating conditional laws of the form $p(\theta_s \mid y_1, \ldots, y_t)$. When $s = t$, *filtering* targets the current state θ_t given information $I_t = \{Y_{1:t}\}$, yielding $p(\theta_t \mid I_t)$. When $s < t$, *smoothing* leverages the full sample to refine $p(\theta_s \mid I_t)$. When $s > t$, *forecasting* uses the current posterior to obtain $p(\theta_{t+k} \mid I_t)$ and the associated predictive distribution for y_{t+k}. Together, these perspectives deliver present-state estimation, future projections, and retrospective refinement within a unified probabilistic framework.

Dynamic Linear Models (DLMs) are a widely used subclass that balance tractability and flexibility. They represent a time series as the combination of an observable process y_t and a latent component θ_t, linked by probabilistic rules for measurement and evolution. Inference has two joint objectives: (i) extract the signal θ_t from $y_{1:t}$; (ii) forecast future observations y_{t+h} with quantified uncertainty. A recursive scheme naturally delivers both by updating beliefs sequentially as new information arrives.

The standard DLM is defined through the following equations:

$$y_t = F_t^\top \theta_t + v_t, \qquad v_t \sim \mathcal{N}(0, V_t), \tag{1.9}$$

$$\theta_t = G_t \theta_{t-1} + w_t, \qquad w_t \sim \mathcal{N}(0, W_t), \tag{1.10}$$

where F_t and G_t are known system matrices, and V_t, W_t are the observation and state noise covariances, respectively. Filtering produces $p(\theta_t \mid y_{1:t})$ and the one-step-ahead predictive density $p(y_{t+1} \mid y_{1:t})$; smoothing yields $p(\theta_{1:n} \mid y_{1:n})$. These quantities feed directly into Bayesian posterior analysis (via priors on ψ) and into frequentist likelihood construction and estimation (via the prediction-error decomposition).

1.3.2 Bayesian Updating and the Kalman Filter in DLMs

The Kalman filter, originally developed by Rudolf E. Kalman in his 1960 paper *A New Approach to Linear Filtering and Prediction Problems*, introduced a systematic method to estimate the state of linear dynamic systems in the presence of noise. Its widespread use is attributed to two key properties. It achieves optimal estimation in the sense of minimizing the mean squared error, and its efficiency and generality have made it foundational in econometrics, engineering, and beyond.

Setup. Under the Gaussian-linear assumptions of Dynamic Linear Models (DLMs), the filtering process starts from a prior distribution on the initial latent state:

$$\theta_0 \sim \mathcal{N}(m_0, C_0),$$

which encodes the initial belief about the hidden state of the system. The model evolves according to a linear-Gaussian state-space formulation, composed of an observation equation and a state transition equation:

$$Y_t = F_t^\top \theta_t + \varepsilon_t, \qquad \varepsilon_t \sim \mathcal{N}(0, V_t), \qquad \theta_t = G_t \theta_{t-1} + \omega_t, \quad \omega_t \sim \mathcal{N}(0, W_t), \tag{1.11}$$

where θ_t is the latent state vector at time t, Y_t is the observed scalar or vector variable, and $I_t = \{y_1, \dots, y_t\}$ denotes the data observed up to time t.

Thanks to the Gaussian distributions and the Markov property of the latent states, the posterior distribution $p(\theta_t \mid I_t)$ can be computed recursively using Bayes' theorem:

$$p(\theta_t \mid I_t) = \frac{p(y_t \mid \theta_t)\, p(\theta_t \mid I_{t-1})}{p(y_t \mid I_{t-1})}. \tag{1.12}$$

This formulation allows for a recursive decomposition of the inference process into three sequential steps: evolution, prediction, and update—each representing a core component of Bayesian learning over time.

Recursive Filtering Steps. The Kalman filter operates in three recursive stages:

1. Evolution Step. Before observing y_t, the *prior distribution* of θ_t is obtained by propagating the previous posterior through the transition model:

$$\theta_t \mid I_{t-1} \sim \mathcal{N}(a_t, R_t) \qquad \text{(Prior distribution)} \qquad (1.13)$$

$$p(\theta_t \mid I_{t-1}) = \int p(\theta_t \mid \theta_{t-1})\, p(\theta_{t-1} \mid I_{t-1})\, d\theta_{t-1} \qquad \text{(Evolution equation)} \qquad (1.14)$$

$$a_t = G_t m_{t-1} \qquad \text{(Predicted mean)} \qquad (1.15)$$

$$R_t = G_t C_{t-1} G_t^\top + W_t \qquad \text{(Predicted covariance)} \qquad (1.16)$$

2. Prediction Step. The forecast distribution of the next observation is:

$$y_t \mid I_{t-1} \sim \mathcal{N}(f_t, Q_t) \qquad \text{(Predictive distribution)} \qquad (1.17)$$

$$p(y_t \mid I_{t-1}) = \int p(y_t \mid \theta_t)\, p(\theta_t \mid I_{t-1})\, d\theta_t \qquad \text{(Marginal likelihood)} \qquad (1.18)$$

$$f_t = F_t^\top a_t \qquad \text{(Predicted observation mean)} \qquad (1.19)$$

$$Q_t = F_t^\top R_t F_t + V_t \qquad \text{(Predicted observation variance)} \qquad (1.20)$$

It updates the previous posterior mean on the dynamics encoded in G_t (the state transition matrix that governs the deterministic evolution of the state from $t-1$ to t). Meanwhile R_t quantifies the uncertainty in this prediction, combining uncertainty from the previous state with new system noise W_t.

The term f_t represents the forecast of the observable variable prior to observing y_t, and Q_t the predictive uncertainty.

3. Update Step. Once y_t is observed, Bayes' rule updates the prior:

$$p(\theta_t \mid I_t) = \frac{p(y_t \mid \theta_t)\, p(\theta_t \mid I_{t-1})}{p(y_t \mid I_{t-1})} \qquad \text{(Bayesian update)} \qquad (1.21)$$

$$K_t = R_t F_t Q_t^{-1} \qquad \text{(Kalman gain)} \qquad (1.22)$$

$$m_t = a_t + K_t(y_t - f_t) \qquad \text{(Updated mean)} \qquad (1.23)$$

$$C_t = R_t - K_t Q_t K_t^\top \qquad \text{(Updated covariance)} \qquad (1.24)$$

Here, K_t is the Kalman gain that weights the prediction error $(y_t - f_t)$ by the relative uncertainty; its interpretation is discussed in Appendix 1.4.2.

This recursive mechanism ensures that the updated state estimate m_t is a weighted average of the prior prediction and the new information from y_t, and that uncertainty is reduced post-update as captured by the posterior covariance C_t.

The posterior distribution of the latent state θ_t is then

$$\theta_t \mid I_t \sim \mathcal{N}(m_t, C_t),$$

where m_t and C_t represent the posterior mean and covariance of the state at time t.

The framework extends naturally to k-step-ahead forecasts:

$$p(\theta_{t+k} \mid I_t) = \int p(\theta_{t+k} \mid \theta_{t+k-1})\, p(\theta_{t+k-1} \mid I_t)\, d\theta_{t+k-1}, \tag{1.25}$$

$$p(y_{t+k} \mid I_t) = \int p(y_{t+k} \mid \theta_{t+k})\, p(\theta_{t+k} \mid I_t)\, d\theta_{t+k}, \tag{1.26}$$

and, given the linear-Gaussian structure, all moments of these predictive distributions can be computed analytically.

Summary. These steps—evolution, prediction, and update—form the backbone of Bayesian inference in DLMs and are compactly expressed via:

$$p(\theta_t \mid I_{t-1}) = \int p(\theta_t \mid \theta_{t-1})\, p(\theta_{t-1} \mid I_{t-1})\, d\theta_{t-1}, \tag{Evolution}$$

$$p(y_t \mid I_{t-1}) = \int p(y_t \mid \theta_t)\, p(\theta_t \mid I_{t-1})\, d\theta_t, \tag{Prediction}$$

$$p(\theta_t \mid I_t) \propto p(\theta_t \mid I_{t-1})\, p(y_t \mid \theta_t). \tag{Update}$$

Each step reflects a fundamental component of sequential Bayesian learning. The evolution step projects the previous posterior distribution of the state forward in time using the state transition model. This step produces the prior distribution $p(\theta_t \mid I_{t-1})$ by integrating out the uncertainty in θ_{t-1}.

The prediction computes the predictive distribution of the next observation y_t, given the evolved prior state. This involves marginalizing over the prior distribution of θ_t to obtain $p(y_t \mid I_{t-1})$.

The update refines the prior belief about the current state θ_t by incorporating the actual observation y_t. Using Bayes' rule, this yields the posterior distribution $p(\theta_t \mid I_t)$, combining the likelihood $p(y_t \mid \theta_t)$ and the prior $p(\theta_t \mid I_{t-1})$.

As long as F_t, G_t, V_t, and W_t are known and the Gaussian assumptions hold, this sequence yields exact solutions and enables fast, adaptive filtering. When these assumptions are relaxed, simulation-based methods are required—these will be addressed in the following chapter.

Signal-to-Noise Ratio Interpretation A useful way to interpret the Kalman filter is through the *signal-to-noise ratio* (SNR), which quantifies the relative magnitude of the process (state) variability compared to the observation noise. In the context of a Dynamic Linear Model, the process noise variance W_t controls how much the latent state θ_t is allowed to evolve over time, while the observation variance V_t captures how noisy the measurements are.

Intuitively, a high signal-to-noise ratio means that the system is highly dynamic (large W_t) relative to the observation noise (small V_t), so each new observation carries substantial information about the current state. This situation leads to a larger Kalman gain K_t, and the filter rapidly adapts to new data. Conversely, when the signal-to-noise ratio is low (small W_t and large V_t), the system is assumed to evolve slowly and measurements are unreliable, resulting in a smaller Kalman gain and more conservative updates that rely heavily on past estimates.

Formally, in the univariate case with $F_t = 1$ and $G_t = 1$, the Kalman gain simplifies to:

$$K_t = \frac{R_t}{R_t + V_t},$$

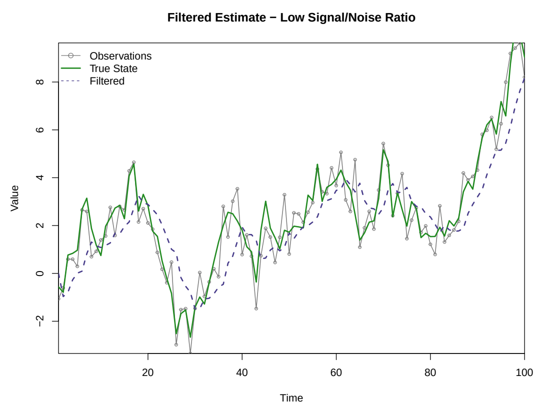
where R_t captures the predictive uncertainty about the state. This expression shows that the Kalman gain is always between 0 and 1, increasing when R_t (process variability or signal) dominates and decreasing when V_t (measurement noise) dominates. The signal-to-noise ratio can be approximated as:

$$\text{SNR}_t = \frac{\text{Var}(\text{Predictable component})}{\text{Var}(\text{Observation noise})} = \frac{R_t}{V_t}.$$

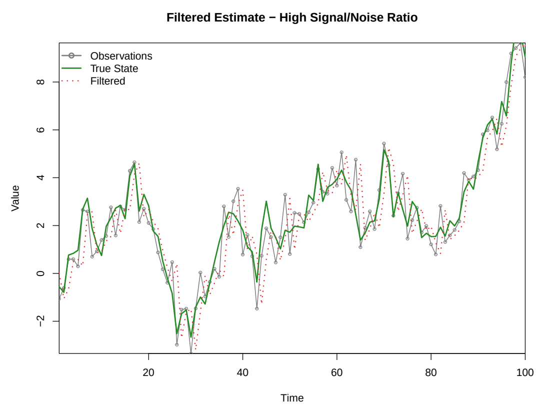
More generally, for arbitrary design vector F_t, the SNR is given by:

$$\text{SNR}_t = \frac{F_t^\top R_t F_t}{V_t}.$$

This ratio influences how much the filter relies on new observations relative to the prior prediction. In practice, adjusting W_t and V_t controls the balance between the responsiveness to data and the smoothness of the estimated trajectory. We illustrate this comparison using the RWPN example defined in Chapter 1.1.3, contrasting low and high SNR cases:



(a)



(b)

Figure 1.7: Comparison of filtering under different W and V settings.

Figure 1.7 illustrates the impact of the signal-to-noise ratio on the behavior of the Kalman filter and its adaptability. In the low signal-to-noise ratio case (Figure 1.7a), the observation noise variance $V = 1.0$ is large relative to the state noise variance $W = 0.1$. Consequently, the filter yields smoother and more conservative estimates. This means that the filtered trajectory responds gradually to new observations, remaining close to the prior model dynamics. In contrast, in the high signal-to-noise ratio scenario (Figure 1.7b), the observation noise $V = 0.1$ is much smaller than the state noise $W = 1.0$. Here, the filtered estimates react rapidly to observed data, closely tracking the measured values. In summary, lower observation noise leads to more responsive estimation, while higher observation noise results in greater smoothing and reliance on the model.

When parameters V_t and W_t are unknown, they can be estimated by maximum likelihood or integrated out in a fully Bayesian framework, for example via MCMC methods. However, the Kalman filter relies on linearity and Gaussian assumptions. When these conditions are violated—for example, in models with nonlinear dynamics or heavy-tailed observation errors—its estimates may become biased or suboptimal.

For a detailed derivation of the filtering recursions and the expectations underlying the Kalman filter, see Appendix 1.4.1.

1.3.3 Smoothing and State Prediction

In addition to filtering, DLMs also support *state prediction* and *smoothing*. While filtering produces online estimates of the latent state θ_t using data up to time t, smoothing is solved using backward-recursive methods that estimate the entire sequence of states $\theta_{1:t}$ given $y_{1:t}$ (retrospectively). State prediction is addressed by computing the k-step-ahead predictive distribution for future states and observations at time $t + k$. It quantifies uncertainty about future latent states and observations, which increases as the forecast horizon lengthens. Therefore, in state prediction the primary objective is to estimate the conditional mean—known as the optimal k-step-ahead forecast or *the forecast function*—which minimizes the conditional expected square prediction error.

These tasks differ by the amount of information available when estimating the latent state. Specifically:

- *Filtering*: $p(\theta_t \mid y_{1:t})$ (online estimation).
- *Smoothing*: $p(\theta_s \mid y_{1:t})$ (retrospective estimation).
- *Prediction*: $p(\theta_{t+k} \mid y_{1:t})$ for $k \geq 1$ (forecasting).

1.3.4 Smoothing

Smoothing provides the posterior distribution of past states given all observations up to T. In other words, it retrospectively reconstructs each state θ_s for $s < T$ by leveraging all the information contained in the complete observation sequence $y_{1:T}$. This often leads to estimates with reduced uncertainty compared to filtering, because it weights future data in addition to past and current observations.

Under Gaussian assumptions, smoothing recursions are implemented through two phases:

1. **Forward pass (Kalman filter):** compute all $m_t, C_t, a_{t+1}, R_{t+1}$ for $t = 1, \ldots, T$;
2. **Backward pass:** via the *Rauch–Tung–Striebel* (RTS) smoother, an algorithm that performs a backward recursion that refines the state estimates computed by the Kalman filter by incorporating later data.

Once the forward quantities are available (Section 1.3.2), the RTS backward recursion refines the estimate of the latent state by comparing the model's one-step prediction with the smoothed estimate at the next time step. Using the standard notation

$$m_{t|t} = \mathbb{E}[\theta_t \mid y_{1:t}], \quad C_{t|t} = \mathrm{Var}(\theta_t \mid y_{1:t}), \quad a_{t+1|t} = G_t m_{t|t}, \quad R_{t+1|t} = G_t C_{t|t} G_t^\top + W_t,$$

the RTS recursions for $t = T-1, \ldots, 1$ are

$$J_t = C_{t|t}\, G_t^\top\, R_{t+1|t}^{-1} \qquad \text{(smoothing gain)}, \tag{1.27}$$

$$m_{t|T} = m_{t|t} + J_t\big(m_{t+1|T} - a_{t+1|t}\big), \tag{1.28}$$

$$C_{t|T} = C_{t|t} + J_t\big(C_{t+1|T} - R_{t+1|t}\big) J_t^\top, \tag{1.29}$$

initialized at $m_{T|T}$ and $C_{T|T}$. These equations yield posterior estimates that integrate both past and future observations, refining the filtered trajectory based on information that was not yet available at the time of estimation. The adjusted mean $m_{t|T}$ corrects the filtered estimate $m_{t|t}$ via the term $(m_{t+1|T} - a_{t+1|t})$, aligning it with a trajectory more consistent with the complete dataset. The covariance $C_{t|T}$ is typically smaller than $C_{t|t}$, reflecting reduced uncertainty due to the incorporation of future data.

The practical advantage of smoothing is best appreciated visually: Figure 1.8 contrasts smoothed and filtered estimates along with their associated uncertainty.

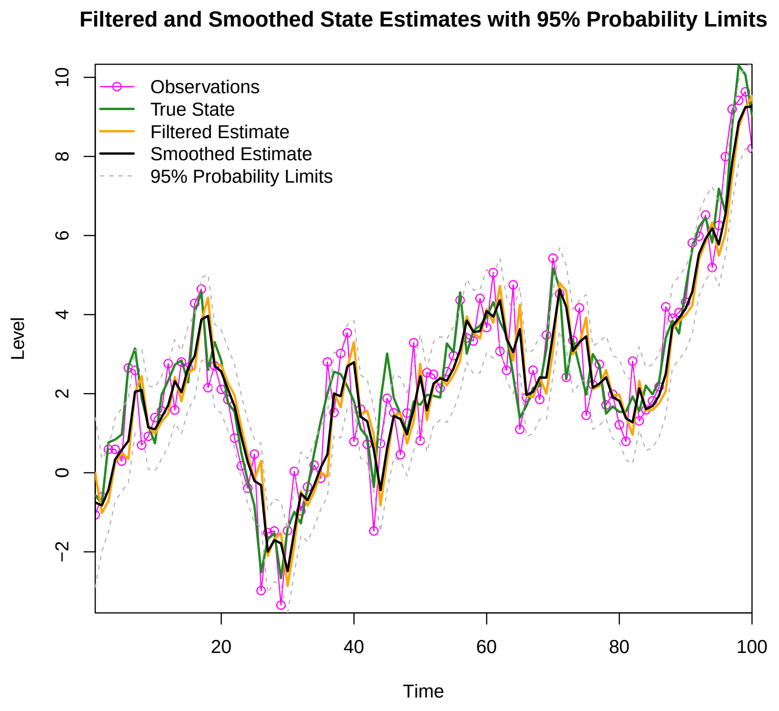


Figure 1.8: Comparison of Filtered and Smoothed State Estimates with 95% Probability Bands

The addition of the 95% probability intervals, or credible intervals, around the smoothed estimate serves two purposes. First, it quantifies the uncertainty about the latent state, even after incorporating all available observations up to time T. Second, it enables a visual assessment of estimation precision: narrow intervals indicate high confidence in the smoothed estimate, while wider intervals signal higher uncertainty, typically observed at the beginning and end of the series. At the beginning, the intervals are wider because little information has been accumulated; in the middle, they narrow as the smoother exploits all data; toward the end, they widen slightly since no future data are available beyond the final time point. As a result, the smoothed estimates coincide with the filtered estimates, which rely only on past information.

Overall, this pattern is typical of smoothing in state-space models such as the Random Walk Plus Noise: uncertainty decreases as more observations are incorporated, yet it can-

not fully vanish at the edges of the observation window. The smoothed mean (black line) closely tracks the true state trajectory (green), illustrating the effectiveness of retrospective inference in recovering latent dynamics. This visualization highlights how smoothing not only produces point estimates but also provides probabilistic information that helps interpret the reliability of the inferred trajectory.
At the sample boundary the smoothed and filtered quantities coincide. In fact, the RTS backward pass is initialized at T with $m_{T|T}$ and $C_{T|T}$, so

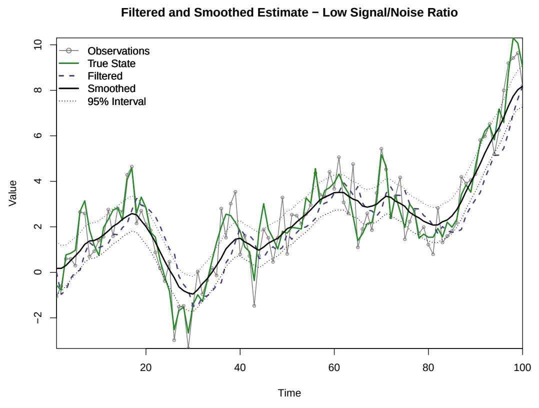
$$m_{T|T} = m_{T|T}, \qquad C_{T|T} = C_{T|T},$$

and, more generally, $m_{s|T} \to m_{T|T}$ and $C_{s|T} \to C_{T|T}$ as $s \uparrow T$. This formalizes the widening of uncertainty near the end of the sample and explains why smoothed intervals approach the filtered ones at T.

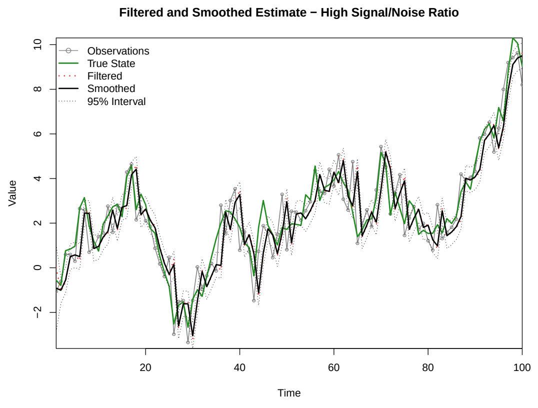
Sensitivity to Signal-to-Noise Ratio

To further illustrate how smoothing reacts to different assumptions about the relative magnitude of the observation noise and the latent state variance, we recall the same example developed earlier in the context of filtering. Figures 1.9a and 1.9b display smoothed estimates for the same data simulated under the Random Walk Plus Noise model, fitted with two alternative parameter configurations:

- **Low Signal-to-Noise Ratio:** $V = 1.0$, $W = 0.1$. Most variability is attributed to observation noise. Here, the smoothed trajectory is smoother, with narrower uncertainty bands.
- **High Signal-to-Noise Ratio:** $V = 0.1$, $W = 1.0$. Most variability is attributed to the latent state. Here, the smoothed trajectory exhibits higher variability, tracking the latent process more closely.



(a) Low SNR (large V, small W).



(b) High SNR (small V, large W).

Figure 1.9: Comparison between low and high SNR.

Comparing the two cases shows that when W is lower, the estimates are smoother and attribute most deviations to observation noise. Conversely, a higher state variance W makes the smoother more adaptive to short-term changes in the data, resulting in more variable estimated trajectories.

1.3.5 State Prediction

Prediction provides the distribution of future latent states and observations, conditional on the information available up to time t. In a dynamic linear model, this is called k-step-ahead prediction, and plays a central role in forecasting applications. Unlike filtering and smoothing, which reconstruct the past and current latent states, prediction is concerned with forecasting how the system will evolve over time.

Recall that the model's Markov structure implies that all the information about the future contained in the data $y_{1:t}$ flows through the filtering distribution of the last latent state θ_t. This dependence can be summarized visually as follows:

$$\begin{tikzcd}
\theta_t \arrow[r] \arrow[d] & \theta_{t+1} \arrow[r] \arrow[d] & \cdots \arrow[r] & \theta_{t+k} \arrow[d] \\
Y_{1:t} & Y_{t+1} & & Y_{t+k}
\end{tikzcd}$$

Figure 1.10: Flow of information from past data to future predictions

This diagram shows that $y_{1:t}$ provides information about θ_t, which drives the future evolution of the state and of future observations Y_{t+k}. As k increases, more uncertainty enters the system, and forecasts become less precise.

Having estimated the latent state up to time t, the Dynamic Linear Model framework enables k-step-ahead forecasting of both latent states and observations. This forecasting recursion proceeds by projecting the current posterior moments (m_t, C_t) forward in time using the system dynamics. Let:

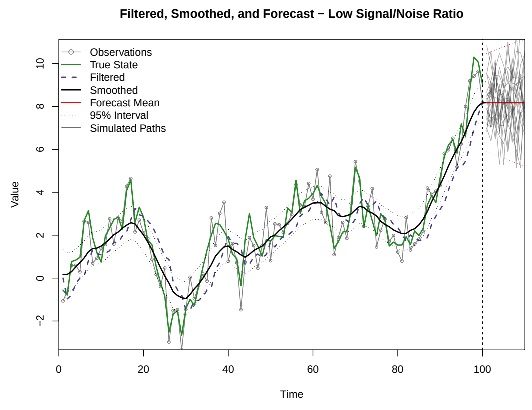
$$a_t(0) = m_t, \qquad R_t(0) = C_t.$$

Then, for any horizon $k \geq 1$, the forecast distributions are recursively defined as follows:

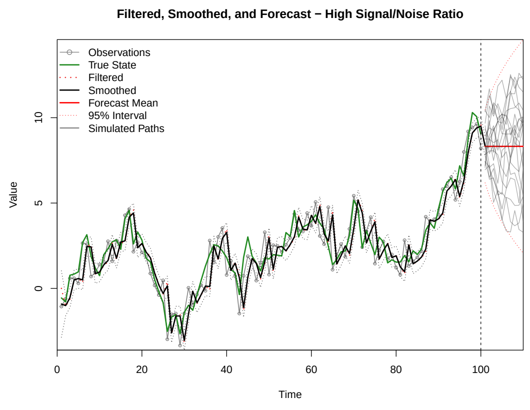
$$\begin{aligned}
a_t(k) &= \mathbb{E}\big[\theta_{t+k} \mid y_{1:t}\big] = G_{t+k}\, a_t(k-1) && \text{(Forecasted state mean)}\\
R_t(k) &= \mathrm{Var}\big[\theta_{t+k} \mid y_{1:t}\big] = G_{t+k}\, R_t(k-1)\, G_{t+k}^\top + W_{t+k} && \text{(Forecasted state variance)}\\
f_t(k) &= \mathbb{E}\big[y_{t+k} \mid y_{1:t}\big] = F_{t+k}^\top a_t(k) && \text{(Forecasted observation mean)}\\
Q_t(k) &= \mathrm{Var}\big[y_{t+k} \mid y_{1:t}\big] = F_{t+k}^\top R_t(k)\, F_{t+k} + V_{t+k} && \text{(Forecasted observation variance)}
\end{aligned}$$

A detailed proof of these recursions is reported in Appendix 1.4.1.

Illustration of Prediction Intervals Figures 1.11a and 1.11b show the forecast means and the corresponding 95% probability intervals for different horizons k, comparing the low and high signal-to-noise ratio settings. These plots highlight how forecast uncertainty increases over time and how predictions gradually move away from the last observed values, especially under higher latent process variability.



(a)



(b)

Figure 1.11: Comparison of forecasting under low SNR (a) and high SNR (b).

Comparing the two cases illustrates how the relative size of the state variance W and the observation noise V affects forecast behavior. Under low signal-to-noise (W small relative to V), most variability is attributed to measurement error, so forecasts remain closer to the smoother and to the last observations. In contrast, under high signal-to-noise (W large relative to V), the model assumes the latent process itself is highly variable. Consequently, forecasts diverge more quickly from the last observed values, and the uncertainty bands grow rapidly as the prediction horizon increases. This pattern is typical in Random Walk Plus Noise models, where uncertainty accumulates over time as

$$R_t(k) = R_t(k-1) + W,$$

when $G = 1$ and W is constant. In the univariate observation case with $F = 1$, the k-ahead predictive variance for the observation is

$$Q_t(k) = R_t(k) + V.$$

This property makes DLMs especially valuable for quantifying the range of plausible future scenarios. They are particularly useful in applications such as time series forecasting and risk assessment.

In a *Random Walk Plus Noise* (RWPN) model:

$$Y_t = \theta_t + v_t, \quad v_t \sim \mathcal{N}(0, V),$$

$$\theta_t = \theta_{t-1} + \omega_t, \quad \omega_t \sim \mathcal{N}(0, W),$$

Forecasts at short horizons are strongly anchored to the last observations, since the measurement noise v_t still contributes information about the latent state. However, as the prediction horizon k increases, the influence of past observations decays because no new data are incorporated, while the process noise ω_t accumulates step by step. Consequently, the forecast variance grows linearly over time,

$$\mathrm{Var}\big(\theta_{t+k} \mid y_{1:t}\big) = R_t(k) = R_t(k-1) + W,$$

(where $G = 1$ and W is constant), whereas the forecast mean remains fixed at the last estimated level.

This implies that, in the limit as $k \to \infty$, the process evolves as an unobserved random walk whose uncertainty increases without bound. In practical terms, for sufficiently long prediction horizons, the RWPN model behaves equivalently to a random walk.

With the foundations of state-space modeling and the core inferential procedures of filtering, smoothing, and prediction in place, the discussion now turns to model specifications that incorporate structured latent components—such as deterministic trends, seasonal effects, and external covariates—to enhance interpretability and adaptivity in time series analysis.

Appendix

1.4 Formal Derivations and Additional Remarks

This appendix gathers the formal derivations and proofs underlying the recursive equations used in Dynamic Linear Models (DLMs).

1.4.1 Derivation of the Kalman Filter Recursions

The derivation expands the filtering recursions in Section 1.3.2 of Chapter 1, showing how they follow from the state–space structure under Gaussian assumptions. All expectations and variances are assumed to exist; arguments use the law of iterated expectations and the law of total variance.

Conditional Independence and Factorization

We recall the probabilistic identities that build the filtering distribution step by step. **(i) Prior prediction for the state.** The goal is $p(\theta_t \mid I_{t-1})$. Marginalizing over θ_{t-1},

$$p(\theta_t \mid I_{t-1}) = \int p(\theta_t, \theta_{t-1} \mid I_{t-1})\, d\theta_{t-1} = \int p(\theta_t \mid \theta_{t-1}, I_{t-1})\, p(\theta_{t-1} \mid I_{t-1})\, d\theta_{t-1}.$$

By the Markov property, $p(\theta_t \mid \theta_{t-1}, I_{t-1}) = p(\theta_t \mid \theta_{t-1})$, hence

$$p(\theta_t \mid I_{t-1}) = \int p(\theta_t \mid \theta_{t-1})\, p(\theta_{t-1} \mid I_{t-1})\, d\theta_{t-1}.$$

(ii) Prior prediction for the observation. Because y_t is conditionally independent of I_{t-1} given θ_t,

$$p(y_t \mid I_{t-1}) = \int p(y_t, \theta_t \mid I_{t-1})\, d\theta_t = \int p(y_t \mid \theta_t)\, p(\theta_t \mid I_{t-1})\, d\theta_t.$$

(iii) Posterior update. Bayes' rule gives

$$p(\theta_t \mid y_{1:t}) = \frac{p(\theta_t \mid y_{1:t-1})\, p(y_t \mid \theta_t)}{p(y_t \mid y_{1:t-1})}.$$

Since $p(\theta_t \mid y_{1:t-1})$ and $p(y_t \mid \theta_t)$ are Gaussian, the product is proportional to a Gaussian in θ_t with parameters given by the Kalman update below. These factorizations show how predictive distributions arise by integrating out the latent state.

Notation Recap

Consider the DLM

$$Y_t = F_t^\top \theta_t + \varepsilon_t, \quad \varepsilon_t \sim \mathcal{N}(0, V_t), \qquad \theta_t = G_t \theta_{t-1} + \omega_t, \quad \omega_t \sim \mathcal{N}(0, W_t),$$

with filtering output at time $t-1$ given by $\theta_{t-1} \mid y_{1:t-1} \sim \mathcal{N}(m_{t-1}, C_{t-1})$. We derive

$$a_t = \mathbb{E}[\theta_t \mid y_{1:t-1}], \quad R_t = \mathrm{Var}(\theta_t \mid y_{1:t-1}), \quad f_t = \mathbb{E}[Y_t \mid y_{1:t-1}], \quad Q_t = \mathrm{Var}(Y_t \mid y_{1:t-1}),$$

and the posterior update

$$m_t = a_t + K_t(y_t - f_t), \qquad C_t = R_t - K_t Q_t K_t^\top, \quad \text{with } K_t = R_t F_t (Q_t)^{-1}.$$

Prior Prediction for the State

From the state equation,

$$\theta_t = G_t \theta_{t-1} + \omega_t,$$

the law of iterated expectations yields

$$\mathbb{E}[\theta_t \mid y_{1:t-1}] = G_t \mathbb{E}[\theta_{t-1} \mid y_{1:t-1}] + \mathbb{E}[\omega_t] = G_t m_{t-1},$$

since $\mathbb{E}[\omega_t] = 0$. For the variance,

$$\mathrm{Var}(\theta_t \mid y_{1:t-1}) = \mathrm{Var}\big(G_t \theta_{t-1} \mid y_{1:t-1}\big) + \mathrm{Var}(\omega_t) = G_t C_{t-1} G_t^\top + W_t.$$

Thus

$$a_t = G_t m_{t-1}, \qquad R_t = G_t C_{t-1} G_t^\top + W_t.$$

Prior Prediction for the Observation

Given $Y_t = F_t^\top \theta_t + \varepsilon_t$,

$$f_t = \mathbb{E}[Y_t \mid y_{1:t-1}] = F_t^\top a_t, \qquad Q_t = \mathrm{Var}(Y_t \mid y_{1:t-1}) = F_t^\top R_t F_t + V_t.$$

Posterior Update

By Gaussian conjugacy,

$$\theta_t \mid y_{1:t} \sim \mathcal{N}(m_t, C_t), \qquad m_t = a_t + R_t F_t (Q_t)^{-1}(y_t - f_t), \qquad C_t = R_t - R_t F_t (Q_t)^{-1} F_t^\top R_t.$$

This completes the derivation of the Kalman filter recursion.

Forecasting Recursion (k-step-ahead Prediction)

Goal Starting from $\theta_t \mid y_{1:t} \sim \mathcal{N}(m_t, C_t)$, show that

$$\begin{aligned} a_t(k) &= G_{t+k}\, a_t(k-1), \\ R_t(k) &= G_{t+k}\, R_t(k-1)\, G_{t+k}^\top + W_{t+k}, \\ f_t(k) &= F_{t+k}^\top a_t(k), \\ Q_t(k) &= F_{t+k}^\top R_t(k)\, F_{t+k} + V_{t+k}. \end{aligned}$$

Proof The result for $k = 1$ is immediate. For $k \geq 1$,

$$a_t(k) = \mathbb{E}[\theta_{t+k} \mid y_{1:t}] = \mathbb{E}\Big[\mathbb{E}[\theta_{t+k} \mid y_{1:t}, \theta_{t+k-1}] \,\Big|\, y_{1:t}\Big].$$

Because $\theta_{t+k} = G_{t+k}\, \theta_{t+k-1} + \omega_{t+k}$,

$$\mathbb{E}[\theta_{t+k} \mid y_{1:t}, \theta_{t+k-1}] = G_{t+k}\, \theta_{t+k-1},$$

hence $a_t(k) = G_{t+k}\, a_t(k-1)$.

For the variance, by the law of total variance,

$$R_t(k) = \mathrm{Var}(\theta_{t+k} \mid y_{1:t}) = \mathrm{Var}(G_{t+k}\, \theta_{t+k-1} \mid y_{1:t}) + \mathbb{E}[W_{t+k}] = G_{t+k}\, R_t(k-1)\, G_{t+k}^\top + W_{t+k}.$$

For the predictive observation moments, since $\mathbb{E}[y_{t+k} \mid \theta_{t+k}] = F_{t+k}\theta_{t+k}$ and $\mathrm{Var}(y_{t+k} \mid \theta_{t+k}) = V_{t+k}$

$$f_t(k) = F_{t+k}\, a_t(k), \qquad Q_t(k) = F_{t+k}\, R_t(k)\, F_{t+k}^\top + V_{t+k}.$$

1.4.2 Additional Remarks

Interpretation of the Kalman Gain The *Kalman gain* $K_t = R_t\, F_t\, (Q_t)^{-1}$ controls the weight of the forecast error $y_t - f_t$ relative to the prior prediction. Small V_t (relative to R_t) increases K_t, giving more weight to the new observation. In mean–update form,

$$m_t = a_t + K_t\, (y_t - f_t),$$

the posterior mean blends prior and data through K_t (estimation–correction form). The covariance update

$$C_t = R_t - K_t\, Q_t\, K_t^\top$$

shows the reduction in uncertainty after incorporating y_t.

Interpretation of the Smoothing Gain The *smoothing gain* $J_s = C_s\, G^\top\, R_{s+1}^{-1}$ plays the backward–time analogue of the Kalman gain. It scales the discrepancy between the smoothed future state and its one–step prediction,

$$m_s^s = m_s + J_s\,(m_{s+1}^s - a_{s+1}),$$

Here, m_s is the filtered mean at time s, a_{s+1} the one–step-ahead predicted mean for $s+1$, and m_{s+1}^s the smoothed mean at $s+1$. The smoothing gain J_s quantifies how informative the next state is about the current one relative to the uncertainty in the prior prediction: when R_{s+1} is large (high uncertainty about the predicted next state), J_s is smaller and the adjustment is conservative; when R_{s+1} is small (high confidence in the predicted next state), J_s is larger and the correction is stronger. In short, J_s governs the backward flow of information that reduces uncertainty in past states.

Chapter 2

Computational Techniques

This chapter provides an overview of the computational methods that enable estimation, inference, and prediction in the context of Dynamic Linear Models (DLMs), as introduced in Chapter 1. These models extend classical regression by allowing parameters to evolve stochastically over time, thus capturing structural changes, latent dynamics, and time-varying uncertainty in time series data.

The standard formulation of DLMs relies on assumptions of linearity, Gaussianity, and variances treated as known. However, empirical data often challenge these assumptions: volatility may change over time, observation processes may be incomplete, and dynamics may deviate from Gaussian or linear forms. These complexities necessitate computational strategies that can flexibly accommodate uncertainty and update model estimates as new data arrive.

Both Frequentist and Bayesian approaches offer frameworks for state estimation and parameter learning in DLMs. Frequentist methods provide efficient recursive algorithms under fixed parameter assumptions, emphasizing likelihood-based inference and optimization. Bayesian approaches, by contrast, incorporate prior knowledge and fully characterize uncertainty through posterior distributions, often requiring simulation-based tools such as MCMC or particle filtering to perform inference in more complex models.

This chapter bridges theory and implementation, highlighting how assumptions about model structure, prior information, and computational constraints guide the selection of algorithms. Particular attention is paid to variance specification strategies—such as discount factors, conjugate priors, and hierarchical models—that influence model flexibility and computational tractability.

Extensions beyond the standard framework address situations involving regime switching, time-varying volatility, deviations from Gaussianity, nonlinear state dynamics, and incomplete data. These scenarios require advanced computational techniques capable of handling structural uncertainty and irregularities in the data-generating process, including methods based on approximation, simulation, and adaptive estimation.

Through this computational lens, both Frequentist and Bayesian perspectives converge in addressing the challenges of dynamic inference, offering complementary tools for robust and interpretable time series modeling.

Frequentist and Bayesian Estimation

Dynamic Linear Models can be analyzed through two principal inferential frameworks: the frequentist and the Bayesian approach. These paradigms differ in their interpretation of probability, their treatment of uncertainty, and the mechanisms by which information is incorporated into inference.

The frequentist perspective treats model parameters as fixed but unknown quantities and bases inference on the likelihood function and sampling properties. The Bayesian approach, in contrast, models parameters as random variables and updates prior beliefs using Bayes' theorem, yielding posterior distributions that fully characterize uncertainty. This divergence leads to distinct estimation strategies and computational techniques.

The choice between these approaches is not merely philosophical. It reflects deeper assumptions about the nature of information and the role of prior knowledge, as well as practical considerations related to model complexity, data structure, and computational feasibility. In particular, dynamic models pose specific challenges—such as time dependence, latent structures, and non-stationary processes—that require inference to be both recursive and adaptive. When information arrives sequentially and the underlying process evolves over time, the contrast between frequentist and Bayesian paradigms becomes especially relevant.

Each framework brings complementary strengths in addressing the demands of state estimation, learning, and prediction in time series analysis.

2.1 Frequentist Estimation

Frequentist inference is not a monolithic school of thought, but rather encompasses a variety of philosophical approaches that share a common premise: parameters are fixed but unknown, and probability reflects the long-run frequency of repeatable events. Uncertainty is attributed to randomness in the data, not to the model parameters. Inference is therefore conditional on the specified model and the sampling mechanism. Within this framework, probability statements refer to hypothetical repetitions of the data-generating process, and inferential procedures are evaluated based on their long-run error rates.

Differences within frequentist thought arise in the emphasis placed on aspects such as the role of hypothetical repetitions, experimental design, and the control of inductive errors. This diversity influences both the development and application of frequentist tools.

The origins of frequentist methods can be traced to the foundational work of Fisher, who introduced maximum likelihood estimation as a general approach to parameter inference. Subsequent developments by Neyman and Pearson led to the formalization of hypothesis testing and the construction of confidence intervals. In time series analysis and state-space models, these principles evolved into recursive estimation techniques, most notably the Kalman filter, which offers optimal linear state estimation under Gaussian and linear assumptions. Owing to its efficiency and analytical tractability, the Kalman filter remains widely used in fields such as engineering, econometrics, and real-time systems.

Frequentist inference typically follows a sequential process, each step conditions on the results of the preceding one, as illustrated in Figure 2.1.:

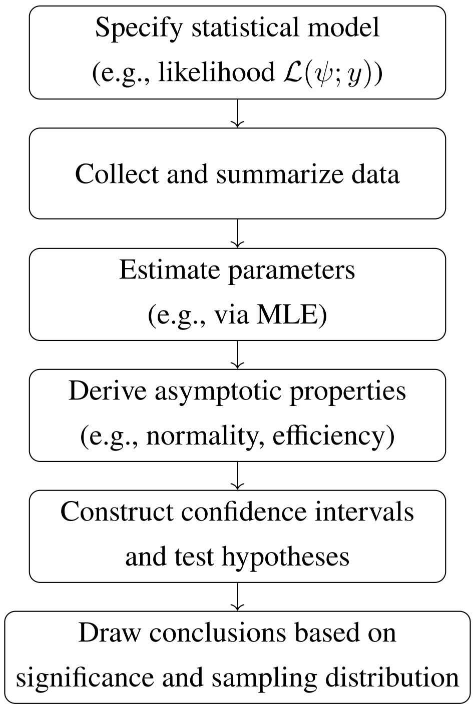

Specify statistical model
(e.g., likelihood $\mathcal{L}(\psi; y)$)
Collect and summarize data
Estimate parameters
(e.g., via MLE)
Derive asymptotic properties
(e.g., normality, efficiency)
Construct confidence intervals
and test hypotheses
Draw conclusions based on
significance and sampling distribution

Figure 2.1: Core Steps in Frequentist Inference

2.1.1 Model Specification and Likelihood Construction in DLMs

Frequentist modeling begins by specifying the probabilistic structure of the observed data. In a Dynamic Linear Model (DLM),

$$Y_t = F_t^\top \theta_t + \varepsilon_t, \qquad \varepsilon_t \sim \mathcal{N}(0, V_t), \tag{2.1}$$

$$\theta_t = G_t \theta_{t-1} + \omega_t, \qquad \omega_t \sim \mathcal{N}(0, W_t), \tag{2.2}$$

with initial state $\theta_0 \sim \mathcal{N}(m_0, C_0)$. Here Y_t is the observed value, θ_t the latent state, F_t the design vector, and G_t the system matrix; ε_t and ω_t are independent, centered Gaussian noises with (possibly time-varying) variances V_t and W_t. Under these linear–Gaussian assumptions all conditional and marginal distributions remain Gaussian, enabling exact recursive inference via the Kalman filter.

In likelihood-based inference, the likelihood function quantifies the probability of the observed data under the model for a given set of parameters. In DLMs, the marginal likelihood of the data $\mathbf{y}_{1:T} = (y_1, \ldots, y_T)$ is obtained by integrating over all latent states:

$$L(\psi) \;=\; p(y_{1:T} \mid \psi) \;=\; \prod_{t=1}^{T} p(y_t \mid y_{1:t-1}, \psi),$$

The vector $\psi = \{V_t, W_t\}$ denotes the set of unknown variance parameters to be estimated and, under Gaussianity,

$$p(y_t \mid y_{1:t-1}, \psi) \;=\; \mathcal{N}(f_t,\, Q_t),$$

with forecast mean f_t and variance Q_t produced by the Kalman filter. Taking logs yields

$$\log L(\psi) \;=\; -\frac{1}{2}\sum_{t=1}^{T}\Big[\log(2\pi Q_t) + \tfrac{(y_t - f_t)^2}{Q_t}\Big].$$

This recursive formulation is numerically stable and scalable, and it underpins maximum likelihood estimation and information-criteria model comparison.

2.1.2 Data Collection and Summarization

A critical step prior to estimation involves examining the statistical properties of the observed series. This diagnostic phase ensures alignment between the chosen model structure and the empirical properties of the data. In particular, the assumptions of stationarity, independence, and normality must be critically assessed and validated.

Descriptive statistics

Preliminary analysis of time series data often involves the computation of summary statistics and dependence measures. Among the most commonly used are the sample mean and sample variance, which help detect potential shifts in level or dispersion over time. To explore temporal dependence, the autocorrelation function (ACF) is employed. It measures the linear correlation between observations separated by a lag h, and is defined as:

$$\hat{\rho}(h) = \frac{\sum_{t=1}^{T-h}(y_t - \bar{y})(y_{t+h} - \bar{y})}{\sum_{t=1}^{T}(y_t - \bar{y})^2} = \frac{\hat{\gamma}(h)}{\hat{\gamma}(0)},$$

where $\hat{\gamma}(0)$ corresponds to the sample variance. A high value of $\hat{\rho}(h)$ indicates strong persistence or correlation at that specific lag. Complementing the ACF, the partial autocorrelation function (PACF) isolates the direct effect of each lag h on the current value, controlling for the influence of all shorter lags $1, \ldots, h-1$. This tool is particularly useful for identifying the appropriate autoregressive orders.

Visual inspection through ACF and PACF plots—displayed below—provides a practical means to identify significant lags and potential structures in the data. These diagnostics complement numerical summaries and guide the formulation of suitable dynamic models.

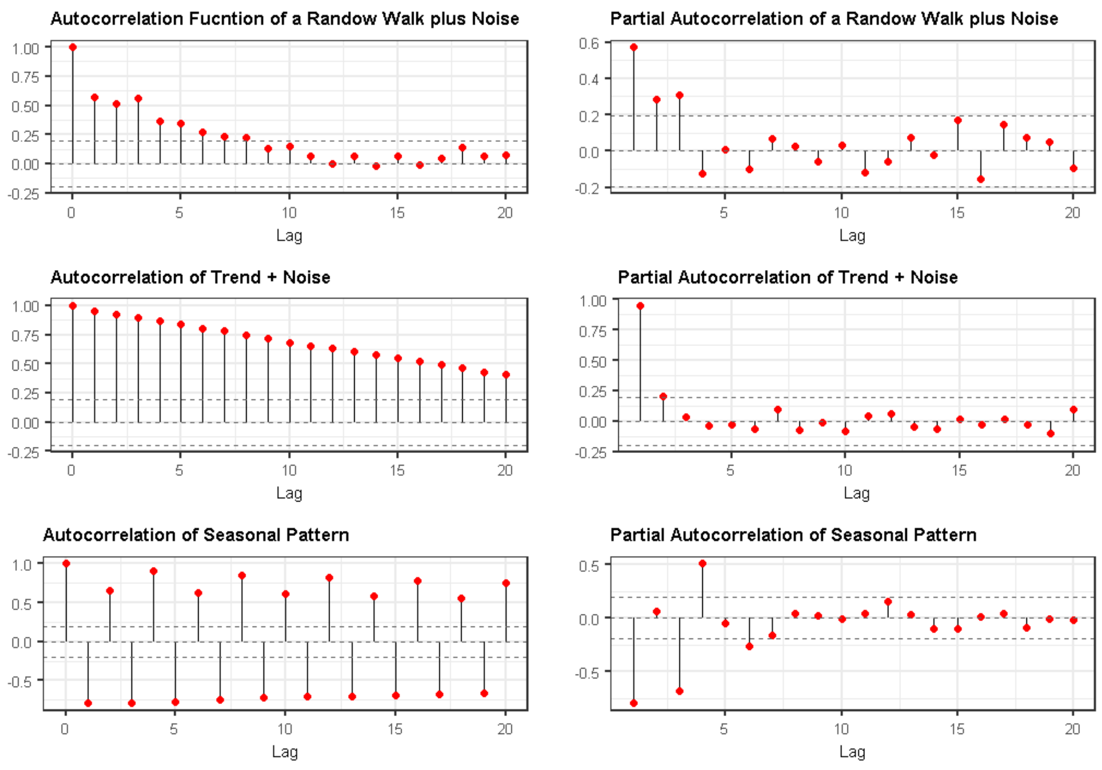


Figure 2.2: ACF and PACF

Figure 2.2 shows three stylized processes. The Random Walk Plus Noise (RWPN) exhibits a slowly decaying ACF—consistent with nonstationarity—and a pronounced lag-1 spike in the PACF. The trend-plus-noise series displays moderate persistence with a strong first-order partial autocorrelation, suggestive of deterministic drift. The seasonal series has clear ACF peaks at seasonal multiples, while the PACF lacks a comparable periodic pattern. Dashed lines denote 95% bounds under the white-noise null; correlations outside the bands are statistically significant. These diagnostics help distinguish latent components (stochastic level/trend, seasonality) before specifying a Dynamic Linear Model.

From diagnostics to remedies. The baseline Gaussian DLM yields closed-form filtering and likelihood, yet empirical series often exhibit residual serial correlation, heteroskedasticity, missing observations, regime changes, or mild nonlinearity. In a frequentist workflow, departures are handled as follows: (i) *serial dependence*—use HAC long-run variance (Newey–West) in tests and sandwich/OPG corrections for inference, or enrich the state/observation equations to absorb persistence; (ii) *heteroskedasticity and unknown/time-varying variances*—adopt quasi-maximum likelihood or EM-based updates for the variance components (V_t, W_t); (iii) *non-Gaussian or nonlinear dynamics*—use extended/unscented Kalman filtering or simulated-likelihood methods; (iv) *regime changes*—apply regime-switching filters within the same estimation pipeline. When preliminary diagnostics (ACF/PACF, unit-root and KPSS tests) reveal significant autocorrelation or heteroskedasticity, standard errors must be corrected and likelihood-based inference adapted accordingly via the tools above.

Checking stationarity

Assessing stationarity—both in mean and variance—is a foundational requirement for many time series models. Visual inspection may suggest the presence of a deterministic trend or evolving variance, but formal statistical tests provide greater objectivity.

A standard approach is the Augmented Dickey–Fuller (ADF) test, which evaluates the presence of a unit root, indicating whether the series follows a stochastic trend (e.g., a random walk) and is thus non-stationary.

Formally, the ADF test estimates the following regression:

$$\Delta y_t = \alpha + \beta t + \gamma y_{t-1} + \sum_{i=1}^{p} \delta_i \Delta y_{t-i} + \varepsilon_t,$$

where $\Delta y_t = y_t - y_{t-1}$ denotes the first difference of the series, α is a constant (drift), β allows for a deterministic linear trend, γ tests for mean reversion, and the lagged differences control for higher-order serial correlation. The error term ε_t is assumed to be white noise. The null and alternative hypotheses are:

$$\begin{cases} H_0 : \gamma = 0 & \text{(unit root; non-stationary)} \\ H_1 : \gamma < 0 & \text{(stationary process)} \end{cases}$$

The test statistic is computed as a t-ratio:

$$\tau_{\text{ADF}} = \frac{\hat{\gamma}}{\text{SE}(\hat{\gamma})}$$

which assesses how far $\hat{\gamma}$ deviates from zero in units of its standard error, under the null hypothesis. Its distribution under the null is non-standard and depends on whether the model includes a drift (α), a trend (βt), or neither. Therefore, critical values[1] must be drawn from dedicated unit root distribution tables rather than conventional t-tables.

$$\text{If} \quad \tau_{\text{ADF}} < \text{CV} \quad \Rightarrow \quad \text{Reject } H_0 \text{ (evidence of stationarity).}$$

The Phillips–Perron (PP) test provides an alternative to the ADF test for detecting unit roots. It is based on the same regression:

$$\Delta y_t = \alpha + \beta t + \gamma y_{t-1} + \varepsilon_t,$$

and tests $H_0 : \gamma = 0$ (unit root) against $H_1 : \gamma < 0$ (stationarity).

Unlike the ADF, which includes lagged differences to correct for serial correlation, the PP test uses a non-parametric adjustment to the t-statistic of $\hat{\gamma}$. This correction addresses both serial correlation and heteroskedasticity in ε_t without augmenting the regression. Serial correlation refers to temporal dependence in residuals, while heteroskedasticity denotes changing variance over time—both of which can invalidate standard inference.

The PP test statistic is adjusted as:

$$Z_t = t_\gamma - \frac{\hat{s}^2 - \hat{\sigma}^2}{2 \cdot \hat{\sigma} \cdot \sqrt{T} \cdot \hat{\tau}},$$

where t_γ is the standard t-statistic, $\hat{s}^2$ is a long-run variance estimator (accounting for both autocorrelation and heteroskedasticity), $\hat{\sigma}^2$ is the short-run variance from the regression, $\hat{\tau}$ is the standard error of $\hat{\gamma}$, and T is the sample size. In practice,

$$\hat{s}^2 \; = \; \hat{\gamma}_0 \; + \; 2 \sum_{j=1}^{L} k\!\left(\frac{j}{L+1}\right) \hat{\gamma}_j,$$

where $\hat{\gamma}_j$ are residual autocovariances and $k(\cdot)$ is a kernel (e.g., Bartlett). The short–run variance from the ADF regression is denoted by $\hat{\sigma}^2$, and $\hat{\tau}$ denotes the standard error of $\hat{\gamma}$; T denotes the sample size. Under the null, Z_t follows a non-standard distribution. As with the ADF, the rejection of H_0 implies stationarity.

The KPSS (Kwiatkowski–Phillips–Schmidt–Shin) test inverts the logic of the ADF and PP procedures. Here, the null hypothesis assumes stationarity, and rejection provides

[1]The critical value (CV) is a threshold derived from the distribution of the test statistic under the null hypothesis.

evidence for the presence of a unit root.

$$\begin{cases} H_0 : \text{The series is (level or trend) stationary} \\ H_1 : \text{The series contains a unit root (non-stationary)} \end{cases}$$

The test statistic is based on the cumulative sum of residuals e_t from an OLS regression of y_t on deterministic components (either a constant or a constant plus trend). Defining the partial sum process:

$$S_t = \sum_{i=1}^{t} e_i = \sum_{i=1}^{t} (y_i - \hat{y}_i), \qquad \text{for } t = 1, \dots, T,$$

where $\hat{y}_i$ denotes the predicted value of y_i from a regression on deterministic components (e.g., a constant or a constant and trend). The KPSS statistic is computed as:

$$\text{KPSS} = \frac{1}{T^2} \sum_{t=1}^{T} S_t^2 \Big/ \hat{\sigma}^2 \quad = \quad \frac{1}{T^2} \sum_{t=1}^{T} \left(\sum_{i=1}^{t} (y_i - \hat{y}_i) \right)^2 \Big/ \hat{\sigma}^2,$$

where $\hat{\sigma}^2$ is a consistent estimator of the long-run variance of e_t, typically calculated via a Newey–West estimator[2] to accommodate autocorrelation and heteroskedasticity.

Under the alternative hypothesis, the KPSS statistic diverges, and its distribution is non-standard. Critical values are obtained through simulation, and rejection of the null occurs when the test statistic exceeds the threshold.

In summary, the ADF and PP tests evaluate the presence of a unit root under the null, using different treatments of serial correlation. The KPSS test reverses the hypotheses by assuming stationarity under the null and employs a variance-ratio approach based on cumulative residuals. These methods are complementary: applying them jointly strengthens inference. If ADF or PP fail to reject the null while KPSS rejects stationarity, this provides stronger evidence of non-stationarity.

A recurring concern across all tests is size distortion—the tendency to overreject or underreject the null hypothesis due to poor choices of lag length (in ADF), bandwidth (in PP and KPSS), or small sample sizes. This underscores the importance of diagnostic checks and data-driven selection criteria.

To assess stationarity in the simulated series, all three tests (ADF, PP, KPSS) are

[2]The Newey–West estimator is a heteroskedasticity- and autocorrelation-consistent (HAC) estimator of the long-run variance. It adjusts the standard variance estimate by accounting for serial correlation up to a specified lag, using weighted autocovariances. This makes it suitable for time series data where residuals are not independently and identically distributed.

applied, as previously discussed. This combined approach increases diagnostic reliability and captures diverse departures from stationarity.

Table 2.1: Stationarity Test Results for Simulated Series

Series	ADF Stat.	ADF p-val	PP Stat.	PP p-val	KPSS Stat.	KPSS p-val
RWPN	-2.043	0.559	-64.818	0.010	1.075	0.010
Trend	-6.209	0.010	-84.925	0.010	2.096	0.010
Seasonal	-5.194	0.010	-104.457	0.010	0.078	0.100

Using the ADF and PP tests (which assume a unit root under the null) alongside the KPSS test (which assumes stationarity), the RWPN series shows mixed evidence: the PP rejects a unit root but both the ADF and KPSS point to non-stationarity, reflecting its near-unit-root behavior. The trend-plus-noise series also rejects a unit root under ADF and PP but fails the KPSS stationarity test, indicating non-stationarity in level driven by the deterministic trend. In contrast, the seasonal series consistently rejects a unit root in ADF and PP while the KPSS fails to reject stationarity, confirming a stable seasonal pattern around a constant mean.
Taken together, ADF/PP/KPSS provide the baseline verdict on persistence; when evidence points to structural features beyond linear–Gaussian dynamics, dedicated state–space extensions (EKF/UKF and simulation-based filters) become the appropriate diagnostic–and–modeling step.

Non-Gaussian and Nonlinear Models

Empirical time series often deviate from the linear–Gaussian assumptions that underlie standard DLMs. Nonlinear state dynamics, regime shifts, and heavy-tailed innovations are common in economic and financial data, motivating extensions that keep the state–space structure while relying on local approximations or simulation-based inference.

A widely used technique is the Extended Kalman Filter (EKF), which linearizes the nonlinear state and observation equations at each time step via a first-order Taylor expansion around the current estimate. This produces time-varying linear approximations based on the Jacobians

$$F_t = \left.\frac{\partial f(\theta_{t-1})}{\partial \theta_{t-1}}\right|_{\hat{\theta}_{t-1}}, \qquad H_t = \left.\frac{\partial h(\theta_t)}{\partial \theta_t}\right|_{\hat{\theta}_t},$$

that replace fixed system matrices within the Kalman recursion. EKF is easy to implement and effective under mild departures from linearity, but performance can deteriorate when dynamics are strongly nonlinear or errors are markedly non-Gaussian.

To mitigate the limitations of local linearization, the Unscented Kalman Filter (UKF) replaces derivatives with a deterministic set of sigma points that are propagated through the nonlinear functions, capturing posterior mean and covariance up to second-order accuracy. With a prior $\theta \sim \mathcal{N}(\hat{\theta}, P)$ and a nonlinear map $f(\cdot)$,

$$\hat{y} \approx \sum_{i=0}^{2L} w_i^{(m)} f(\chi_i), \qquad P_y \approx \sum_{i=0}^{2L} w_i^{(c)} \big(f(\chi_i) - \hat{y}\big)\big(f(\chi_i) - \hat{y}\big)^\top,$$

where $\{\chi_i\}$ are sigma points derived from $\hat{\theta}$ and P, and $w_i^{(m)}$, $w_i^{(c)}$ are the associated weights. UKF typically improves filtering in moderately nonlinear or mildly non-Gaussian settings and avoids the need for Jacobians.

In highly nonlinear or non-Gaussian environments—such as models with regime switching, stochastic volatility, or fat-tailed errors—simulation-based methods provide greater flexibility. Particle filtering and iterated filtering propagate an ensemble of latent trajectories and approximate the likelihood through Monte Carlo integration. For a parameter vector θ, particle methods deliver an estimator of the likelihood $\widehat{p}(y_{1:T} \mid \theta)$, which can be maximized to obtain

$$\hat{\theta}_{\text{MLE}} = \arg\max_{\theta} \widehat{p}(y_{1:T} \mid \theta).$$

These procedures accommodate complex dynamics while maintaining a likelihood-based focus, at the cost of increased computation, tuning choices (e.g., resampling schemes, number of particles), and potential weight degeneracy over long horizons.

These tools extend frequentist state-space modeling to cases where analytical filtering breaks down, yielding practical estimators for nonlinear and non-Gaussian time series.

Model Diagnostics

Model diagnostics assess the adequacy of a fitted Dynamic Linear Model by testing whether its assumptions hold in practice. In particular, residuals are examined to determine whether they exhibit the expected properties under correct specification: absence of serial correlation, constant variance, and Gaussian distribution. These features are essential for valid inference, especially when relying on maximum likelihood estimation or Gaussian state-space formulations. Diagnostic results inform whether the model captures the underlying structure or requires refinement.

Residual Diagnostics Once a time series model has been tentatively fitted, residual analysis is essential to assess whether the remaining errors resemble white noise. If the model has adequately captured the underlying structure, the residuals should exhibit no systematic pattern. Departures from white noise behavior may signal model misspecification—such as omitted dynamics, residual autocorrelation, structural breaks, or heteroskedasticity—and can undermine inference and predictive accuracy.

This diagnostic step is central to the Box–Jenkins methodology, which offers a structured approach for model specification, estimation, and evaluation—particularly for ARIMA models, typically expressed as $\phi(L)(1-L)^d y_t = \theta(L)\varepsilon_t$, with $\phi(L)$ and $\theta(L)$ denoting AR and MA polynomials, and d the differencing order for stationarity.

While developed for ARIMA models, the same residual-based diagnostics extend naturally to broader frameworks such as Dynamic Linear Models (DLMs). In this setting, residuals—defined as one-step-ahead forecast errors—should also resemble white noise under correct specification; any detectable structure may indicate model inadequacy.

To assess the independence of residuals, two widely used tests are the Box–Pierce and Ljung–Box statistics. Both evaluate whether autocorrelation remains in the residuals up to lag h, based on the sample autocorrelations $\hat{\rho}_k$.

The **Box–Pierce test** is defined as:

$$Q_{BP} = T \sum_{k=1}^{h} \hat{\rho}_k^2,$$

where T is the sample size. However, this test may over-reject the null hypothesis in small samples.

The **Ljung–Box test** refines the statistic with a finite-sample correction:

$$Q_{LB} = T(T+2) \sum_{k=1}^{h} \frac{\hat{\rho}_k^2}{T-k}.$$

Under the null hypothesis of no residual autocorrelation up to lag h, both Q_{BP} and Q_{LB} approximately follow a chi-squared distribution with h degrees of freedom.

These tests, along with visual tools such as residual time plots and autocorrelation function (ACF) diagnostics, serve not only to validate model assumptions but also to guide structural modifications. Rejection of the null hypothesis suggests residuals are not white noise, prompting further model refinement—such as including additional lags, re-specifying the differencing order, or modeling conditional heteroskedasticity.

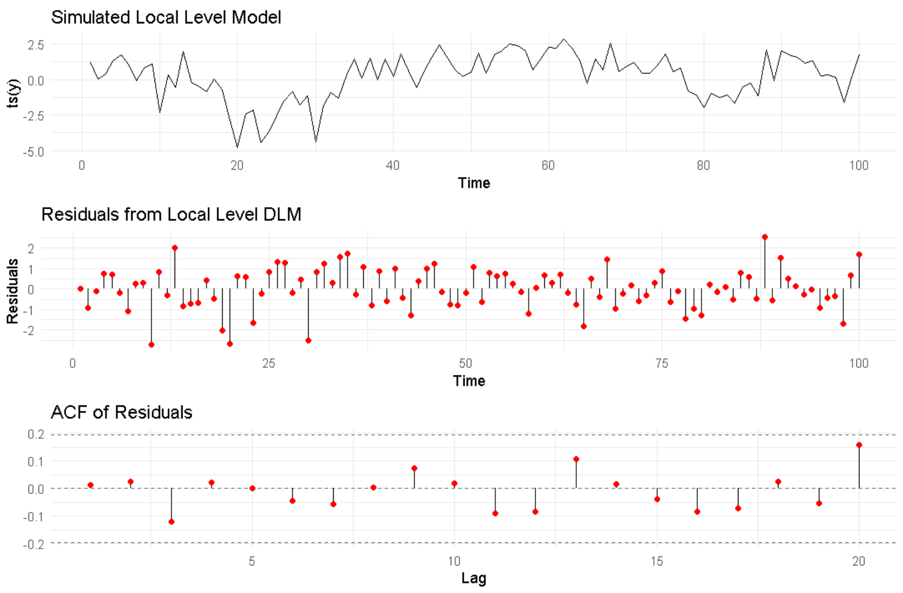


Figure 2.3: Residual analysis of the local level model.

As illustrated in Figure 2.3, the residuals from the local level model do not exhibit any visible autocorrelation structure. The top panel shows the simulated time series from the fitted local level DLM, the middle panel displays the resulting residuals, and the bottom panel shows the ACF of the residuals, where most values fall within the 95% confidence bounds, visually suggesting that the residuals resemble white noise. To formally test this hypothesis, we apply the Box–Pierce and Ljung–Box tests using 10 lags.

Table 2.2: Residual diagnostic test results for the local level DLM.

Test	**Statistic**	**Degrees of Freedom**	**p-value**
Box–Pierce	2.6915	10	0.9878
Ljung–Box	2.8903	10	0.9839

The choice of lag 10 is standard practice in time series diagnostics, balancing the detection of short-term dependencies against the risk of overfitting noise—especially in small samples. This also determines the degrees of freedom for the chi-squared distribution under the null hypothesis. The results in Table 2.2 confirm the visual findings: both p-values exceed 0.98, indicating no statistically significant evidence of autocorrelation in the residuals.

It is worth clarifying that white noise and normality are not contradictory but complementary properties. White noise implies no serial dependence and constant variance (homoskedasticity), while normality pertains to the distributional shape of the residuals. Both are essential for valid statistical inference.

In practice, residuals may be white noise without being normally distributed, and vice versa. However, in Gaussian state-space models or when using Maximum Likelihood Estimation (MLE), residuals are assumed to be both. Ideally, $\varepsilon_t \sim$ i.i.d. $\mathcal{N}(0, \sigma^2)$. Thus, residual diagnostics should jointly assess white noise behavior and normality: the former validates the model's structural adequacy, while the latter ensures reliable statistical inference.

Normality Diagnostics Assessing the normality of residuals is fundamental when statistical inference relies on Gaussian assumptions. In the context of time series models estimated via Maximum Likelihood, deviations from normality—such as skewness or excess kurtosis—can compromise the accuracy of standard errors and lead to misleading conclusions from hypothesis tests, particularly in finite samples.

The Q–Q plot of the residuals—shown in Figure 2.4—visually suggests that the residuals closely follow a normal distribution. Points lie approximately along the reference line, with only mild deviations in the tails.

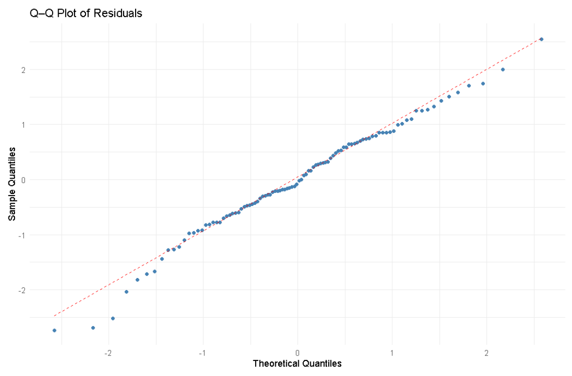


Figure 2.4: Q–Q plot of residuals from the fitted local level DLM.

This visual evidence is complemented by two standard normality tests: the Jarque–Bera (JB) and Shapiro–Wilk (SW) procedures.

The JB test is a moment-based diagnostic that evaluates whether a sample deviates from a normal distribution in terms of skewness and kurtosis[3]:

$$S = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{x_i - \bar{x}}{\hat{\sigma}}\right)^3, \qquad K = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{x_i - \bar{x}}{\hat{\sigma}}\right)^4,$$

Here, x_i represents the sample values to be tested for normality—typically observations or residuals from a fitted model, $\bar{x}$ is the sample mean and $\hat{\sigma}$ the sample standard deviation. For a normal distribution, $S = 0$ and $K = 3$. The JB test statistic jointly captures these

[3]Skewness measures asymmetry in the data distribution; kurtosis measures "tailedness" or peakedness.

deviations:

$$JB = \frac{n}{6}\left(S^2 + \frac{(K-3)^2}{4}\right),$$

which, under H_0, follows asymptotically a chi-squared distribution with 2 degrees of freedom:

$$\begin{cases} H_0 : S = 0, \quad K = 3 & \text{(normality)} \\ H_1 : S \neq 0 \text{ or } K \neq 3 & \text{(non-normality)} \end{cases}$$

A small JB and large p-value (e.g., > 0.05) indicate no evidence against normality; a large JB and small p-value suggest violations due to skewness, kurtosis, or both.

The Shapiro–Wilk (SW) test is especially powerful in small to moderate samples. Its statistic is:

$$W = \frac{\left(\sum_{i=1}^{n} a_i x_{(i)}\right)^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2},$$

where a_i are weights based on expected values of normal order statistics. The value $W \in (0, 1]$, with values closer to 1 indicating normality.

$$\begin{cases} H_0 : x_i \sim \mathcal{N}(\mu, \sigma^2) & \text{(normal)} \\ H_1 : x_i \nsim \mathcal{N}(\mu, \sigma^2) & \text{(non-normal)} \end{cases}$$

A small W and low p-value (e.g., < 0.05) suggest deviation from normality.

Formal test results on the DLM residuals are shown in Table 2.3. Both tests yield large p-values, supporting the assumption of normality and confirming the visual impression from the Q–Q plot.

Table 2.3: Normality test results on DLM residuals

Test	**Statistic**	**p-value**
Shapiro–Wilk	$W = 0.987$	0.455
Jarque–Bera	$X^2 = 2.093$	0.351

When baseline assumptions fail. Rejection in Box–Pierce/Ljung–Box or normality tests signals omitted dynamics or non-Gaussian noise. Frequentist remedies include: (i) adding lags or seasonal blocks, (ii) modeling conditional variance (time-varying V_t or W_t via QMLE/EM), (iii) switching to EKF/UKF for mild nonlinearities, and (iv) simulated-likelihood/particle methods when deviations are substantial.

2.1.3 Estimation and Asymptotic Properties

Once a Dynamic Linear Model (DLM) is specified, parameter estimation proceeds by maximizing the marginal likelihood of the observed data $y_{1:T}$, where latent states have been integrated out using recursive techniques such as the Kalman filter.

Asymptotic Properties

Maximum Likelihood Estimation (MLE) seeks parameter values that maximize the likelihood of the observed data under the assumed model. Under standard regularity conditions, maximum likelihood estimators possess a set of desirable asymptotic properties that justify their widespread use.

First, MLEs are consistent:

$$\hat{\psi}_T \xrightarrow{p} \psi_0 \quad \text{as} \quad T \to \infty,$$

meaning that the estimator converges in probability to the true parameter value ψ_0 as the sample size T increases. This implies that with more data, the estimates become increasingly reliable.

Second, MLEs are asymptotically normal. As $T \to \infty$, the scaled distribution of the estimator converges to a normal distribution centered at the true value:

$$\sqrt{T}(\hat{\psi}_T - \psi_0) \xrightarrow{d} \mathcal{N}(0, \mathcal{I}^{-1}(\psi_0)),$$

where $\mathcal{I}(\psi_0)$ is the *Fisher Information Matrix*. This matrix measures the curvature of the log-likelihood function around the true parameter and reflects the amount of information the sample provides about ψ_0. Intuitively, the steeper the curvature (i.e., the more peaked the likelihood), the more information is available to estimate the parameter precisely. Mathematically, the Fisher information is defined as the expected value of the negative second derivative (Hessian) of the log-likelihood:

$$\mathcal{I}(\psi_0) = -\mathbb{E}\left[\frac{\partial^2}{\partial\psi\,\partial\psi^\top}\log p(y_{1:T} \mid \psi)\right]\Bigg|_{\psi=\psi_0}.$$

It captures how sensitive the likelihood is to changes in the parameter: high sensitivity implies more information, leading to tighter (lower-variance) estimates.

Third, MLEs are asymptotically efficient, achieving the Cramér–Rao lower bound:

$$\mathrm{Var}(\hat{\psi}_T) \geq \mathcal{I}^{-1}(\psi_0),$$

meaning that in large samples, no unbiased estimator has a lower variance. Thus, MLEs extract the maximum amount of information from the data for estimating the parameter, making them optimal in the class of regular estimators.

Together, these properties justify the use of MLE for inference in large samples. They enable the construction of confidence intervals and hypothesis tests based on normal approximations, such as the Wald test, the likelihood ratio test, and score tests.

Maximum Likelihood Estimation in DLMs

Under the Gaussian assumption for both observation and state equations, the marginal likelihood of a DLM is analytically tractable and can be written recursively via one-step-ahead predictive distributions:

$$L(\psi) = \prod_{t=1}^{T} p(y_t \mid y_{1:t-1}, \psi),$$

$$\log L(\psi) = -\frac{1}{2} \sum_{t=1}^{T} \left[\log |Q_t| + (y_t - \hat{y}_t)^\top Q_t^{-1} (y_t - \hat{y}_t) \right] + \text{const},$$

where $\hat{y}_t = F_t^\top \hat{\theta}_{t|t-1}$ is the predictive mean, $Q_t = F_t^\top R_t F_t + V_t$ is the forecast variance and R_t denotes the predicted state covariance.

Example: Maximum Likelihood Estimation in the Local Level Model A special case of the DLM is the local level model, defined by:

$$\begin{aligned} y_t &= \theta_t + \varepsilon_t, \quad \varepsilon_t \sim \mathcal{N}(0, V), \\ \theta_t &= \theta_{t-1} + \omega_t, \quad \omega_t \sim \mathcal{N}(0, W), \end{aligned}$$

where both the design and transition matrices are equal to one, and the latent state θ_t follows a random walk.

Given a time series $\{y_1, \ldots, y_T\}$ and initial guesses for V and W, the Kalman filter generates, for each t, the one-step-ahead forecast $\hat{y}_t$ and its variance Q_t. In Gaussian state-space models, the log-likelihood is computed by aggregating forecast errors and their associated variances over time. Maximum likelihood estimation involves identifying the variance parameters V and W that maximize this log-likelihood, based on the one-step-ahead predictions and their uncertainties produced by the Kalman filter. The goal is to determine the parameter values that maximize the probability of the observed data under the model.

In a simulated dataset with $T = 100$ and true $(V, W) = (0.8, 0.3)$, a coarse grid search around initial guesses returns $\hat{V} = 0.802$, $\hat{W} = 0.16$ with $\log L \approx -157.51$. For illustrative purposes, a subset of the filtered results is reported below:

Table 2.4: One-step-ahead predictions and forecast variances from the local level model.

Time t	y_t	$\hat{y}_t$	Q_t
1	-0.66	0.00	1.80
2	0.83	-0.37	1.80
3	-2.15	0.13	1.54
$\vdots$	$\vdots$	$\vdots$	$\vdots$
100	0.63	1.62	1.46

Using the full sequence $(\hat{y}_t, Q_t)$, the log-likelihood at these estimates is

$$\log L(\hat{V}, \hat{W}) = -\tfrac{1}{2} \sum_{t=1}^{T} \left[\log(2\pi Q_t) + \tfrac{(y_t - \hat{y}_t)^2}{Q_t} \right] \approx -\tfrac{1}{2} \cdot 315.0252 = -157.5126.$$

Using the true parameters yields a log-likelihood of -157.51, whereas a subsequent call to `dlmMLE` (quasi-Newton starting from broad initials) attains a higher value of -70.23, illustrating that full optimization can substantially improve over coarse grid values. This aligns with the frequentist view that optimal estimates depend on the realized sample and need not match the data-generating parameters.

The log-likelihood surface over a grid of (V, W) values is shown below.

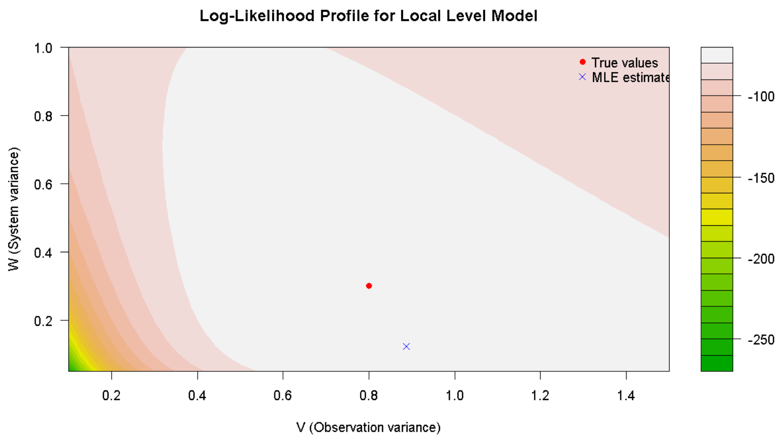


Figure 2.5: Log-likelihood surface for the local level model.

This illustrates how recursive forecasting underpins likelihood-based estimation and informs model comparison via information criteria.

Beyond the analytical expression of the log-likelihood, optimizing it requires numerical methods, especially when dealing with latent states, as in the previous example.

Since closed-form maximization is often infeasible, numerical optimization methods are employed. Two commonly used approaches are:

(i) Expectation–Maximization (EM) Algorithm. It iteratively alternates between computing the expected complete-data log-likelihood(*E-step*),

$$Q(\psi \mid \psi^{(k)}) = \mathbb{E}_{\theta|y,\psi^{(k)}} \left[\log p(y, \theta \mid \psi)\right],$$

and maximizing it with respect to ψ (*M-step*),

$$\psi^{(k+1)} = \arg\max_{\psi} Q(\psi \mid \psi^{(k)}).$$

which guarantees monotonic improvement and numerical stability.

(ii) BFGS. When the likelihood is differentiable, the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm is an efficient quasi-Newton optimization routine that iteratively updates parameter estimates using gradient and Hessian approximations:

$$\psi^{(k+1)} = \psi^{(k)} + \alpha_k H_k \nabla \log L(\psi^{(k)}),$$

with step size α_k and H_k the (updated) inverse Hessian approximation:

$$H_{k+1} = H_k + \frac{y_k y_k^\top}{y_k^\top s_k} - \frac{H_k s_k s_k^\top H_k}{s_k^\top H_k s_k},$$

where $s_k = \psi^{(k+1)} - \psi^{(k)}$ and $y_k = \nabla L(\psi^{(k+1)}) - \nabla L(\psi^{(k)})$.

BFGS is efficient for smooth likelihoods and avoids computing exact second derivatives. In practice, the asymptotic covariance can be estimated via the observed Hessian, the outer product of gradients (OPG), or their sandwich combination; under misspecification (e.g., QMLE with non-Gaussian errors), robust sandwich/OPG standard errors should be reported.

Variance Uncertainty

Assuming known and constant variances in state–space models is often too restrictive. Economic and financial series commonly display time-varying volatility, structural breaks, and heteroskedastic shocks, calling for more flexible variance components.

Within a frequentist state–space framework, unknown (possibly time-varying) variances are estimated by maximum likelihood—typically via the Expectation–Maximization (EM) algorithm—using the marginal likelihood built from one-step-ahead forecasts of the Kalman filter. Maximization delivers variance components (e.g., V_t, W_t) and transition coefficients; in practice, parsimony is imposed through structured forms such as $W_t = \omega_t I$

or low-dimensional parametrizations to aid identifiability relative to V_t.

When Gaussianity is only a working assumption, quasi-maximum likelihood (QMLE) remains consistent under mild conditions, and inference should rely on robust (sandwich/OPG) standard errors. Departures from linear–Gaussian assumptions are handled with approximate or simulation-based methods: the Extended Kalman Filter (EKF) for mild nonlinearities, and iterated filtering or simulated-likelihood schemes for pronounced nonlinearity or non-Gaussian noise. These procedures retain the recursive structure of filtering while accommodating volatility changes, structural breaks, and heteroskedasticity.

Kalman filtering, maximum likelihood, and EM. In linear–Gaussian DLMs, Kalman filtering and Rauch–Tung–Striebel (RTS) smoothing provide baseline state estimates and one–step–ahead forecasts that build the marginal likelihood. Consider

$$y_t = F_t\theta_t + v_t,\ \ v_t \sim \mathcal{N}(0, V_t), \qquad \theta_t = G_t\theta_{t-1} + w_t,\ \ w_t \sim \mathcal{N}(0, W_t),$$

with predicted state covariance $P_{t|t-1}$ given by the usual time update $P_{t|t-1} = G_t P_{t-1|t-1} G_t^\top + W_t$. Unknown parameters $\psi = (V_t, W_t, F_t, G_t, \ldots)$ are estimated by maximizing the marginal likelihood computed from the Kalman outputs:

$$\log p(y_{1:T} \mid \psi) = -\tfrac{1}{2}\sum_{t=1}^{T}\Big[\log|Q_t| + e_t^\top Q_t^{-1} e_t\Big] + \text{const},$$

$$e_t = y_t - F_t\hat{\theta}_{t|t-1},$$

$$Q_t = F_t P_{t|t-1} F_t^\top + V_t.$$

Optimization uses either quasi-Newton routines (e.g., BFGS/L-BFGS on the negative log-likelihood) or the Expectation–Maximization (EM) algorithm. In EM, the E-step applies RTS smoothing under the current ψ to obtain sufficient statistics such as $\mathbb{E}[\theta_t]$, $\mathbb{E}[\theta_t\theta_t^\top]$, and $\mathbb{E}[\theta_t\theta_{t-1}^\top]$; the M-step maximizes the expected complete–data log-likelihood to update ψ, yielding closed-form updates for variance components and, when applicable, for linear coefficients. This unified procedure treats Kalman filtering/smoothing as the computational backbone for likelihood evaluation and parameter estimation.

Strategies for Handling Unknown or Time-Varying Variances In practice, variances in both observation and state equations are rarely known and may evolve over time. Economic and financial series often display time-varying volatility, structural breaks, and heteroskedastic shocks; estimation must accommodate these features within a frequentist DLM.

A common solution is quasi-maximum likelihood (QMLE): specify a parametric form for the noise covariances, for example $W_t = \omega I$ with scalar ω, and maximize the Gaussian likelihood

$$\ell(\omega) = -\tfrac{1}{2}\sum_{t=1}^{T}\big[\log|Q_t(\omega)| + e_t^\top Q_t(\omega)^{-1}e_t\big],$$

where e_t are one-step-ahead forecast errors and $Q_t(\omega)$ their covariances from the Kalman filter. QMLE remains consistent under mild conditions even if Gaussianity is violated; inference should rely on robust (sandwich/OPG) standard errors. Because separating measurement and state noise can be delicate, identifiability benefits from structural restrictions or anchoring.

To capture gradual temporal variation without specifying volatility dynamics, a practical approach is rolling (or local) likelihood: re-estimate parameters on moving windows of length h — or with kernel weights — by

$$\hat{\psi}_{(t)} = \arg\max_{\psi}\, \ell_t^{(h)}(\psi), \qquad \ell_t^{(h)}(\psi) = \log p(y_{t-h+1:t};\psi),$$

improving robustness to breaks and allowing smooth tracking of variance changes.

Alternatively, changing variances can be modeled explicitly by augmenting the state with a stochastic-volatility process,

$$\log\sigma_t^2 = \mu + \phi\,\log\sigma_{t-1}^2 + \eta_t, \quad \eta_t \sim \mathcal{N}(0,\tau^2),\ |\phi| < 1,$$

which yields a nonlinear, non-Gaussian state-space model. Inference then uses approximate or simulation-based methods such as the Extended/Unscented Kalman filter or particle filters, and proceeds via simulated likelihood or EM routines that leverage filtering-based expectations.

These strategies extend the frequentist DLM toolkit to handle volatility clustering, structural breaks, and heteroskedasticity while retaining a likelihood-based estimation path.

Strategies for Handling Unknown or Time-Varying Variances In practice, variances in both observation and state equations are rarely known and may evolve over time. Economic and financial series often display time-varying volatility, structural breaks, and heteroskedastic shocks; estimation must accommodate these features within a frequentist DLM.

A common solution is quasi-maximum likelihood (QMLE): specify a parsimonious form for noise covariances, e.g. $W_t = \omega I$ with scalar ω, and maximize the Gaussian likelihood

$$\ell(\omega) = -\tfrac{1}{2}\sum_{t=1}^{T} \big[\log|Q_t(\omega)| + e_t^\top Q_t(\omega)^{-1} e_t\big],$$

where e_t are one-step-ahead forecast errors and $Q_t(\omega)$ their forecast-error covariances from the Kalman filter. QMLE remains consistent under mild conditions even if Gaussianity is violated; inference should rely on robust standard errors (sandwich/OPG, and HAC when residuals are autocorrelated). Because separating measurement and state noise can be delicate, identifiability benefits from structural restrictions or anchoring (e.g., tying one variance, using ratios, or low-dimensional parameterizations).

To capture gradual temporal variation without specifying volatility dynamics, a practical approach is rolling (or local) likelihood: re-estimate parameters on moving windows of length h (or with kernel weights) via

$$\hat{\psi}_{(t)} = \arg\max_{\psi} \ell_t^{(h)}(\psi), \qquad \ell_t^{(h)}(\psi) = \log p(y_{t-h+1:t};\psi),$$

improving robustness to breaks and allowing smooth tracking of variance changes; common choices set $h \approx \sqrt{T}$ or use data-driven bandwidths.

Alternatively, changing variances can be modeled explicitly by augmenting the state with a stochastic-volatility process,

$$\log\sigma_t^2 = \mu + \phi \, \log\sigma_{t-1}^2 + \eta_t, \quad \eta_t \sim \mathcal{N}(0,\tau^2),\ |\phi| < 1,$$

which yields a nonlinear, non-Gaussian state-space model. Inference then uses approximate or simulation-based methods such as the Extended/Unscented Kalman filter or particle filters, proceeding via simulated likelihood or EM routines that leverage filtering-based expectations.

These strategies extend the frequentist DLM toolkit to handle volatility clustering, structural breaks, and heteroskedasticity while retaining a likelihood-based estimation path.

Non-Gaussian or nonlinear dynamics. For models $y_t = h(\theta_t) + v_t$, $\theta_t = g(\theta_{t-1}) + w_t$, the EKF linearizes via Jacobians, the UKF propagates sigma points, and particle/iterated filtering deliver Monte Carlo likelihood estimates $\widehat{p}(y_{1:T} \mid \psi)$ that are maximized to obtain $\hat{\psi}_{\mathrm{MLE}}$.

2.1.4 Uncertainty Quantification

After point estimation, frequentist inference proceeds with quantifying uncertainty through *confidence intervals* and *hypothesis tests*. These tools formalize variability due to sampling error and support statistical decision-making under model assumptions.

A **confidence interval** is a range of plausible values for an unknown parameter. A $(1-\alpha)\%$ confidence interval for a parameter ψ is constructed such that, under repeated sampling, it would contain the true value ψ_0 approximately $(1-\alpha)\%$ of the time. When the estimator $\hat{\psi}$ is asymptotically normal, the interval is

$$\hat{\psi} \ \pm \ z_{\alpha/2} \cdot \mathrm{SE}(\hat{\psi}),$$

where $z_{\alpha/2}$ is the $(1-\alpha/2)$ standard-normal quantile, and $\mathrm{SE}(\hat{\psi})$ denotes the estimator's standard error.

When the sampling distribution is analytically intractable or asymptotics perform poorly in small samples, **bootstrap** methods offer a data-driven alternative. The idea is to approximate the sampling distribution by resampling from the observed data. Two common variants are: (i) the *parametric bootstrap*, which simulates datasets from the fitted model and re-estimates the target quantity; (ii) the *residual bootstrap* (common in regression/time-series), which resamples residuals to generate new responses and an empirical estimator distribution. Bootstrap intervals often outperform classical asymptotic ones in small samples.

Hypothesis testing assesses consistency of data with a proposed parameter value. It starts with a null H_0 versus an alternative H_1,

$$\begin{cases} H_0 : \psi = \psi_0, \\ H_1 : \psi \neq \psi_0, \end{cases}$$

and evaluates how strongly the observed data deviate from what is expected under H_0. Among classical tools, three widely used tests are the Likelihood Ratio Test (LRT), the Wald test, and the Score (Lagrange Multiplier, LM) test.

Likelihood Ratio Test (LRT). The LRT evaluates the loss in fit from imposing H_0 by contrasting restricted and unrestricted log-likelihoods:

$$\lambda \ = \ -2\big[\log L(\psi_{\text{restricted}}) - \log L(\psi_{\text{unrestricted}})\big] \ \xrightarrow{d} \ \chi^2_k,$$

where k is the number of restrictions, under standard regularity (with caveats at parameter boundaries).

Wald test. The Wald statistic standardizes the distance between $\hat{\psi}$ and ψ_0 by its variance:

$$W = (\hat{\psi} - \psi_0)^\top \left[\widehat{\mathrm{Var}}(\hat{\psi})\right]^{-1} (\hat{\psi} - \psi_0) \xrightarrow{d} \chi^2_k,$$

with $k = \dim(\psi)$. It is convenient (requires estimation only under the alternative) but can be unreliable near boundaries or under misspecification.

Score (LM) test. The LM test is based on the score (gradient) at the null:

$$LM = U(\psi_0)^\top \mathcal{I}(\psi_0)^{-1} U(\psi_0) \xrightarrow{d} \chi^2_k, \quad U(\psi) \equiv \frac{\partial \log L(\psi)}{\partial \psi},$$

and avoids re-estimation under the alternative, making it efficient when the unrestricted model is costly.

Each test has trade-offs: the LRT is powerful but computationally intensive, the Wald test is convenient but less robust, and the LM test is efficient but sensitive if the null is poorly specified.
p-values and critical values. The p-value is the probability, under H_0, of observing a statistic at least as extreme as the realized one. A small p-value (e.g., $< \alpha = 0.05$) indicates inconsistency with H_0 and leads to rejection. Equivalently, reject when the statistic exceeds the critical value from the reference distribution, e.g., $\chi^2_{k,1-\alpha}$.

To support the theoretical discussion, Table 2.5 reports a numerical illustration with maximum likelihood estimates (MLEs) of the observation and state variances (V and W), bootstrap confidence intervals, and hypothesis tests via LRT and Wald.

Table 2.5: Uncertainty Quantification: Estimation, Bootstrap CIs, and Hypothesis Testing

Parameter	**MLE**	**95% CI (Bootstrap)**	**Test H_0**	**p-value**
V (Obs. noise)	0.885	[0.606, 1.201]	LRT: $V = 0.8$	1.000
			Wald: $\log V = \log(0.8)$	0.570
W (State noise)	0.104	[0.014, 0.240]	LRT: $W = 0.3$	1.000
			Wald: $\log W = \log(0.3)$	0.061

The MLEs slightly overestimate the true values but lie within the 95% bootstrap intervals. The LRT finds no evidence against the null for either parameter ($p = 1.000$), while the Wald test flags W as borderline ($p = 0.061$), underscoring the value of multiple inference tools.

When homoskedasticity or no-serial-correlation assumptions are violated, Fisher-based standard errors may be biased or inconsistent. In such cases, **robust variance estimators** (sandwich) provide valid standard errors under wider conditions. A generic form is

$$\widehat{\mathrm{Var}}_{\text{robust}}(\hat{\psi}) = \left(\widehat{\mathcal{I}}(\hat{\psi})\right)^{-1} \widehat{\mathcal{J}}(\hat{\psi}) \left(\widehat{\mathcal{I}}(\hat{\psi})\right)^{-1},$$

where $\widehat{\mathcal{I}}(\hat{\psi})$ is the observed information and

$$\widehat{\mathcal{J}}(\hat{\psi}) = \frac{1}{T}\sum_{t=1}^{T} s_t(\hat{\psi})\, s_t(\hat{\psi})^\top, \qquad s_t(\psi) \equiv \frac{\partial}{\partial\psi} \log p\big(y_t \mid y_{1:t-1}; \psi\big),$$

is the empirical outer product of the per-period score (for state-space models the likelihood factorizes as predictive densities). For serially correlated scores, a HAC version of $\widehat{\mathcal{J}}$ can be used. This adjustment ensures valid inference even under heteroskedasticity or mild autocorrelation; the name "sandwich" refers to its matrix product form.

2.1.5 Drawing Conclusions: Information Criteria

Based on test statistics and confidence intervals, conclusions are drawn regarding the plausibility of hypotheses and the precision of parameter estimates. These inferential decisions are guided by predetermined significance levels (e.g., $\alpha = 0.05$) and rely on the sampling distribution of the estimators. It is important to note that *statistical significance*—operationalized via p-values as the probability, under H_0, of observing a statistic at least as extreme as the realized one—is frequently conflated with *substantive relevance*, which concerns practical or economic importance. The two are distinct: an effect may be statistically significant yet lack real-world impact, or be substantively meaningful but not statistically detectable in small samples. The analysis below focuses on statistical significance for validating model structure and inference, with substantive interpretation discussed where appropriate.

Beyond testing individual hypotheses, evaluating competing specifications is integral to drawing valid and parsimonious conclusions. Model selection criteria such as the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC) provide a systematic trade-off between fit and complexity. While improvements in likelihood signal better fit, additional parameters increase flexibility and risk overfitting; AIC and BIC penalize complexity to favor models that generalize.

The Akaike Information Criterion is

$$\mathrm{AIC} = -2\log L(\hat{\psi}) + 2k,$$

where $\log L(\hat{\psi})$ is the maximized log-likelihood and k is the number of free parameters. Rooted in information theory, AIC approximates (up to an additive constant) the

Kullback–Leibler divergence between the true distribution $p(y)$ and the model $q(y \mid \hat{\psi})$:

$$D_{\mathrm{KL}}(p \parallel q) = \int p(y) \log\left(\frac{p(y)}{q(y \mid \hat{\psi})}\right) dy.$$

For small samples relative to k, a finite-sample correction

$$\mathrm{AICc} \;=\; \mathrm{AIC} \;+\; \frac{2k(k+1)}{T-k-1}$$

is often recommended.

The Bayesian Information Criterion is

$$\mathrm{BIC} = -2\log L(\hat{\psi}) + k \log T,$$

where T is the sample size. BIC provides a large-sample approximation to the (negative twice) log marginal likelihood and imposes a stronger penalty as T grows.

Both criteria rank models fitted on the *same* dataset: lower values are preferred. AIC emphasizes predictive accuracy and tends to minimize out-of-sample error; BIC is consistent for selecting the true model when it is among the candidates.

These tools complement classical testing by offering a holistic framework for model comparison, especially in time series, where penalizing complexity helps prevent overfitting. In state-space models, k typically counts variance parameters (e.g., V, W) and any additional free elements in F_t, G_t, etc.; diffuse initializations contribute only constants and do not affect AIC/BIC comparisons across models fitted to the same data.

Numerical Example In the estimated local level model, the maximized log-likelihood obtained via `dlmMLE` is $\log L(\hat{\psi}) \approx -70.23$. With $k = 2$ (observation variance V and state variance W) and $T = 100$,

$$\mathrm{AIC} \;=\; -2\cdot(-70.23) + 2\cdot 2 \;=\; 140.46 + 4 \;=\; \mathbf{144.46},$$

$$\mathrm{BIC} \;=\; -2\cdot(-70.23) + (\log 100)\cdot 2 \;=\; 140.46 + 9.210 \;=\; \mathbf{149.67}.$$

These results suggest that the fitted local level DLM attains a reasonable balance between fit and parsimony; lower AIC/BIC would indicate better performance when comparing alternative specifications on the same sample.

2.1.6 Summary

Frequentist estimation in dynamic linear models (DLMs) offers a structured and efficient methodology for time series analysis, particularly in financial and economic contexts where real-time decision-making is critical. Built on maximum likelihood and Kalman filtering, this approach delivers consistent parameter estimates and rigorous uncertainty quantification via confidence intervals and hypothesis tests. In applications such as volatility modeling, interest-rate dynamics, and macroeconomic forecasting, DLMs flexibly represent latent processes and evolving structures.

Analytical tractability and computational speed make these methods well suited to high-frequency or large-scale data. Effectiveness, however, hinges on specification and sample size; inference may deteriorate under structural breaks, heteroskedasticity, or non-Gaussian features. Model selection criteria (AIC, BIC, and AICc when T is small) guide comparisons across competing specifications, though they remain sensitive to misspecification and must be applied to models fitted on the same sample.

Beyond the standard linear–Gaussian setup with known variances, frequentist practice extends naturally to anomalous settings: variance uncertainty is handled via QMLE or EM (with robust—sandwich/OPG/HAC—standard errors for valid inference), gradual change can be tracked with rolling or locally weighted likelihood, mild nonlinearity or near-Gaussian departures through EKF/UKF, and severe non-Gaussianity or complex dynamics via simulated-likelihood approaches (particle or iterated filtering) that maximize $\widehat{p}(y_{1:T} \mid \psi)$. Missing observations are accommodated by skipping measurement updates or by EM with smoothing, while structural breaks and latent heterogeneity are modeled through regime-switching DLMs estimated with the Hamilton filter and Kim smoother. These extensions preserve the plug-in, likelihood-based workflow and yield fast, tractable point estimates, though they typically propagate parameter uncertainty only partially and rely on large-sample approximations.

Taken together, these considerations motivate a complementary Bayesian route: by encoding prior information and propagating both state- and parameter-uncertainty into the full predictive distribution, Bayesian DLMs provide probability statements about future outcomes and remain effective with small samples, ragged/missing data, non-Gaussian noise, and structural breaks—offering unified uncertainty quantification where frequentist extensions prioritize speed and operational simplicity.

2.2 Bayesian Estimation

The Bayesian approach to inference in Dynamic Linear Models (DLMs) provides a probabilistic framework by modeling both parameters and latent states as random variables.

Prior distributions encode beliefs about unknown quantities before observing data, and Bayes' theorem updates those beliefs into a posterior distribution once data are available.

The philosophical roots trace back to Thomas Bayes and Pierre-Simon Laplace, while modern Bayesian practice gained momentum with computational advances. Techniques such as Gibbs sampling (Gelfand and Smith, 1990), bootstrap/particle filtering (Gordon et al., 1993), forward–filtering backward–sampling (Carter and Kohn, 1994), and Hamiltonian Monte Carlo (Neal, 2011) have transformed Bayesian estimation into a practical tool for high-dimensional and dynamic models.

Formally, given observed data $y_{1:T}$, parameters ψ, and latent states $\theta_{1:T}$, Bayesian inference centers on the posterior

$$p(\psi, \theta_{1:T} \mid y_{1:T}) \ \propto \ p(y_{1:T} \mid \theta_{1:T}, \psi)\, p(\theta_{1:T} \mid \psi)\, p(\psi),$$

where the likelihood encodes the data-generating process and the priors reflect available knowledge or assumptions. The posterior represents the full probabilistic characterization of uncertainty about both latent states and model parameters.

Bayesian reasoning views the observed $y_{1:T}$ as one realization from a potentially infinite sequence and treats parameters as random. A central concept is *exchangeability*: the joint distribution of $(y_1, \ldots, y_T)$ is invariant under permutations. By De Finetti's theorem, infinitely exchangeable sequences admit a representation as conditionally independent draws given latent structure, motivating hierarchical modeling.

These principles support a unified treatment of DLMs that incorporates prior information, handles parameter uncertainty, and facilitates robust learning through probabilistic updating. The overall workflow—modeling, prior selection, posterior inference, model checking/comparison, and forecasting—is summarized schematically in Figure 2.6.

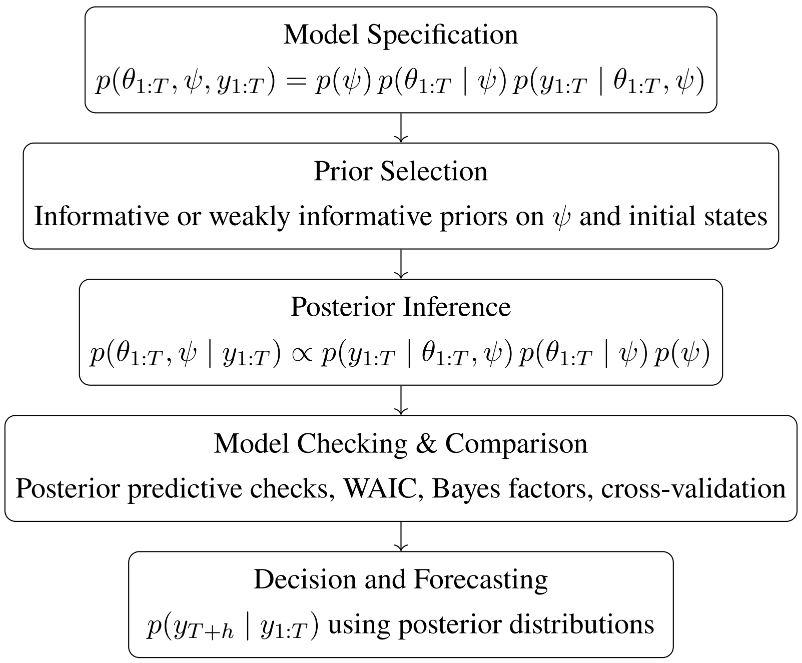

Model Specification
$p(\theta_{1:T}, \psi, y_{1:T}) = p(\psi)\, p(\theta_{1:T} \mid \psi)\, p(y_{1:T} \mid \theta_{1:T}, \psi)$
Prior Selection
Informative or weakly informative priors on ψ and initial states
Posterior Inference
$p(\theta_{1:T}, \psi \mid y_{1:T}) \propto p(y_{1:T} \mid \theta_{1:T}, \psi)\, p(\theta_{1:T} \mid \psi)\, p(\psi)$
Model Checking & Comparison
Posterior predictive checks, WAIC, Bayes factors, cross-validation
Decision and Forecasting
$p(y_{T+h} \mid y_{1:T})$ using posterior distributions

Figure 2.6: Bayesian inference workflow for Dynamic Linear Models (DLMs)

2.2.1 Model Specification

Dynamic Linear Models (DLMs), introduced in Chapter 1, provide a flexible Bayesian state–space formulation. The joint distribution over data $y_{1:T}$, latent states $\theta_{1:T}$, and parameters ψ factorizes as

$$p(y_{1:T}, \theta_{1:T}, \psi) \;=\; p(\psi)\, p(\theta_{1:T} \mid \psi)\, p(y_{1:T} \mid \theta_{1:T}, \psi).$$

The canonical linear–Gaussian specification is

$$y_t = F_t^\top \theta_t + \varepsilon_t, \quad \varepsilon_t \sim \mathcal{N}(0, V_t), \qquad \theta_t = G_t \theta_{t-1} + \omega_t, \quad \omega_t \sim \mathcal{N}(0, W_t),$$

with F_t and G_t the design and transition matrices, and V_t, W_t the observation and evolution covariances. The framework encompasses local level, local linear trend, seasonal and polynomial components; observed covariates x_t can enter via

$$y_t = x_t^\top \beta + F_t^\top \theta_t + \varepsilon_t,$$

so that variation in y_t is explained by static effects β and dynamic components θ_t. While the canonical DLM assumes linearity and Gaussian noise, extensions allow for nonlinear and non-Gaussian dynamics, often requiring MCMC or sequential Monte Carlo.

Prior Specification and Variance Modeling

Within the Bayesian framework, prior distributions are central to encoding beliefs about unknown quantities before observing the data. In practice, prior choice directly impacts posterior behavior—particularly in low-data or high-noise regimes.

A standard approach for modeling uncertainty in variances is to use conjugate priors. The Inverse-Gamma distribution is often employed for variances:

$$\sigma^2 \sim \text{Inv-Gamma}(a, b), \quad p(\sigma^2) = \frac{b^a}{\Gamma(a)}(\sigma^2)^{-a-1} \exp\left(-\frac{b}{\sigma^2}\right),$$

where a and b control the shape and scale. While analytically convenient, small values for a and b may produce heavy-tailed posteriors and unstable estimates.

To address this, weakly informative priors such as the half-Cauchy on the standard deviation are often preferred:

$$\sigma \sim \text{Cauchy}^{+}(0, A),$$

providing regularization without undue constraint, especially valuable in sparse or noisy contexts.

Hierarchical Priors for Structured Data

When multiple related time series (e.g., by region, market, or demographic segment) are modeled, hierarchical priors allow for partial pooling:

$$\theta_j \sim \mathcal{N}(\mu, \tau^2), \quad y_{ij} \sim \mathcal{N}(\theta_j, \sigma^2),$$

where group-level parameters θ_j are shrunk toward a global mean μ, balancing between complete pooling and no pooling. In DLMs this extends naturally to variance components, for instance

$$\theta_t^{(j)} = \theta_{t-1}^{(j)} + \omega_t^{(j)}, \quad \omega_t^{(j)} \sim \mathcal{N}(0, W^{(j)}), \quad W^{(j)} \sim \text{IG}(a, b) \text{ or } \sigma_W^{(j)} \sim t_\nu^{+}(0, A),$$

so that hyperpriors enable information sharing across series while accommodating local heterogeneity.

Discount Factor Specification

An alternative to full variance modeling is the use of a *discount factor* $\delta \in (0, 1]$, which defines the system variance recursively as

$$W_t \;=\; \frac{1-\delta}{\delta}\, G_t\, C_{t-1|t-1}\, G_t^{\top},$$

where $C_{t-1|t-1}$ is the posterior covariance of the state at $t-1$. Lower δ values induce more adaptivity by increasing the variance, while higher values enforce smoother state evolution; block-specific δ can tailor adaptivity to level, trend, or seasonal components.

In combination, hierarchical structures and discount factors furnish a robust and flexible apparatus for modeling time-varying dynamics across a broad range of applied domains, particularly in economics and finance, where data may be sparse, heterogeneous, or evolving.

2.2.2 Bayesian Estimation and Computation in DLMs

The Bayesian estimation of Dynamic Linear Models (DLMs) extends classical filtering by treating both latent states and model parameters as random variables with specified prior distributions. Under linear–Gaussian assumptions, the Kalman filter arises as a special case of recursive Bayesian updating and yields closed-form posteriors for the latent states conditional on observed data.

Formally, given observations $y_{1:t}$, latent states $\theta_{1:t}$, and static parameters ψ, the posterior factorizes as

$$p(\theta_{1:t}, \psi \mid y_{1:t}) \;\propto\; p(\psi)\, p(\theta_1 \mid \psi) \prod_{s=2}^{t} p(\theta_s \mid \theta_{s-1}, \psi) \prod_{s=1}^{t} p(y_s \mid \theta_s, \psi),$$

and the filtering update from $p(\theta_{t-1} \mid y_{1:t-1})$ to $p(\theta_t \mid y_{1:t})$ is obtained by marginalization. In the Gaussian case this update is analytic (Kalman filter and Rauch–Tung–Striebel smoother), providing one-step-ahead predictive distributions and state posteriors in closed form.

When normality, linearity, or conjugacy fail—or when parameter and variance uncertainty must be accounted for—the posterior becomes analytically intractable and calls for simulation-based computation. Markov Chain Monte Carlo (MCMC) and Sequential Monte Carlo (SMC) supply a unified route by learning about all unknowns (parameters, latent states, and possibly structural components) from $p(\theta_{1:T}, \psi \mid y_{1:T})$. In conditionally Gaussian DLMs, *Gibbs sampling* with Forward–Filtering Backward–Sampling (FFBS) delivers exact draws of the state trajectories given parameters, while conjugate or simple Metropolis steps update variance blocks. For more complex posteriors, *Hamiltonian Monte Carlo* (and its adaptive variant NUTS) improves mixing in high dimension by exploiting gradients of the log-posterior. In online or non-Gaussian/nonlinear settings, *SMC/particle filtering* propagates a weighted ensemble of states to approximate filtering distributions and predictive likelihoods; *particle MCMC (PMCMC)* combines SMC within MCMC to target the exact joint posterior when the likelihood is intractable but unbiasedly estimable.

These computational tools extend DLMs to variance uncertainty (unknown or time-varying V_t, W_t), non-Gaussian and nonlinear dynamics, missing observations, and regime switching. Variance components can be treated as random and inferred jointly with states; heavy tails or discrete measurements can be handled via data augmentation or particle methods; ragged/missing data are integrated probabilistically rather than imputed; and latent regimes are learned alongside states and parameters, propagating structural uncertainty into forecasts. In all cases, Bayesian updating preserves a coherent probabilistic treatment of uncertainty, while the choice among Gibbs/FFBS, HMC/NUTS, SMC, or PMCMC is guided by model structure (linearity/Gaussianity), dimensionality, and computational budget.
What follows details the main computational engines used in Bayesian DLMs—*MCMC* (including Gibbs with *FFBS* and Metropolis variants), gradient-based samplers (*HMC/NUTS*), *SMC*/particle filtering, and their hybrids (*PMCMC*)—together with practical diagnostics.

Markov Chain Monte Carlo (MCMC). MCMC methods are a core component of Bayesian computation, particularly when analytical or deterministic approximations are infeasible. The central idea is to construct a Markov chain whose stationary distribution is the target posterior, enabling inference through simulation rather than direct integration.

The procedure begins by defining a transition kernel $K(\theta' \mid \theta)$ that moves the chain from the current state θ to a proposal θ'. Stationarity with respect to the posterior $\pi(\theta)$ can be written as

$$\pi(\theta) \;=\; \int \pi(\theta')\, K(\theta \mid \theta')\, d\theta',$$

and a common sufficient condition is detailed balance,

$$\pi(\theta')\, K(\theta \mid \theta') \;=\; \pi(\theta)\, K(\theta' \mid \theta).$$

Under standard regularity conditions (irreducibility, aperiodicity, ergodicity), posterior expectations are consistently approximated by Monte Carlo averages:

$$\frac{1}{N}\sum_{i=1}^{N} f(\theta^{(i)}) \;\to\; \mathbb{E}_\pi[f(\theta)] \quad \text{as } N \to \infty.$$

In practice, early draws may lie far from high–posterior–density regions. These initial iterations (the *burn-in*) are typically discarded, with length assessed via diagnostics such as trace plots, autocorrelation, and the potential scale reduction factor $\hat{R}$; sampling efficiency is further summarized by the Effective Sample Size (ESS).

Despite sample autocorrelation, MCMC is broadly applicable and robust across hierarchical structures, latent state sequences, and nonlinear dynamics, making it a preferred tool for complex time-series models where linearization is inadequate. The MCMC family includes Metropolis–Hastings, Gibbs sampling, and Hamiltonian Monte Carlo (HMC/NUTS), each trading off mixing efficiency and implementation complexity according to model structure and computational budget. In DLMs, block Gibbs combined with Forward–Filtering Backward–Sampling (FFBS) often provides highly efficient state updates within conditionally Gaussian components.

Gibbs Sampling and Conditional Conjugacy. *Gibbs sampling*[4] is a structured MCMC method particularly effective when the joint posterior can be decomposed into tractable full conditionals. Given a parameter vector $\theta = (\theta_1, \ldots, \theta_K)$, the algorithm iteratively samples each component from its conditional distribution:

$$\theta_k^{(i+1)} \sim p\Big(\theta_k \mid \theta_{-k}^{(i)}, y\Big), \quad k = 1, \ldots, K,$$

where θ_{-k} denotes all components except θ_k (conditioning on static parameters ψ is suppressed for brevity). The resulting sequence $\{\theta^{(i)}\}$ forms a Markov chain targeting the joint posterior. In dynamic linear models, Gibbs sampling is especially useful when conditional conjugacy holds—most notably for variance components with inverse-gamma priors (or weakly-informative priors on standard deviations). Combined with latent-state updates, this blockwise scheme enables scalable posterior sampling, including hierarchical and time-varying structures.

Forward Filtering Backward Sampling (FFBS). In linear-Gaussian state-space models, the latent state sequence $\theta_{1:T}$ can be sampled exactly via *Forward Filtering Backward Sampling (FFBS)*[5]. The procedure first runs the Kalman filter forward to obtain filtering distributions $p(\theta_t \mid y_{1:t}) = \mathcal{N}(m_{t|t}, C_{t|t})$. The backward pass then samples smoothed states recursively:

$$\theta_T \sim \mathcal{N}(m_{T|T}, C_{T|T}), \qquad \theta_t \sim p(\theta_t \mid \theta_{t+1}, y_{1:T}) \quad (t = T-1, \ldots, 1).$$

[4]Introduced by Geman and Geman (1984) for image reconstruction, Gibbs sampling has become a foundational technique in Bayesian computation, especially for hierarchical models and latent variable inference.

[5]See Carter and Kohn (1994) and Frühwirth-Schnatter (1994) for formal treatments.

These conditionals are Gaussian; explicitly, with the one-step prediction $C_{t+1|t}$ and transition G_{t+1}, the smoothing gain is $J_t = C_{t|t}G_{t+1}^\top C_{t+1|t}^{-1}$ and

$$\theta_t \mid \theta_{t+1}, y_{1:T} \sim \mathcal{N}\Big(m_{t|t} + J_t(\theta_{t+1} - m_{t+1|t}),\ C_{t|t} - J_t C_{t+1|t} J_t^\top\Big).$$

FFBS complements Gibbs sampling by providing efficient block updates of the full state trajectory, fully propagating temporal uncertainty and preserving the model's dependence structure.

Metropolis–Hastings for General Posterior Simulation *The Metropolis–Hastings algorithm*[6] is a flexible and broadly applicable MCMC technique for simulating from complex posterior distributions. It is especially useful when full conditionals are unavailable or analytically intractable—e.g., with non-conjugate priors, nonlinear relationships, or constrained parameter spaces—where direct Gibbs sampling is not feasible.

The algorithm constructs a Markov chain with stationary distribution $\pi(\theta \mid y)$. Given the current state $\theta^{(i)}$, propose $\theta^* \sim q(\theta^* \mid \theta^{(i)})$ and accept with probability

$$\alpha \;=\; \min\left(1,\ \frac{\pi(\theta^* \mid y)\, q(\theta^{(i)} \mid \theta^*)}{\pi(\theta^{(i)} \mid y)\, q(\theta^* \mid \theta^{(i)})}\right),$$

which ensures detailed balance and convergence to the target. If accepted, set $\theta^{(i+1)} = \theta^*$; otherwise $\theta^{(i+1)} = \theta^{(i)}$. With *symmetric* proposals (e.g., random walk), $q(\theta^* \mid \theta^{(i)}) = q(\theta^{(i)} \mid \theta^*)$ and the ratio simplifies to $\alpha = \min\{1, \pi(\theta^* \mid y)/\pi(\theta^{(i)} \mid y)\}$.

A key advantage is that only the posterior *kernel* is needed, $\pi(\theta \mid y) \propto p(y \mid \theta)\, p(\theta)$, since normalizing constants cancel. For numerical stability, acceptance is typically computed on the log scale. Performance depends crucially on the proposal: common choices include the random-walk $q(\theta^* \mid \theta^{(i)}) = \mathcal{N}(\theta^{(i)}, \Sigma)$ and the independence sampler $q(\theta^*)$. Tuning the scale/shape of Σ (preconditioning, e.g., via a Cholesky factor aligned with posterior curvature) balances acceptance and mixing; *block* updates can mitigate strong posterior correlations. For constrained parameters (e.g., variances), reparameterizations such as $\eta = \log \sigma^2$ improve geometry and mixing. Simple adaptive schemes (with diminishing adaptation) can calibrate Σ online while preserving ergodicity.

In Bayesian Dynamic Linear Models, Metropolis–Hastings is often used to update non-conjugate parameter blocks or components appearing in nonlinear state equations. Coupled with FFBS for latent states, it enables full posterior inference under flexible modeling assumptions.

[6]Originally introduced by Metropolis et al. (1953) in the context of thermodynamic simulations and extended by Hastings (1970).

Advanced MCMC: Hamiltonian Monte Carlo and NUTS. While standard MCMC algorithms like Gibbs and Metropolis–Hastings are broadly applicable, their efficiency often deteriorates in high-dimensional spaces or when posteriors exhibit strong correlations or complex geometry. Hamiltonian Monte Carlo (HMC) mitigates these issues by introducing auxiliary momentum variables and simulating Hamiltonian dynamics; trajectories follow level sets that traverse posterior valleys efficiently. HMC requires gradients of the log-posterior (typically via automatic differentiation) and tuning of step size ϵ and mass matrix M. Its adaptive variant, the No-U-Turn Sampler (NUTS), further improves usability by eliminating the need to predefine the trajectory length L and by adapting ϵ and M during warmup. These gradient-based samplers are particularly valuable for hierarchical and latent-state models with continuous parameters, where traditional MCMC can suffer from poor mixing and slow convergence.

Comparison and Summary. To summarize the main characteristics of the MCMC techniques discussed, the following table compares typical applications, strengths, and weaknesses.

Table 2.6: Comparison of Monte Carlo Methods: Applications, Strengths, and Weaknesses

Method	**Application**	**Strengths**	**Weaknesses**
Gibbs Sampling	Models with closed-form full conditionals.	Easy to implement; no manual tuning when conjugate.	Requires conditional tractability; can be slow with strong correlations.
Forward Filtering Backward Sampling (FFBS)	Linear–Gaussian state sequences (used inside Gibbs).	Exact block updates for states; exploits Kalman filter.	Not a standalone sampler; limited to linear–Gaussian blocks.
Metropolis–Hastings	General models with intractable conditionals.	Versatile; only needs posterior kernel.	Requires proposal tuning; struggles in high dimension without preconditioning.
Hamiltonian Monte Carlo (HMC)	Continuous, correlated high-dimensional posteriors.	Efficient exploration using gradients; high ESS per iteration.	Needs gradients and warmup to tune ϵ and M.
No-U-Turn Sampler (NUTS)	Complex hierarchical/adaptive Bayesian models.	Adapts path length and step size; robust warmup.	Computationally demanding; still requires differentiability.

Each method provides a pathway to sample from complex posteriors, but they differ in assumptions, scalability, and computational cost. Their mechanisms reflect trade-offs between generality, efficiency, and ease of implementation.

Empirical Comparison. The first estimation employs Gibbs sampling combined with Forward Filtering Backward Sampling (FFBS). Figure 2.7 displays the MCMC diagnostics for the observation and system variances. The top row shows the trace plots, which indicate satisfactory convergence and mixing for both parameters. The posterior distribution of V is symmetric and sharply concentrated around the true variance, suggesting low posterior uncertainty. In contrast, the posterior for W is more dispersed and skewed, reflecting higher uncertainty in the latent-state dynamics.

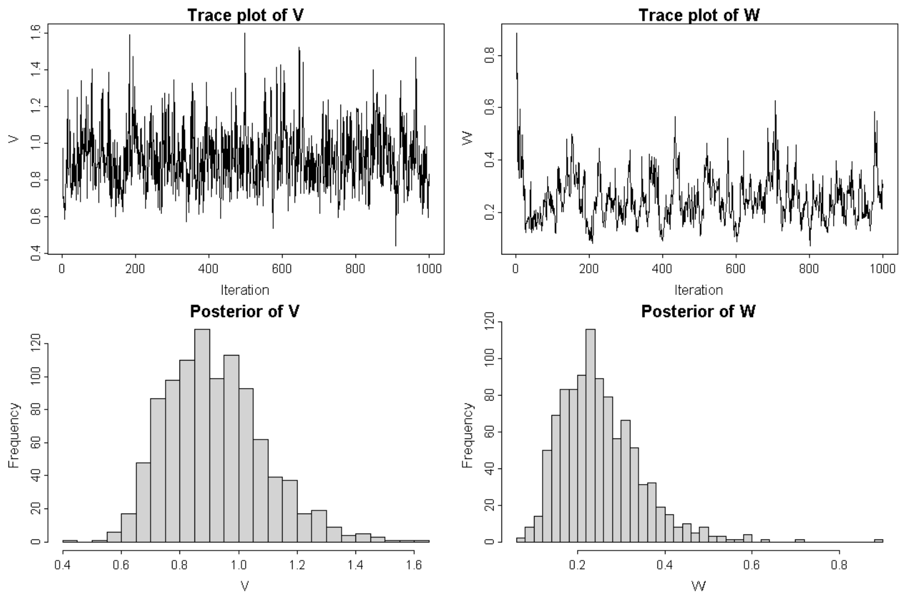


Figure 2.7: Trace plots and posterior histograms for V and W, Gibbs sampling.

Figure 2.8 compares the true latent-state trajectory with the posterior mean obtained from FFBS. The FFBS estimate closely tracks the underlying latent process throughout the series. In regions of low volatility, the match is nearly exact; where rapid shifts occur, the FFBS trajectory is smoother, consistent with Bayesian shrinkage toward persistence.

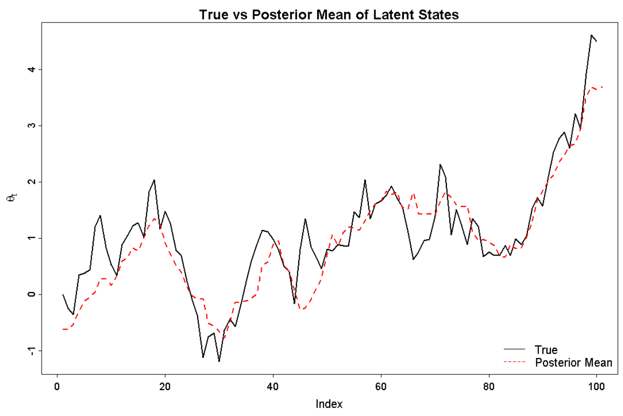


Figure 2.8: True latent states θ_t versus FFBS posterior means.

Trace plots obtained via the Metropolis–Hastings (MH) sampler are displayed in Figure 2.9.

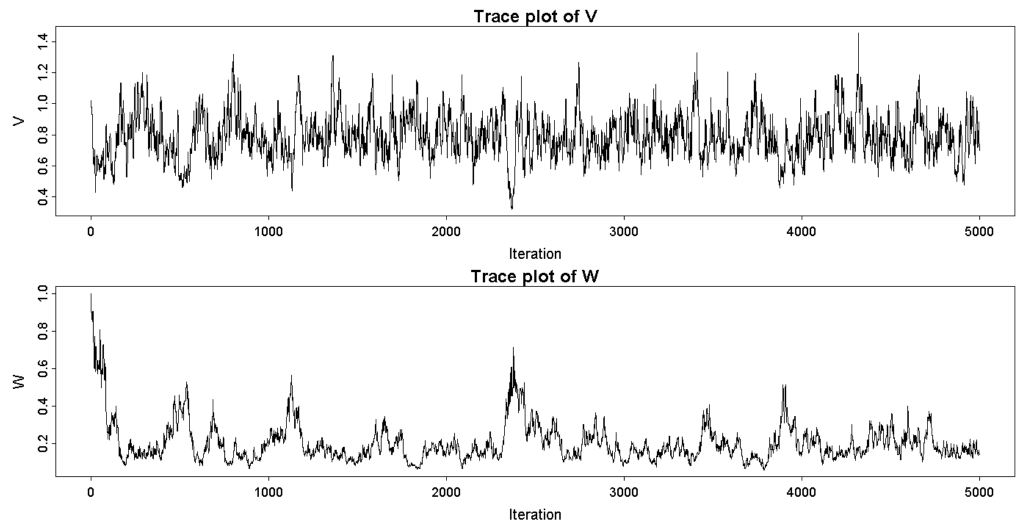


Figure 2.9: Trace plots for V and W, Metropolis–Hastings.

Compared to Gibbs sampling, convergence under MH is noticeably slower, with greater variability and more pronounced fluctuations—especially during initial iterations. The chain for W shows persistent autocorrelation and occasional sticking before stabilizing, a common feature of random-walk MH in hierarchical or weakly identified models. Despite these challenges, the chains eventually stabilize within plausible posterior regions. The slower convergence and higher autocorrelation underline the sensitivity of MH to proposal tuning and preconditioning. Nevertheless, its general applicability—without closed-form conditionals—makes it a robust fallback when conjugacy is not available.

Beyond Gibbs and Metropolis–Hastings, gradient-based samplers such as HMC and NUTS often deliver faster mixing in high dimensions. While not detailed here, they typically provide higher effective sample sizes per iteration in hierarchical state-space models with continuous parameters.

Sequential Monte Carlo (SMC) Sequential Monte Carlo (SMC) methods, also referred to as particle filters, are a family of simulation-based techniques developed to approximate sequences of posterior distributions that evolve over time. They are particularly effective for online (sequential) inference in dynamic models with latent variables and have found widespread application in areas such as signal processing, econometrics, and Bayesian filtering.

Unlike traditional Markov Chain Monte Carlo (MCMC) methods, which operate in batch mode and require access to the full dataset, SMC methods are inherently designed to handle streaming data. This makes them ideal for state-space models and real-time applications. The central idea is to represent the filtering distribution $p(\theta_t \mid y_{1:t})$ by a set of particles $\{\theta_t^{(i)}, w_t^{(i)}\}_{i=1}^N$, where $\theta_t^{(i)}$ denotes the i-th particle and $w_t^{(i)}$ its associated weight. These particles are propagated and reweighted recursively as new data y_t arrive.

The algorithm begins with an initialization step in which particles $\theta_0^{(i)}$ are sampled from the prior distribution $p(\theta_0)$, and each weight is initialized to $w_0^{(i)} = \frac{1}{N}$. At each time step t, particles are propagated via a proposal distribution, $\theta_t^{(i)} \sim q(\theta_t \mid \theta_{t-1}^{(i)}, y_t)$, and their weights are updated using importance sampling principles according to

$$w_t^{(i)} \propto w_{t-1}^{(i)} \cdot \frac{p(y_t \mid \theta_t^{(i)}) p(\theta_t^{(i)} \mid \theta_{t-1}^{(i)})}{q(\theta_t^{(i)} \mid \theta_{t-1}^{(i)}, y_t)}.$$

To prevent degeneracy—where most particles carry negligible weights—a resampling step is introduced, selecting particles in proportion to their *normalized* weights.

SMC methods offer several advantages. They naturally accommodate models with time-varying dynamics and nonlinear, non-Gaussian structures, and they provide full posterior approximations at each time step. However, they also come with limitations. Over time, weight degeneracy can severely reduce the effective sample size, and resampling, while necessary, introduces additional variance. Furthermore, the computational burden can become substantial in high-dimensional spaces.

Numerous variants have been proposed to mitigate these issues. Rao-Blackwellized particle filters exploit conditional Gaussian structures to analytically marginalize parts of the state, improving efficiency. Adaptive SMC techniques dynamically adjust proposals and resampling criteria based on real-time diagnostics. More recently, Particle MCMC (PMCMC) methods have combined SMC with MCMC algorithms, embedding particle filters within a broader sampling framework to perform exact posterior inference in models with intractable likelihoods.

From MCMC and SMC to Particle MCMC. Both Markov Chain Monte Carlo (MCMC) and Sequential Monte Carlo (SMC) provide powerful tools for Bayesian inference but address different computational challenges. MCMC excels at sampling from complex posteriors in static (batch) settings, yet struggles in time-evolving models with intractable likelihoods. Conversely, SMC yields effective online inference for dynamic latent variables but often lacks MCMC's asymptotic guarantees for high-dimensional parameter estimation.

Particle MCMC (PMCMC) integrates the strengths of both approaches by embedding particle filters within MCMC. It enables joint inference on static parameters and latent states in nonlinear, non-Gaussian, and high-dimensional state-space models, operating efficiently even when the marginal likelihood cannot be evaluated in closed form.

Formally introduced by Andrieu, Doucet, and Holenstein[7], PMCMC constructs an exact MCMC sampler targeting $p(\theta, \theta_{1:T} \mid y_{1:T})$, where θ are static parameters and $\theta_{1:T}$ latent states. The intractable marginal likelihood $p(y_{1:T} \mid \theta)$ is replaced by an *unbiased* estimate $\hat{p}(y_{1:T} \mid \theta)$ from a particle filter; by pseudo-marginal theory, using an unbiased likelihood estimator preserves the correct stationary distribution. The Metropolis–Hastings acceptance probability becomes

$$\alpha = \min\left(1, \frac{\hat{p}(y_{1:T} \mid \theta^*)\, p(\theta^*)\, q(\theta^{(i)} \mid \theta^*)}{\hat{p}(y_{1:T} \mid \theta^{(i)})\, p(\theta^{(i)})\, q(\theta^* \mid \theta^{(i)})}\right),$$

with acceptance typically computed on the log scale. In practice, the variance of $\hat{p}(y_{1:T} \mid \theta)$—controlled by the number of particles and resampling scheme—strongly affects mixing and acceptance rates.

Several implementations exist. *Particle Marginal Metropolis–Hastings* (PMMH) directly substitutes the particle-filter likelihood in the MH ratio. *Particle Gibbs* (PG) alternates state-path updates (conditional SMC) and parameter updates; *PG with Ancestor Sampling* (PG-AS) improves mixing by backward sampling of ancestors during resampling. These variants trade off computational cost and mixing efficiency but all target the exact joint posterior under an unbiased likelihood estimator.

Comparison Between MCMC and SMC. While both Markov Chain Monte Carlo (MCMC) and Sequential Monte Carlo (SMC) approximate posterior distributions, they differ in algorithmic structure, target use cases, and computational strategy.

Table 2.7: Comparison between MCMC and SMC Methods

Aspect	**MCMC**	**SMC**
Purpose	Batch inference from a fixed posterior.	Real-time updates for evolving posteriors.
Data handling	Uses full dataset (offline).	Processes data sequentially (online).
Output (typical)	Joint posterior over parameters/states (smoothing).	Filtering/predictive distributions at each t.
Sampling mechanism	Single Markov chain over parameters/states.	Population of particles with propagate–weight–resample.
Proposal design	Fixed/tuned kernels; preconditioning helpful.	Time-varying, often model-guided (e.g., bootstrap/guided).
Convergence/diagnostics	Burn-in, $\hat{R}$, ESS, trace/ACF.	ESS/degeneracy checks; resampling when needed.
High dimensionality	Can be slow without good tuning; HMC/NUTS help.	Weight/path degeneracy limits scalability.
Use cases	Static/batch posteriors; offline state smoothing.	Dynamic state-space models; online forecasting/filtering.

[7]C. Andrieu, A. Doucet, and R. Holenstein (2010). "Particle Markov Chain Monte Carlo Methods." *JRSS B*.

In summary, MCMC is best for batch posterior inference (including dynamic models treated offline), whereas SMC excels in sequential settings requiring online updates and predictive filtering; hybrids such as PMCMC bridge the two when likelihoods are intractable.

Practical Issues. Once MCMC or SMC algorithms have been implemented, validating their output is essential to ensure the reliability of posterior estimates. Convergence diagnostics and efficiency metrics help determine whether the sampling process has adequately explored the target distribution and whether the results are trustworthy for inference.

A key diagnostic is the *potential scale reduction factor*, denoted $\hat{R}$, introduced by Gelman and Rubin (1992). This statistic compares the variance within multiple chains to the variance between them. When $\hat{R}$ is close to 1, typically below 1.01 or 1.05, the chains are considered to have mixed well and to be sampling from the stationary distribution. Given m chains each of length n, the diagnostic is computed as:

$$\hat{R} = \sqrt{\frac{\hat{V}}{W}}, \quad \text{where } \hat{V} = \frac{n-1}{n}W + \frac{1}{n}B, \quad W = \frac{1}{m}\sum_{j=1}^{m} s_j^2, \quad B = \frac{n}{m-1}\sum_{j=1}^{m}(\bar{\theta}_j - \bar{\theta})^2.$$

Here, W is the within-chain variance (the average variability of samples within each individual chain), and B is the between-chain variance (the variability among the means of different chains). If all chains have converged to the same distribution, both quantities should be similar. The estimate $\hat{V}$ combines W and B to provide a stabilized measure of the total marginal posterior variance, balancing intra-chain noise and inter-chain dispersion.

The *effective sample size* (ESS) is another crucial quantity, reflecting how many independent samples are equivalent to the correlated MCMC output. Because MCMC samples exhibit autocorrelation, especially in latent state models, the raw sample size overstates the amount of information. ESS is given by:

$$\text{ESS} = \frac{N}{1 + 2\sum_{k=1}^{\infty} \rho_k},$$

where ρ_k denotes the lag-k autocorrelation. Lower autocorrelation implies higher ESS and more efficient inference. Algorithms such as HMC and NUTS often produce higher ESS values for a given number of iterations, especially in high-dimensional or hierarchical settings.

Additional tools, such as trace plots, autocorrelation plots, and kernel density overlays, offer qualitative assessments of convergence and sampling efficiency. These diagnostics help detect issues like non-stationarity, multimodality, or poor mixing that may invalidate inference.

In practice, it is standard to run multiple chains, discard initial burn-in periods, and monitor both $\hat{R}$ and ESS across all parameters of interest. This is particularly critical in dynamic linear models, where temporal dependencies and latent structures can slow mixing and distort estimates. Proper diagnostic checks ensure that sampling-based inference is robust and replicable.

2.2.3 Posterior Inference and Forecasting Accuracy

Once the joint posterior distribution $p(\theta_{1:T}, \psi \mid y_{1:T})$ is available, Bayesian inference proceeds by extracting summaries and predictive insights to support interpretation and decision-making. The most common point estimates include the posterior mean, which minimizes squared loss; the posterior median, which minimizes absolute loss; and the maximum a posteriori (MAP), defined as:

$$\mathrm{MAP}(\psi) = \arg\max_{\psi} p(\psi \mid y).$$

The MAP corresponds to the mode of the posterior and serves as a Bayesian analogue to the maximum likelihood estimate, particularly useful for skewed or unimodal distributions. Uncertainty is typically conveyed through credible intervals. A 95% credible interval $[a, b]$ satisfies

$$\mathbb{P}(\psi \in [a, b] \mid y) = 0.95,$$

providing a probabilistic interpretation of parameter uncertainty given the data; this contrasts with frequentist confidence intervals, which rely on repeated-sampling logic.

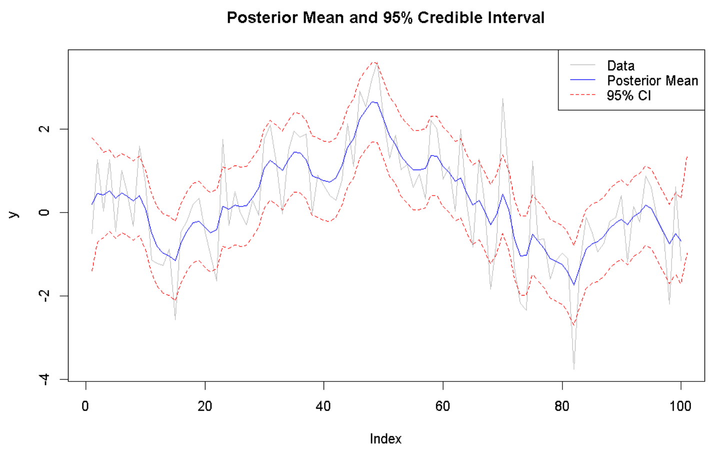


Figure 2.10: Posterior mean estimate of the latent state with 95% credible intervals.

The figure illustrates posterior smoothing in a dynamic linear model: the posterior mean tracks the latent state, while the credible bands widen in volatile periods, reflecting increased uncertainty. The table reports a simulated local-level example (posterior via Kalman smoother; one-step-ahead forecasts):

Table 2.8: Posterior Summary and Predictive Accuracy Metrics

Quantity	**Value**
MAP (state at T)	-0.548
Predictive Mean ($T+1$)	-0.548
Predictive Variance ($T+1$)	1.246
MAD (Mean Absolute Deviation)	0.900
MSE (Mean Squared Error)	1.258
MAPE (Mean Absolute Pct. Error)	2.624

The MAP estimate for the final time point is -0.548, which is also used as the one-step-ahead predictive mean. The predictive variance of 1.246 reflects both state and measurement uncertainty. Turning to accuracy metrics, the Mean Absolute Deviation (MAD) summarizes the typical error on the original scale and is comparatively robust to outliers because it penalizes deviations linearly; the Mean Squared Error (MSE) emphasizes large misses through quadratic penalization and is therefore sensitive to outliers but appropriate when large errors are particularly costly and when Gaussian loss is a reasonable approximation; the Mean Absolute Percentage Error (MAPE) expresses error in relative terms and is scale-free, which aids comparisons across series and units, yet it can become unstable when y_t approaches zero and may over-emphasize small denominators. In the example, the reported MAPE value 2.624 corresponds to 262.4%, an inflation that is typical when some y_t are near zero; in such cases, symmetric alternatives (sMAPE) or scale-free ratios (MASE) are often preferable, while MAD and MSE remain informative on the original scale.

These metrics are defined as

$$\text{MAD} = \frac{1}{n}\sum_{t=1}^{n}|e_t|, \qquad \text{MSE} = \frac{1}{n}\sum_{t=1}^{n}e_t^2, \qquad \text{MAPE} = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{e_t}{y_t}\right|,$$

where $e_t = y_t - \hat{y}_t$ denotes the one-step-ahead forecast error and $\hat{y}_t$ the posterior predictive mean. Bayesian models also support probabilistic forecasting through the posterior predictive distribution,

$$p(\tilde{y} \mid y) = \int p(\tilde{y} \mid \theta, \psi)\, p(\theta, \psi \mid y)\, d\theta\, d\psi,$$

with $\tilde{y}$ a future observation; in static models this simplifies to $\int p(\tilde{y} \mid \theta)\, p(\theta \mid y)\, d\theta$. Such predictive distributions are central to decision-making, model comparison, and graphical diagnostics because they propagate both state and parameter uncertainty. Unlike frequentist procedures based on point estimates and asymptotic approximations, Bayesian methods naturally incorporate parameter uncertainty, yielding richer inference—especially in small samples or structurally dynamic systems.

This unified posterior analysis and predictive evaluation lays the foundation for model checking and validation, addressed in the following section.

2.2.4 Real-World Violations and Extensions

The preceding sections developed posterior inference and forecasting under the linear–Gaussian DLM, where variances are fixed, observations are complete, and the measurement and evolution equations are well approximated by Gaussian noise and linear dynamics. Empirical time series—especially in economics and finance—rarely adhere to these assumptions. Volatility clusters and drifts over time, samples arrive with gaps or ragged edges, structural regimes switch discretely, and measurement or state innovations exhibit heavy tails or nonlinear responses. These features, if ignored, bias likelihood-based estimation, distort uncertainty quantification, and degrade predictive performance.

This subsection introduces extensions that restore adequacy under such departures while preserving a state–space foundation. Variance uncertainty is addressed by learning unknown (possibly time-varying) noise components within the filtering/smoothing machinery or via stochastic–volatility layers that render the model nonlinear and non-Gaussian. Missing observations are handled by treating unobserved y_t as latent and integrating them in posterior sampling, so that forecasts and intervals reflect information gaps rather than ad hoc imputations. Regime switching is incorporated through a latent Markov indicator that modulates system matrices and variances across states, allowing discrete changes in level, persistence, or volatility to be inferred jointly with the latent trajectory. Finally, when linearity or Gaussianity fail more fundamentally, computation moves beyond closed-form recursions to approximation and simulation (EKF/UKF, particle filtering, particle MCMC), ensuring that inference and prediction remain coherent despite complex dynamics. The next subsections detail these developments in turn: variance uncertainty, regime switching, missing data, and nonlinear/non-Gaussian models.

Variance Uncertainty

Bayesian inference provides a coherent probabilistic framework for uncertainty in both fixed parameters and time-varying latent states. In Dynamic Linear Models (DLMs), this is encoded in the joint posterior

$$p(\psi, \theta_{1:T} \mid y_{1:T}) = \frac{p(y_{1:T} \mid \theta_{1:T}, \psi)\, p(\theta_{1:T} \mid \psi)\, p(\psi)}{p(y_{1:T})},$$

where $p(y_{1:T})$ is the marginal likelihood. High dimensionality, temporal dependence, and non-conjugacy often preclude analytical solutions, so inference relies on stochastic simulation.

Markov Chain Monte Carlo (MCMC) generates posterior samples once the full dataset is observed, enabling offline estimation; Sequential Monte Carlo (SMC) constructs a rolling approximation as new observations arrive, supporting online learning and real-time forecasting.

In most economic and financial applications, the assumption of known, constant variances is unrealistic: observation noise and state innovations exhibit volatility clustering, regime shifts, or stochastic drift. A Bayesian approach treats fixed-but-unknown variances as parameters and infers them via Gibbs sampling when inverse-gamma full conditionals are available; when variances are time-varying, they are modeled as latent processes within the state-space.

A common specification introduces stochastic volatility through an autoregressive log-variance:

$$\log \sigma_t^2 \;=\; \mu + \phi \, \log \sigma_{t-1}^2 + \eta_t, \qquad \eta_t \sim \mathcal{N}(0, \tau^2).$$

This yields a nonlinear, non-Gaussian model that calls for simulation-based algorithms—such as Metropolis-within-Gibbs, Hamiltonian Monte Carlo, or particle MCMC—to sample from the joint posterior.

These tools propagate variance uncertainty into latent-state estimation, prediction intervals, and risk measures. Relative to frequentist plug-in or purely asymptotic corrections, the Bayesian treatment often delivers more accurate and credible inference under complex and volatile conditions.

Non-Gaussian and Nonlinear Models

While standard Dynamic Linear Models assume Gaussian errors and linear state evolution, many applications deviate from these assumptions. Economic indicators may be asymmetric or heavy-tailed, and measurement or transition relations often depend nonlinearly on past states. Such features violate the Kalman filter's prerequisites and call for more flexible computation.

In the Bayesian paradigm, non-Gaussianity and nonlinearity are handled by simulation-based inference within a generalized state-space model

$$y_t \sim p(y_t \mid \theta_t), \qquad \theta_t \sim p(\theta_t \mid \theta_{t-1}),$$

where either $p(y_t \mid \theta_t)$ or $p(\theta_t \mid \theta_{t-1})$ (or both) may be non-Gaussian/nonlinear. Examples include count data models (Poisson/binomial with log or logit links), financial returns with Student-t or α-stable innovations, and latent processes with threshold or exponential dynamics.

Closed-form recursions are generally unavailable. Particle filters (Sequential Monte Carlo, SMC) approximate the filtering law $p(\theta_t \mid y_{1:t})$ online, while Particle MCMC (PMCMC)—such as Particle Marginal Metropolis–Hastings (PMMH)—targets the full posterior by embedding a particle filter inside an MCMC sampler. These "pseudo-marginal" schemes are exact in the sense that, with an unbiased likelihood estimator, the stationary distribution of the Markov chain is the true posterior.

When the model is differentiable, Hamiltonian Monte Carlo (HMC) and its adaptive variant NUTS can explore high-dimensional posteriors efficiently by leveraging gradients of the log posterior. For models with known links (e.g., log or logistic) this often yields rapid convergence. In highly nonstandard settings, data augmentation or auxiliary-variable constructions restore tractable conditionals and improve mixing—for instance, the scale-mixture representation of the Student-t introduces latent variances that render conditionals Gaussian, and Pólya–Gamma augmentation makes logistic likelihoods conditionally Gaussian.

Bayesian computation thus provides a unified route to inference and forecasting in models far from the linear–Gaussian ideal. Although more demanding computationally, these methods deliver robust estimation and coherent uncertainty propagation under heavy tails, skewness, and nonlinear interactions frequently observed in economic and financial series. For example, daily transaction counts can be modeled with a Poisson observation equation and log-link, with latent intensity evolving nonlinearly in past states and covariates; asset returns can employ Student-t innovations to capture volatility clustering and tail risk. In both cases, SMC or HMC/PMCMC propagate posterior uncertainty through the nonlinear structure, producing credible intervals and forecasts that reflect the system's true dynamics.

Missing Data

In real-world time series, missing observations are common. Financial holidays, asynchronous macro releases, and sensor outages interrupt data collection, yet classical Dynamic Linear Models (DLMs) assume complete streams. When gaps occur—sporadically or in blocks—this assumption must be relaxed.

Bayesian inference handles missing data by treating unobserved values as latent variables jointly estimated with parameters and states. Let $y_{1:T}^{\text{obs}}$ and $y_{1:T}^{\text{mis}}$ denote observed and missing values, and write ψ for static parameters and $\theta_{1:T}$ for latent states. The posterior factorizes as

$$p(\psi, \theta_{1:T}, y_{1:T}^{\text{mis}} \mid y_{1:T}^{\text{obs}}) \;\propto\; p(y_{1:T}^{\text{obs}}, y_{1:T}^{\text{mis}} \mid \theta_{1:T}, \psi)\, p(\theta_{1:T} \mid \psi)\, p(\psi),$$

so that the full-data likelihood is reconstructed through the observation model.

This formulation admits direct MCMC. Under a Gaussian DLM,

$$y_t^{\text{mis}} \mid \theta_t, \psi \;\sim\; \mathcal{N}\big(F_t^\top \theta_t,\; V_t\big),$$

so a Gibbs step can impute y_t^{mis} at each iteration. Embedding this imputation in the sampling cycle propagates uncertainty from data gaps into state estimates, forecasts, and risk measures. With long gaps, Kalman prediction naturally bridges missing spans in the forward pass, while FFBS smoothing reconstructs the path using information before and after the gap. In multivariate or ragged-edge panels, a selection matrix H_t can drop missing components from the measurement equation so filtering proceeds without ad hoc preprocessing.

Validity hinges on an *ignorable* missingness mechanism. For MCAR (independent of both observed and unobserved values), MAR (depends on observed but not unobserved values), or structured monotone dropout, posterior inference remains valid without modeling selection. When missingness is non-ignorable (MNAR), the selection process must be modeled explicitly, e.g., by augmenting the likelihood with $p(m_t \mid y_t, \cdot)$.

The main challenge is computational: missing values increase posterior dependence and can slow mixing. Blocked updates (e.g., jointly sampling y_t^{mis} with θ_t), reparameterizations (standardized-innovation forms), and adaptive tuning help maintain efficiency. Combined with FFBS (or particle methods in non-Gaussian cases), these strategies enable robust inference even with substantial data loss.

Regime Switching

Structural breaks and regime shifts are central features of many economic and financial time series. Markets alternate between bullish and bearish conditions; macro indicators evolve through expansions and recessions; asset returns fluctuate between low- and high-volatility phases. Standard DLMs, constrained by smooth linear–Gaussian dynamics, struggle to capture abrupt transitions. Markov-switching Dynamic Linear Models (MS–DLMs) address this by introducing a latent regime indicator governed by a finite-state Markov chain.

Following Hamilton (1989), augment the DLM with a discrete state $s_t \in \{1, \ldots, K\}$, each indexing a regime-specific DLM:

$$\begin{aligned} y_t &= F_t^{(s_t)}\theta_t + v_t^{(s_t)}, \quad v_t^{(s_t)} \sim \mathcal{N}(0, V_t^{(s_t)}), \\ \theta_t &= G_t^{(s_t)}\theta_{t-1} + w_t^{(s_t)}, \quad w_t^{(s_t)} \sim \mathcal{N}(0, W_t^{(s_t)}), \end{aligned}$$

and let the regime path $s_{1:T}$ evolve as a first-order Markov chain with transition matrix $\Pi = [\pi_{ij}]$.

In the Bayesian framework, $s_{1:T}$ is inferred jointly with latent states and static parameters ψ (including Π and regime-specific blocks). The joint posterior is

$$p(\psi, \theta_{1:T}, s_{1:T} \mid y_{1:T}) \;\propto\; p(y_{1:T} \mid \theta_{1:T}, s_{1:T}, \psi)\, p(\theta_{1:T} \mid s_{1:T}, \psi)\, p(s_{1:T} \mid \Pi)\, p(\psi),$$

with convenient priors such as independent Dirichlet rows on Π. Sampling typically alternates continuous-state updates (Kalman-based FFBS within each regime configuration) with discrete-state updates of $s_{1:T}$. The forward recursion for regime probabilities is the *Hamilton filter*; backward smoothing of regimes via dynamic programming is often called the *Kim smoother* (see Kim, 1994; Kim & Nelson, 1999). For moderate K and T, exact discrete FFBS is feasible; for longer series or richer specifications, particle methods with discrete resampling and Rao–Blackwellisation provide scalable approximations.

Bayesian MS–DLMs yield probabilistic regime classification over time, incorporate structural uncertainty through priors on Π and regime-specific parameters, and enable coherent inference under nonstationary conditions. Practical considerations include computational cost (the number of regime paths grows exponentially with T), the need for blocked/ancillary-sufficient updates to improve mixing, and potential label-switching across regimes—often mitigated by identifiability constraints or mildly informative priors. Despite these challenges, MS–DLMs are a powerful extension of the Bayesian state–space toolkit for capturing discrete structural dynamics pervasive in economic and financial data.

2.2.5 Model Checking and Validation

After estimating a Bayesian DLM, adequacy can be assessed via *posterior predictive checks* (PPC), which ask whether the model can reproduce salient features of the observed series. PPCs draw replicated datasets $\tilde{y}^{(s)} \sim p(\tilde{y} \mid \theta^{(s)}, \psi^{(s)})$ at posterior draws $(\theta^{(s)}, \psi^{(s)}) \sim p(\theta, \psi \mid y)$, and compare summaries (discrepancies) $T(y, \theta, \psi)$ with their replicated counterparts $T(\tilde{y}^{(s)}, \theta^{(s)}, \psi^{(s)})$.

Discrepancies $T(\cdot)$ target aspects of fit such as mean, variance, skewness, kurtosis, residual autocorrelation, or covariate sensitivity. By contrasting $T(y, \theta, \psi)$ with the posterior distribution of $T(\tilde{y}, \theta, \psi)$, it is possible to diagnose whether the model reproduces the empirical features of interest.

In time–series settings, a central diagnostic examines the autocorrelation of *posterior one–step residuals*

$$e_t = y_t - \mathbb{E}\big(y_t \mid y_{1:t-1}\big), \qquad \mathbb{E}\big(y_t \mid y_{1:t-1}\big) = \iint \mathbb{E}\big(y_t \mid \theta_t, \psi\big)\, p(\theta_t, \psi \mid y_{1:t-1})\, d\theta_t\, d\psi,$$

i.e., the posterior predictive mean under the filtered information set. Persistent autocorrelation in $\{e_t\}$ indicates unmodeled dynamics or misspecified noise.

These checks are in–sample and fully probabilistic. A useful scalar summary is the *Bayesian p-value*

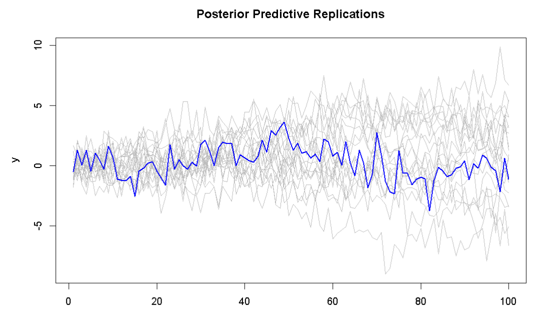
$$p_B = \mathbb{P}\big(T(\tilde{y}, \theta, \psi) \geq T(y, \theta, \psi) \mid y\big),$$

the posterior probability that the replicated discrepancy exceeds the observed one. Values near 0.5 indicate adequate fit; values close to 0 or 1 (e.g., < 0.05 or > 0.95) flag potential inadequacy. Unlike classical p-values, p_B is not uniformly distributed under the null and should be viewed as a descriptive fit indicator.

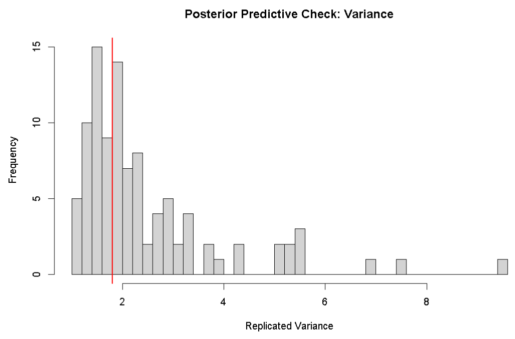
Graphical PPCs—Q–Q plots, overlays of observed and replicated series, and ACFs of replicated residuals—complement numerical summaries and often reveal deficiencies at a glance.

Posterior predictive checks provide a flexible, interpretable framework for validating Bayesian DLMs, probing both structural fidelity and uncertainty representation. When key patterns are not reproduced, PPCs point toward targeted refinements in priors, latent dynamics, or the observation model.

Posterior Predictive Visualization. Figure 2.11 illustrates PPCs for the fitted model. Panel (a) overlays twenty replicated paths (grey) from the posterior predictive distribution on the observed series (blue). Panel (b) compares the distribution of sample variances across replications to the empirical variance (vertical red line). The observed trajectory lies well within the replicate envelope, and the variance check shows no under- or over-dispersion, supporting an adequate fit.



(a) Predictive replications



(b) Replicated variance check

Figure 2.11: Posterior predictive checks: replicated paths and variance distribution.

2.2.6 Model Comparison: Bayesian Framework

Model comparison in the Bayesian framework emphasizes *predictive* performance. The aim is to prefer models that explain the observed data and generalize to unseen outcomes, as quantified by the (log) predictive density

$$\log p(\tilde{y} \mid y) \;=\; \log \int p(\tilde{y} \mid \theta)\, p(\theta \mid y)\, d\theta,$$

with $\tilde{y}$ denoting future data (conditional on any future covariates if present). When external test data are unavailable, predictive performance is approximated in-sample via information criteria, cross-validation surrogates, or marginal likelihoods.

Information Criteria. Information criteria provide computationally efficient approximations to assess a model's out-of-sample predictive performance. The *log pointwise predictive density* (lppd) computed from posterior draws $\{\theta^{(s)}\}_{s=1}^{S}$ is

$$\text{lppd} \;=\; \sum_{t=1}^{T} \log\left(\frac{1}{S}\sum_{s=1}^{S} p(y_t \mid \theta^{(s)})\right),$$

which overestimates out-of-sample accuracy because it is evaluated on the training set. Complexity penalties correct this optimism.

AIC (frequentist baseline under flat priors and regular MLE) is

$$\text{AIC} \;=\; -2\log p(y \mid \hat{\theta}_{\text{MLE}}) + 2k,$$

where k is the number of estimated parameters. Though not inherently Bayesian, AIC serves as a baseline under flat priors and asymptotic normality.

The *Deviance Information Criterion* (DIC)introduced by Spiegelhalter et al. (2002), adapts AIC for Bayesian models by incorporating posterior uncertainty. Defined as:

$$\mathrm{DIC} \;=\; -2\log p\big(y \mid \hat{\theta}_{\text{Bayes}}\big){+}2p_D, \qquad p_D \;=\; 2\Big[\log p\big(y \mid \hat{\theta}_{\text{Bayes}}\big) - \mathbb{E}_{\theta|y}\{\log p(y \mid \theta)\}\Big] .$$

The complexity penalty p_D is calculated as:

$$p_D = 2\left(\log p(y \mid \hat{\theta}_{\text{Bayes}}) - \mathbb{E}_{\theta|y}[\log p(y \mid \theta)]\right) ,$$

representing the discrepancy between the log-likelihood at the posterior mean and its posterior expectation. Larger values of p_D suggest higher model flexibility, which may lead to overfitting. DIC is convenient when posterior samples are available but less reliable for models with strong nonlinearity or multimodal posteriors.

The *Watanabe–Akaike Information Criterion* (WAIC), developed by Watanabe (2010), is fully Bayesian, averaging over the posterior rather than using a point estimate. Using the same posterior draws,

$$\mathrm{WAIC} \;=\; -2\left[\sum_{t=1}^{T}\log\Big(\tfrac{1}{S}\sum_{s=1}^{S}p(y_t \mid \theta^{(s)})\Big) \;-\; p_{\text{WAIC}}\right] ,$$

with effective complexity

$$p_{\text{WAIC}} \;=\; \sum_{t=1}^{T}\mathrm{Var}_s\big(\log p(y_t \mid \theta^{(s)})\big).$$

WAIC is asymptotically equivalent to Bayesian leave-one-out cross-validation and behaves well in many singular/high-dimensional settings where simple parameter counts are misleading.

A closely related and often more stable alternative is *PSIS-LOO* (Pareto-smoothed importance sampling leave-one-out), which directly approximates the expected log predictive density by re-weighting the same posterior draws; in practice, WAIC and PSIS-LOO typically agree, with PSIS-LOO offering diagnostics for influential points via the Pareto k shape parameter.

Lower values of AIC/DIC/WAIC (or higher expected log predictive density/LOO) indicate better expected out-of-sample performance. When comparing models, report both the criterion and its standard error (from posterior draws or PSIS weights) to gauge whether differences are practically meaningful.

Cross-Validation. Cross-validation assesses out-of-sample predictive performance by holding out data and evaluating how well the model predicts unseen observations. *Leave-one-out* (LOO) omits each ith point in turn, fits on y_{-i}, and evaluates $\log p(y_i \mid y_{-i})$, yielding the expected log predictive density

$$\text{elpd}_{\text{LOO}} = \sum_{i=1}^{n} \log p(y_i \mid y_{-i}).$$

Exact LOO is often expensive because it requires n refits. A computationally efficient alternative is *Pareto-Smoothed Importance Sampling* (PSIS), which reuses posterior samples from the full dataset $p(\theta \mid y)$ and corrects for the approximation bias through importance weights smoothed via the generalized Pareto distribution. PSIS-LOO achieves high accuracy with significantly reduced computational cost and is widely adopted in modern Bayesian workflows.

Like WAIC, LOO-CV evaluates model performance pointwise, but without relying on analytic approximations or specific partitioning assumptions. In well-specified models with sufficient data, both criteria tend to converge asymptotically, offering reliable guidance for model selection based on predictive accuracy

Bayes Factors: Formal Hypothesis Testing Between Models While predictive criteria such as WAIC and LOO target out-of-sample accuracy, the Bayes factor compares the plausibility of entire models under both data and priors. This is valuable for nested specifications and theory-driven hypotheses, where the goal is not only prediction but also assessing which model the data support in a probabilistic sense. For two candidates $\mathcal{M}_1$ and $\mathcal{M}_2$, the Bayes factor is the ratio of marginal likelihoods,

$$\text{BF}_{12} = \frac{p(y \mid \mathcal{M}_1)}{p(y \mid \mathcal{M}_2)}, \qquad p(y \mid \mathcal{M}) = \int p(y \mid \theta, \mathcal{M})\, p(\theta \mid \mathcal{M})\, d\theta,$$

so that $\log \text{BF}_{12}$ summarizes relative evidence on a convenient additive scale. Values above zero favor $\mathcal{M}_1$; larger magnitudes indicate stronger evidence. Because the marginal likelihood averages the likelihood over the prior, Bayes factors automatically implement an “Occam’s razor” penalty: more diffuse or higher-dimensional models must earn their complexity by concentrating posterior mass in regions with high likelihood. This prior averaging, however, also implies sensitivity to prior specification—marginal likelihoods are undefined under improper priors and can shift materially under weakly informative choices; in testing point-null hypotheses, this can manifest as Lindley’s paradox when large samples conflict with diffuse priors.

In DLM applications, Bayes factors are used to compare static versus dynamic regression effects, assess seasonal or intervention components, or adjudicate between alternative state-evolution structures. Computation is nontrivial in hierarchical or high-dimensional settings; practical estimators include bridge sampling and thermodynamic (path) integration, nested sampling for complex posteriors, and, in nested models with proper priors on nuisance blocks, the Savage–Dickey density ratio. While theoretically coherent for model choice, Bayes factors complement, rather than replace, predictive criteria: the former quantify evidence about models themselves, the latter emphasize expected performance on future data.

Comparative Overview of Bayesian Model Comparison Tools. To clarify the complementary roles and trade-offs among DIC, WAIC, LOO-CV, and Bayes Factors, Table 2.9 summarizes their core features—purpose, strengths, and limitations.

Table 2.9: Comparison of Bayesian Model Comparison Methods

Criterion	**Goal**	**Advantages**	**Limitations**
DIC (Deviance Information Criterion)	Combine model fit and complexity using posterior means	Simple to compute from MCMC output; interpretable as a Bayesian analog of AIC	Relies on asymptotic approximations; less reliable for hierarchical, nonlinear, or multimodal posteriors
WAIC (Watanabe–Akaike Information Criterion)	Estimate out-of-sample predictive accuracy by averaging over the posterior	Fully Bayesian; suited to hierarchical models; asymptotically close to LOO-CV	Can be variable under misspecification or few posterior draws; sensitive to how p_{WAIC} is estimated
LOO-CV (Leave-One-Out Cross-Validation)	Assess predictive performance by omitting each observation in turn	Robust to priors; broadly applicable; efficiently approximated via PSIS-LOO with Pareto-k diagnostics	Exact LOO is costly for large n; sensitive to influential points (flagged by large Pareto-k)
Bayes Factors	Quantify support for discrete models via marginal likelihoods	Coherent with Bayesian theory; integrates model structure and priors; natural Occam penalty	Sensitive to prior choice; undefined with improper priors; computationally intensive in complex/hierarchical models

Each method targets a distinct inferential aim. Bayes Factors weigh marginal evidence between discrete models (useful for theory-driven hypotheses), whereas DIC, WAIC, and LOO focus on predictive accuracy with different balances between complexity control and computational cost. In applied work, WAIC and PSIS-LOO are often preferred for flexible or hierarchical models; Bayes Factors remain useful when comparisons are well-specified and grounded in informative priors.

Model Expansion as Iterative Refinement. Bayesian modeling need not treat selection as a one-off competition among fixed alternatives. Often it proceeds by model expansion—gradually enriching a baseline to better capture structure—so the emphasis shifts from choosing a single "best" model to iteratively improving predictive performance and interpretability. A common step is moving from fixed-parameter to hierarchical formulations that enable partial pooling across groups, letting the degree of sharing be learned from the data rather than imposed. This perspective recognizes that all models are approximations and seeks a balance between flexibility and parsimony. Predictive tools such as WAIC and PSIS-LOO help judge whether refinements yield substantive gains without overfitting, while posterior predictive checks provide complementary, visual validation; together they support a workflow of continuous model diagnosis and improvement.

2.2.7 Summary

The Bayesian paradigm enables a coherent and flexible framework for estimating Dynamic Linear Models by treating both latent states and parameters as probabilistic entities. This joint modeling approach supports principled uncertainty quantification, allowing full posterior distributions over time-varying dynamics and facilitating credible forecasts, particularly useful in volatile environments like financial markets.

In DLMs, Bayesian methods excel at accommodating variance uncertainty (unknown or time-varying V_t, W_t), non-Gaussian and nonlinear structures, missing data, and regime switching. These features are handled via simulation-based computation—Gibbs/FFBS for conditionally Gaussian blocks, Metropolis–Hastings or HMC/NUTS for nonconjugate or high-dimensional posteriors, SMC for online filtering, and PMCMC when the likelihood is intractable—so that uncertainty about states, parameters, and structural components is consistently propagated to forecasts.

The modular nature of Bayesian computation supports adaptability: priors can be tailored (including weakly informative priors for scales), samplers substituted, and model components refined independently. Model adequacy and predictive performance are assessed with posterior predictive checks (PPC) and predictive criteria such as WAIC and PSIS–LOO, providing complementary evidence alongside domain-driven diagnostics.

These advantages come with greater computational and analytical complexity. High-dimensional hierarchies or latent volatility layers demand advanced algorithms and substantial compute; sensitivity to prior choices—especially for variance components—requires careful regularization and robustness checks.

Overall, Bayesian estimation offers a powerful yet demanding methodology for dynamic time-series analysis. Its ability to capture uncertainty, adapt to structural complexity, and support decision-making makes it well suited to modern econometric applications, provided computational costs and diagnostic rigor are addressed.

2.3 Frequentist vs Bayesian Inference: A Unified View

In Dynamic Linear Models (DLMs), statistical inference may proceed under either the frequentist or the Bayesian paradigm. These traditions differ in their interpretation of probability, in how they handle unknown quantities, and in the computational routes they employ. This section contrasts them along core philosophy, standard workflow, computational machinery, typical fields of application, and behavior under the empirical violations discussed in this chapter.

2.3.1 Core Philosophies and Standard Workflows

Frequentist. Parameters are fixed but unknown; uncertainty arises from data variability under hypothetical repetitions. Estimation typically proceeds via maximum likelihood (MLE), with the Kalman filter delivering closed-form recursions in the linear–Gaussian case. Unknown variances and structural components are handled through optimization-based routines such as the Expectation–Maximization (EM) algorithm, and uncertainty is quantified using asymptotic approximations for confidence intervals and tests.

Bayesian. Both parameters and latent states are modeled as random. Priors encode initial beliefs and Bayes' theorem updates them to a posterior. Under linear–Gaussian assumptions, the Kalman filter appears as recursive Bayesian updating; for general models, simulation-based methods—Markov Chain Monte Carlo (MCMC), Forward Filtering Backward Sampling (FFBS), and Sequential Monte Carlo (SMC)—approximate the joint posterior so that uncertainty is propagated to states, parameters, and forecasts.

Table 2.10: Frequentist and Bayesian Inference in Dynamic Linear Models

Aspect	Frequentist Approach	Bayesian Approach
Interpretation of probability	Long-run frequencies; parameters fixed.	Degrees of belief; parameters and states random.
Treatment of uncertainty	Post-estimation via CIs and asymptotics.	Directly via posteriors and credible intervals.
Estimation tools	Kalman filter, MLE, EM; point estimates.	Kalman (conjugate), FFBS, MCMC, SMC; full posterior.
Sequential updating	States conditional on fixed parameters; re-optimization for updates.	Posterior updated recursively as data arrive.
System variances	Estimated as nuisance via MLE/EM/QMLE.	Modeled with priors (possibly hierarchical); fully propagated.
Handling missing data	Skipped measurement update / EM-based smoothing-imputation.	Marginalization with latent-data augmentation within MCMC/SMC.
Model flexibility	Extensions require new optimizers/filters (EKF/UKF, switching).	Naturally accommodates hierarchical, non-Gaussian, nonlinear, switching.
Model selection	AIC, BIC, residual diagnostics.	PPC, WAIC, PSIS-LOO, Bayes factors.
Advantages	Speed; interpretability; asymptotic guarantees.	Full uncertainty quantification; adaptability; probabilistic forecasts.
Limitations	Understates parameter uncertainty; rigidity under misspecification.	Higher computational cost; prior sensitivity.

2.3.2 Beyond the Standard DLM: Computation Under Real-World Violations

Analytical convenience in DLMs relies on linear dynamics, Gaussian disturbances, and known variances, whereas empirical series often feature irregular sampling, time-varying volatility, missing observations, regime changes, and nonlinearities. The two paradigms address these departures with distinct but parallel toolkits delineated in this chapter.

Under variance uncertainty—whether parameters are unknown or genuinely time-varying—the frequentist route estimates variance components by (quasi-)maximum likelihood, often via the EM algorithm, and can accommodate gradual change with rolling or local likelihood. For nonlinearity or non-Gaussian volatility it employs Extended/Unscented Kalman approximations or simulated-likelihood/iterated filtering; heteroskedasticity- and autocorrelation-robust standard errors (HAC/Newey–West, sandwich/OPG) support inference under mild misspecification. The Bayesian route places priors on variance blocks (e.g., inverse-gamma or weakly informative half-Cauchy) and models time-variation as a latent process (e.g., AR dynamics for $\log \sigma_t^2$ in stochastic volatility), using Gibbs or Metropolis-within-Gibbs steps, gradient-based HMC/NUTS, SMC for online filtering, or PMCMC when the likelihood is intractable but unbiasedly estimable, thereby propagating variance uncertainty through states, parameters, and forecasts.

When observations are missing, the frequentist filter omits the measurement update and proceeds with prediction, while EM with smoothing recovers expectations to re-estimate parameters. The Bayesian formulation augments the posterior with y_t^{mis} and samples them within MCMC under ignorable mechanisms (MCAR/MAR/monotone dropout), often using blocked updates and FFBS to maintain mixing so that predictive distributions reflect data gaps.

For non-Gaussian measurements or nonlinear state evolution, frequentist computation uses EKF/UKF for mild departures and turns to simulated likelihood or iterated filtering for severe deviations and intractable densities, with performance governed by approximation quality and regularity conditions. Bayesian computation specifies generalized state–space models and targets the exact posterior with SMC or PMCMC (leveraging unbiased likelihood estimates), or employs HMC/NUTS where differentiability permits, with data augmentation to restore tractable conditionals in special cases.

In regime switching, the frequentist approach adopts hidden Markov structures, evaluates the likelihood via the Hamilton filter, and obtains smoothed regime probabilities; EM is commonly used to update transition matrices and regime-specific parameters. The Bayesian approach samples the discrete regime path $s_{1:T}$ together with states and parameters using discrete FFBS, mixture filtering, or particle methods, placing priors on Π and regime-specific blocks so that structural uncertainty is represented in posterior summaries.

This comparative overview is intentionally confined to the conceptual, algorithmic, and diagnostic material developed in this chapter. Chapter 3 turns to empirical applications, where these ingredients are implemented side by side under missing data, time-varying volatility, and regime switching to assess their practical behavior.

Chapter 3

Empirical Analysis with Dynamic Linear Models

This chapter uses a single coherent macroeconomic case to turn the tools of Chapter 2 into an operational workflow. The dataset is Italy's annual real GDP growth over 1960–2024, measured year–over–year (y/y) in percent.[1] The long span provides context across expansions, recessions, and exceptional episodes, while a clearly defined modern subsample (e.g., 1995–2024) is used later for focused evaluation without breaking the narrative thread.

The modeling strategy specializes the state–space framework to the Local Level Dynamic Linear Model (random walk plus noise), in which the observed series is interpreted as noisy measurements around a gradually evolving latent level. It preserves the latent level of the growth rate that differencing of levels would otherwise discard, keeps the roles of signal and noise visible at each step, and offers a transparent bridge to classical time–series intuition. A richer Local Linear Trend specification is documented in the Appendix as a robustness check and a route to additional flexibility; here the focus remains on the Local Level so that frequentist and Bayesian results are developed on the same specification and in the sequence set out in Chapter 2.

Because the inferential stance changes how uncertainty is encoded and interpreted, the same specification is analyzed from two complementary perspectives on identical evidence. In the frequentist view, the variance components are estimated by maximizing the likelihood via the Kalman filter; uncertainty is summarized by large–sample confidence intervals and adequacy is assessed through standard diagnostics and information criteria. In the Bayesian view, weakly informative priors are combined with the same likelihood to obtain a posterior by simulation; smoothed states, predictive distributions, and credible

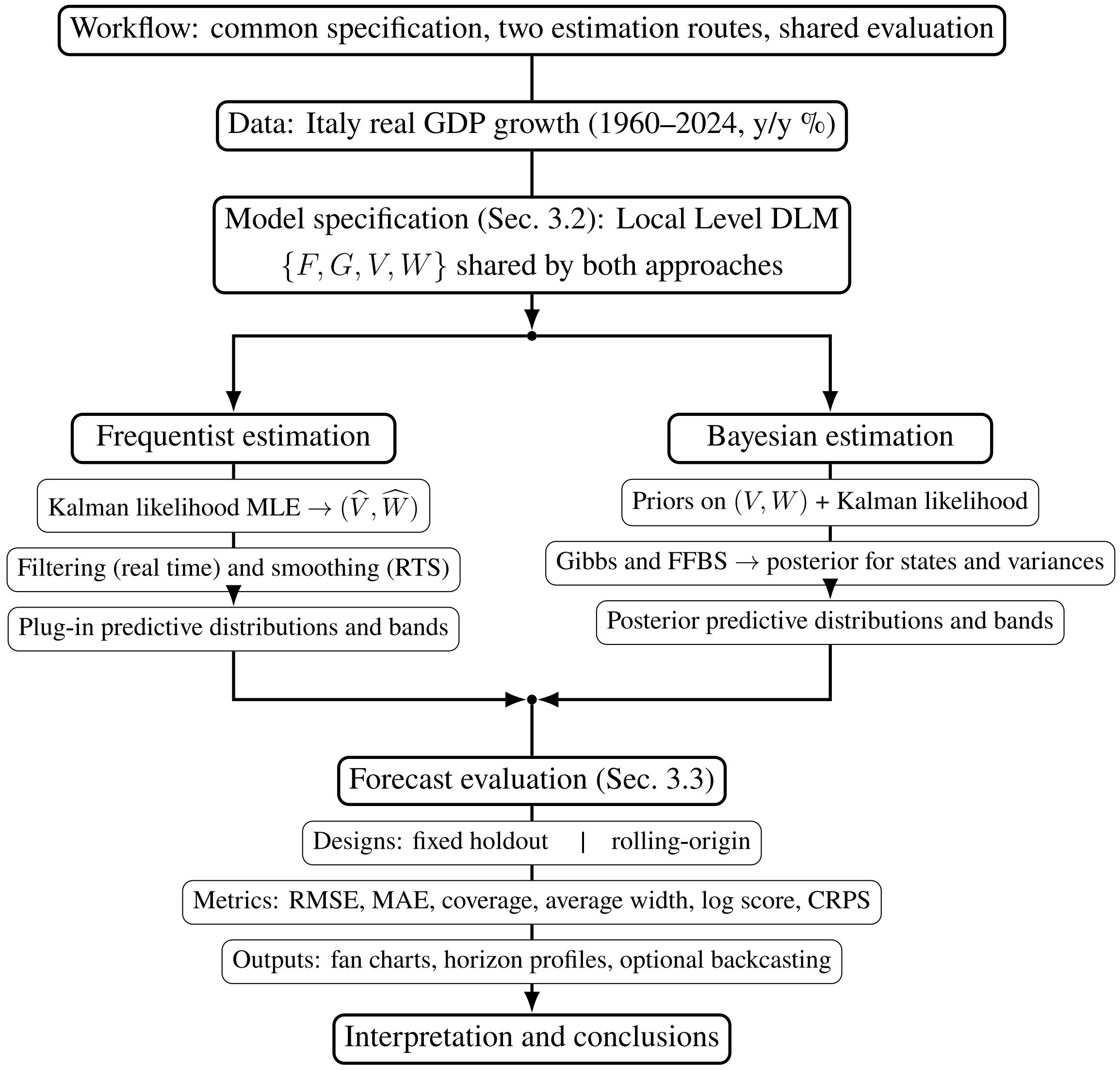
[1]Source: World Development Indicators (indicator `NY.GDP.MKTP.KD.ZG`); latest available vintage at time of writing.

intervals follow, and parameter uncertainty is propagated to downstream summaries. The pipeline—filtering, smoothing, forecasting—remains the same in both cases, so any differences can be attributed to stance rather than design. After establishing the in–sample fit, the analysis moves to out–of–sample forecasting at short and medium horizons, then to the handling of temporary data unavailability (ragged–edge/nowcasting) with the same machinery, and finally to departures from normality suggested by residual diagnostics, together with pragmatic remedies (e.g., Student-t innovations, variance discounting) and pointers to the non–Gaussian extensions outlined in Chapter 2.

Why compare Bayesian and Frequentist inference in this chapter? A side–by–side comparison is warranted because, under the *same* model and data, different uncertainty treatments lead to different decision thresholds and forecast calibration. In Dynamic Linear Models this matters where they are most sensitive: (i) how responsive the latent signal is to new observations; (ii) how wide predictive intervals become under weak identification; and (iii) how robust summaries remain when evidence is uneven across time. Italy's real GDP growth over 1960–2024 displays precisely these features: a mix of tranquil expansions and recessions, shifting signal–to–noise ratios across subperiods, and modern subsamples that compress information for variance components.

Under such conditions, plug–in inference can misstate uncertainty if key components are weakly identified, whereas posterior averaging can mitigate this by weighting plausible configurations without changing the underlying model. The comparison thus serves both a *diagnostic* role—differences in extracted signal, interval width, and out–of–sample calibration can be attributed to stance—and a *pragmatic* one: speed and benchmarking on the frequentist side; full uncertainty propagation and coherent predictive distributions on the Bayesian side. A detailed head–to–head assessment is deferred to the concluding section; the present motivation explains why both lenses are carried forward under the same specification and evidence.

Workflow. The chapter implements a single–model workflow. Section 3.2 fixes the Local Level state–space specification $\{F, G, V, W\}$ (Kalman recursions as in Chapter 2); holding the specification constant ensures that subsequent differences reflect the inferential stance rather than design changes. Estimation then proceeds along two routes: a frequentist route that maximizes the Kalman likelihood for $(\widehat{V}, \widehat{W})$ and treats these as fixed inputs for filtering, smoothing, and plug–in forecasting; and a Bayesian route that combines weakly informative priors with the same likelihood, runs Gibbs and FFBS to sample variance components and states jointly, and produces posterior predictive forecasts that integrate parameter uncertainty. The two branches rejoin in a common evaluation—fixed holdout and rolling–origin—with identical metrics (see Fig. 3.1), after which results are interpreted jointly.



Workflow: common specification, two estimation routes, shared evaluation
Data: Italy real GDP growth (1960–2024, y/y %)
Model specification (Sec. 3.2): Local Level DLM $\{F, G, V, W\}$ shared by both approaches
Frequentist estimation
Kalman likelihood MLE $\rightarrow (\widehat{V}, \widehat{W})$
Filtering (real time) and smoothing (RTS)
Plug-in predictive distributions and bands
Bayesian estimation
Priors on (V, W) + Kalman likelihood
Gibbs and FFBS $\rightarrow$ posterior for states and variances
Posterior predictive distributions and bands
Forecast evaluation (Sec. 3.3)
Designs: fixed holdout | rolling-origin
Metrics: RMSE, MAE, coverage, average width, log score, CRPS
Outputs: fan charts, horizon profiles, optional backcasting
Interpretation and conclusions

Figure 3.1: End-to-end workflow.

3.1 Data Description

The empirical analysis uses the annual growth rate of Italy's gross domestic product, defined as the year-over-year *percent change* in real GDP volume at constant 2015 prices (local currency units). The series spans 1960–2024 at annual frequency and is sourced from the World Development Indicators. One internal missing observation is handled natively by the Kalman filter; for the backcasting exercise, the 1950–1959 block is added as leading `NA`s and estimated within the same state–space framework. No further transformations are applied beyond storing the data as a univariate time series. To provide historical context without breaking the narrative, a modern subsample (1995–2024) is later used for focused evaluation.

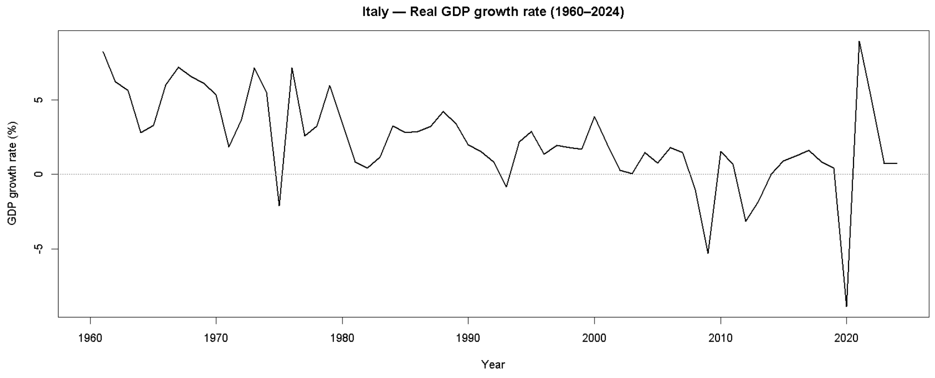


Figure 3.2: Italian real GDP growth (year-over-year, *percent*), 1960–2024. Source: World Development Indicators (`NY.GDP.MKTP.KD.ZG`).

As shown in Figure 3.2, the long post-war expansion is followed by the oil-price disruptions of the 1970s, which raise volatility and lower average growth; in the early 1990s, exchange-rate tensions and fiscal consolidation weigh on activity before a period of stabilization associated with deeper European integration. The late 1990s and early 2000s bring moderate expansion supported by external demand, tempered by structural headwinds such as weak productivity growth and demographic aging. The global financial crisis produces a deep contraction centered on 2009, followed by a slow, uneven recovery; sovereign-debt tensions in 2011–2013 interrupt that recovery, after which growth regains modest momentum through 2019 without altering its underlying pace. The pandemic shock in 2020 triggers the sharpest single-year decline in the sample, with a strong rebound in 2021 as services reopen and policy support peaks; an energy-price spike in 2022 adds renewed volatility, followed by moderation in 2023–2024. Across these episodes the series displays persistent multi-year runs—expansions or slowdowns—punctuated by rare, large shifts.

Using the growth rate rather than the level or per-capita variants keeps the target scale-free and directly comparable over time, aligns it with short- and medium-horizon monitoring, and avoids mixing population dynamics with cyclical signals. Measuring in constant prices isolates real output dynamics within the national-accounts framework; occasional revisions and choices such as rebasing or the use of volume indices can explain unusual movements.

These properties motivate a Dynamic Linear Model as the organizing lens. A state–space representation preserves a latent level of growth, lets it adjust gradually to new information, and provides real-time updates through filtering and smoothing; it also handles occasional gaps without pre-imputation and returns uncertainty bands reflecting both observation noise and state variability. Building on these considerations, the next section turns to the formal specification of the model employed in this study.

3.2 Model Specification

This section formalizes the state–space model used throughout the chapter. The Local Level (LL) Dynamic Linear Model is adopted as the baseline specification; a richer Local Linear Trend (LLT) variant is documented in Appendix 3.6 as a robustness reference that allows additional flexibility without altering the empirical design, and a compact ARIMA(0,1,1) benchmark appears in Appendix 3.7.

Modeling framework. The general linear–Gaussian state–space system is

$$\begin{aligned} y_t &= F_t\theta_t + v_t, \qquad & v_t &\sim \mathcal{N}(0, V), \\ \theta_t &= G_t\theta_{t-1} + w_t, \qquad & w_t &\sim \mathcal{N}(0, W), \end{aligned}$$

with latent state θ_t and mutually independent innovations (v_t) and (w_t).

Local Level (LL). The LL model assumes a latent level μ_t evolving as a random walk:

$$\begin{aligned} y_t &= \mu_t + v_t, \qquad & v_t &\sim \mathcal{N}(0, V), \\ \mu_t &= \mu_{t-1} + w_{\mu,t}, \qquad & w_{\mu,t} &\sim \mathcal{N}(0, W). \end{aligned}$$

Figure 3.3 reports filtered and smoothed estimates under maximum likelihood. The black line is Italy's annual real GDP growth (1960–2024). Volatility rises around well-known episodes: the oil-price shocks of the 1970s, the global financial crisis (2009), the sovereign-debt tensions (2011–2013), the pandemic collapse (2020) with the 2021 rebound, and the energy-price spike (2022). These swings motivate a specification that separates a persistent component from short-run noise while preserving level information.

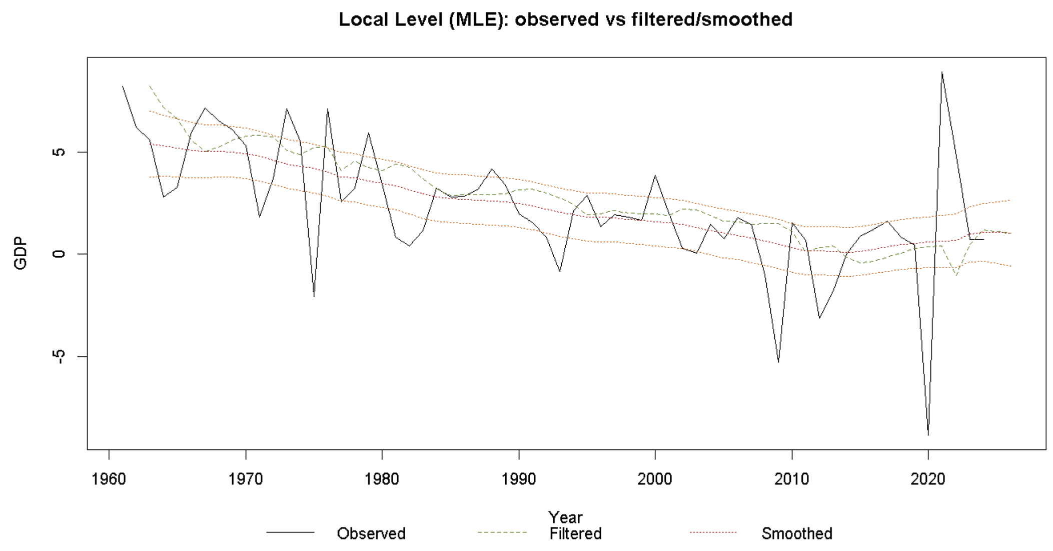


Figure 3.3: Local Level (maximum–likelihood estimation).

The green dashed path is the filtered level, computed recursively by the Kalman filter using only information up to time t; it delivers a real-time signal that updates cautiously when large shocks arrive. The red dotted path is the fixed-interval smoothed level; by using the full sample it revises past values in light of subsequent observations and reveals a slower underlying trajectory that absorbs much of the high-frequency variation. The orange bands are approximate 90% intervals from the Gaussian smoothing distribution; they tighten in calmer stretches and widen near endpoints and major disturbances. The signal–to–noise ratio W/V governs how quickly the latent level is allowed to move relative to measurement noise: higher values make the level more agile (and bands wider), lower values enforce greater smoothness. Before assessing integration properties, it is useful to inspect the serial-dependence pattern of the observed series.

Serial–dependence patterns. As a quick diagnostic, the aim is to assess whether persistence is predominantly low frequency (latent level) or short run. Here, *short run* denotes dependence over a small number of immediately preceding lags; in Box–Jenkins terms, short-run autoregressive (AR) effects correspond to an $\mathrm{AR}(p)$ with small p and are typically indicated by a geometric decay in the ACF with a sharp cut-off in the PACF at lag p. By contrast, *low-frequency* persistence refers to slowly varying behaviour driven by a latent component that evolves over time (a stochastic level or trend); in sample correlograms this appears as a gradual, non-abrupt decline of the ACF and a PACF without a clear truncation point, reflecting diffuse dependence across many lags. Figure 3.4 presents the autocorrelation (ACF) and partial autocorrelation (PACF) of GDP growth.

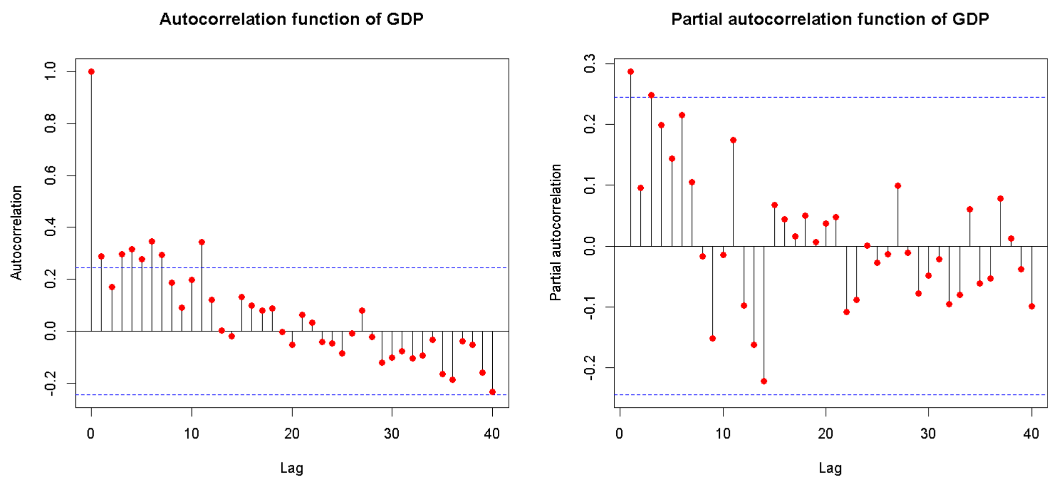


Figure 3.4: ACF and PACF of Italian real GDP growth.

From Figure 3.4, it is apparent that the correlation structure is dominated by low-frequency persistence rather than short-run dynamics. The ACF declines gradually rather than cutting off quickly, indicating persistence more naturally captured by an evolving latent component. The PACF shows only a few notable spikes without a clear truncation pattern, suggesting that short-run autoregressive effects are limited relative to the contribution of the stochastic level. This reading supports a parsimonious LL specification and leads into a check of unit-root behaviour.

Stationarity and integration. Complementary unit-root diagnostics on levels and first differences provide *borderline evidence at levels* and a clearer picture in differences, as summarised in Table 3.1. Tests are run with an intercept (no deterministic trend), using standard data-driven choices for lag length (ADF) and bandwidth (KPSS).

Table 3.1: Stationarity tests on GDP growth

	ADF (level)	KPSS (level)	ADF (diff)	KPSS (diff)
Statistic	-3.480	1.231	-6.539	0.057
p-value	0.051	0.010	0.010	0.100

On first differences, ADF rejects a unit root and KPSS does not reject stationarity. Rather than treating these outcomes as a mechanical classification, these results are read as evidence of a persistent low-frequency component for which a stochastic-level representation is appropriate—consistent with the ACF/PACF in Figure 3.4 and with the LL fit above. In such cases, it is prudent to avoid overdifferencing. By overdifferencing is meant applying additional differences beyond what is needed for stationarity; this tends to remove meaningful low-frequency variation, inflate short-run noise, and complicate interpretation. A more transparent approach is to use complementary diagnostics and adopt a specification that can accommodate persistence without sacrificing interpretability—here, the Local Level model. With the specification fixed, the next step is to quantify smoothness on the data scale through the *Signal-To-Noise Ratio*.

Signal–to–Noise Ratio To anchor the discussion, maximum–likelihood estimates are reported for the variance components of the Local Level specification together with large–sample standard errors and the implied signal–to–noise ratio (SNR). The observation variance V captures measurement noise around the latent level, whereas the state–innovation variance W controls how much the level is allowed to move between periods. All variances are on the data scale of *squared percent*, i.e. $(\%)^2$, because the observable y_t is a year-over-year percent change. Standard errors are computed by (i) estimating on the log-variance scale $\eta = (\log V, \log W$; (ii) inverting the observed information matrix of the Kalman likelihood at $\hat{\eta}$ to obtain $\mathrm{Var}(\hat{\eta})$; and (iii) mapping back to (V, W, SNR) via the delta method using the appropriate Jacobians (see Chapter 2.1.2). Table 3.2 summarizes the estimates and the resulting precision.

Table 3.2: Local Level: Parameters

Parameter	Estimate	Std. Error
V	6.136	1.159
W	0.178	0.135
$\mathrm{SNR} = W/V$	0.029	0.023

Numerically, the observation variance is about $6.1\ (\%)^2$, the state–innovation variance about $0.18\ (\%)^2$, and $\mathrm{SNR} \approx 0.03$. Measurement noise therefore dominates state evolution: the filtered path updates cautiously in real time, and the smoothed level is *markedly persistent* in the sense that the latent level drifts only slowly over time when $W \ll V$ (small Kalman gain), with 90% smoothing bands that tighten in calm stretches and widen near endpoints and large shocks. Interpreted this way, W/V acts as a single dial for smoothness—higher values allow the level to track data more closely at the cost of wider bands, lower values enforce greater persistence.

The next sections develop two complementary inferential routes on this fixed specification—frequentist (plug-in) and Bayesian (posterior predictive)—to quantify uncertainty around states and forecasts. Before turning to estimation results, the *forecast evaluation design and metrics* are introduced to provide a unified reading of the subsequent evidence (fixed holdout versus rolling-origin, accuracy and calibration measures, and their horizon profiles).

3.3 Forecast evaluation design and metrics

This section defines the evaluation step that follows uncertainty quantification and precedes the presentation of estimation results. The goal is to fix a single protocol—common to both inferential routes—so that subsequent comparisons are unambiguous.

Conventional k–fold validation is unsuitable for forecasting because it breaks temporal order and may leak future information into the training set. Evaluation therefore preserves chronology through two complementary designs. A *fixed holdout* reserves the last h observations and fits the model on the preceding span. A *rolling-origin* design mimics real-time use: select forecast origins $\tau_1 < \cdots < \tau_M$ near the end of the sample; at each origin τ, re-estimate the model on the expanding window $Y_{1:\tau}$ and produce s-step-ahead forecasts for $s = 1, \ldots, h$.

For annual data with last observed year $T = 2024$ and horizon $h = 4$, typical origins are $\tau \in \{2016, \ldots, 2020\}$ so that $\tau + 4 \leq T$. The mapping from origins to target years is:

Table 3.3: Rolling-origin mapping for annual data ($h = 4$, last year $T = 2024$).

Origin τ	$s = 1$	$s = 2$	$s = 3$	$s = 4$
2016	2017	2018	2019	2020
2017	2018	2019	2020	2021
2018	2019	2020	2021	2022
2019	2020	2021	2022	2023
2020	2021	2022	2023	2024

At each pair (origin τ, horizon s) the model yields a predictive distribution $p(y_{\tau+s} \mid Y_{1:\tau})$, summarized by mean $f_{\tau,s}$, variance $Q_{\tau,s}$, and a central $(1-\alpha)$ interval $[\ell_{\tau,s}, u_{\tau,s}]$ (with $\alpha = 0.10$ unless stated). Aggregate point accuracy is

$$\mathrm{RMSE} = \sqrt{\frac{1}{Mh}\sum_{\tau}\sum_{s=1}^{h}(\widehat{y}_{\tau,s} - y_{\tau,s})^2}, \qquad \mathrm{MAE} = \frac{1}{Mh}\sum_{\tau}\sum_{s=1}^{h}|\widehat{y}_{\tau,s} - y_{\tau,s}|,$$

with $\widehat{y}_{\tau,s} = f_{\tau,s}$. Calibration and sharpness are read from empirical coverage and average width,

$$\mathrm{Cov}_{1-\alpha} = \frac{1}{Mh}\sum_{\tau}\sum_{s=1}^{h}\mathbf{1}\{\ell_{\tau,s} \leq y_{\tau,s} \leq u_{\tau,s}\}, \qquad \mathrm{AvgWidth} = \frac{1}{Mh}\sum_{\tau}\sum_{s=1}^{h}\big(u_{\tau,s} - \ell_{\tau,s}\big),$$

while horizon profiles $\mathrm{RMSE}(s)$ and $\mathrm{Cov}_{1-\alpha}(s)$ characterize how accuracy and coverage evolve with distance and are defined by fixing s and averaging over τ.

Proper scoring rules assess full predictive distributions. The logarithmic score for a single forecast–outcome pair is

$$\mathrm{LogS}(p, y) = \log p(y),$$

and the quantity reported in tables/figures is its average over all forecast origins and horizons, i.e.

$$\mathrm{LogS} = \frac{1}{Mh} \sum_{\tau,s} \log p\big(y_{\tau,s}\big),$$

and under a Gaussian forecast $y \sim \mathcal{N}(f, Q)$: $\quad \log p(y) = -\frac{1}{2}\left(\log(2\pi) + \log Q + \frac{(y-f)^2}{Q}\right),$

so the empirical mean simply plugs in $(y_{\tau,s}, f_{\tau,s}, Q_{\tau,s})$ inside the summation. Higher is better. The continuous ranked probability score (CRPS) compares the entire predictive CDF F with the empirical step at y,

$$\mathrm{CRPS}(F, y) = \int_{-\infty}^{\infty} \big(F(z) - \mathbf{1}\{z \geq y\}\big)^2 \, dz,$$

and is estimated by Monte Carlo from predictive draws; lower is better.

In the frequentist plug-in approach, $p(\cdot)$ is the conditional Gaussian predictive implied by Kalman recursions at fixed estimates $(\widehat{V}, \widehat{W})$, requiring one filter–smoother pass per origin. In the Bayesian approach, $p(\cdot)$ is the posterior predictive mixture obtained by averaging the same recursions over draws of parameters (and, when needed, terminal states) via Gibbs/FFBS, trading additional computation for coherent propagation of parameter uncertainty. This shared protocol anchors all subsequent comparisons under the common Local Level specification, so that differences can be ascribed to the inferential stance rather than to the evaluation design.

3.4 Uncertainty quantification

This section fixes how uncertainty is handled under a common Local Level specification. Two complementary routes are considered. The frequentist route estimates the variance components by maximum likelihood and carries out filtering, smoothing, and forecasting at fixed hyperparameters (*plug–in*). The Bayesian route treats the same hyperparameters as random, combines weakly informative priors with the Kalman likelihood, and integrates parameter uncertainty in states and forecasts via posterior prediction. Both routes feed into the unified evaluation protocol defined in Section 3.3.

3.4.1 Frequentist estimation

The frequentist approach treats the hyperparameters of the Local Level state–space model as fixed but unknown and estimates them by maximum likelihood. Under the specification in Section 3.2, the Gaussian log–likelihood is evaluated via Kalman recursions with computational cost linear in the sample size. Positivity is enforced by parameterizing the observation variance V and the level–innovation variance W on the log scale; large–sample standard errors and confidence intervals are obtained from the observed information for $(\log V, \log W)$ and mapped back to the variance scale by the delta method.

Forecasting, filtering, and smoothing then proceed under a *plug–in* convention embedded in the workflow. Let $\phi = (V, W)$ and let $\hat{\phi}$ denote the maximum–likelihood estimate. Operations are run *as if* $\phi = \hat{\phi}$, so one–step prediction and variance at time t are

$$\hat{f}_t = f_t(\hat{\phi}), \qquad \hat{Q}_t = Q_t(\hat{\phi}),$$

with standardized innovations

$$z_t = \frac{y_t - \hat{f}_t}{\sqrt{\hat{Q}_t}}.$$

Multi–step forecasts condition on the same fixed $\hat{\phi}$,

$$y_{t+h|t} \sim \mathcal{N}\big(f_{t+h|t}(\hat{\phi}),\, Q_{t+h|t}(\hat{\phi})\big), \qquad Q_{t+h|t}(\hat{\phi}) = F\, P_{t+h|t}(\hat{\phi})\, F^{\top} + V(\hat{\phi}),$$

so prediction bands follow from Gaussian formulas without additional simulation. The convention is operationally attractive—one filter–smoother pass per fit—and provides transparent diagnostics, while leaving parameter uncertainty unaccounted for.

Given the estimates $(\widehat{V}, \widehat{W})$, the filtered states $\hat{\theta}_{t|t}$ deliver real–time readings and the smoothed states $\hat{\theta}_{t|T}$ provide two–sided refinement based on the full sample. Forecasts $y_{t+h|t}$ are obtained in closed form, with predictive variance

$$\mathrm{Var}(y_{t+h|t}) = F\,\mathrm{Var}(\theta_{t+h|t})F^{\top} + \widehat{V}$$

computed at $(\widehat{V}, \widehat{W})$. A single quantity summarizes smoothness on the data scale, $\widehat{W}/\widehat{V}$ (signal–to–noise): larger values permit a more agile latent level, smaller values enforce greater persistence. Under standard regularity conditions, maximum–likelihood estimators are consistent and asymptotically normal, Gaussian forecasts are mean–square optimal, and occasional missing observations are handled natively by the filter and smoother without pre–imputation.

The frequentist plug–in fit thus provides point trajectories, residual diagnostics, and predictive bands under fixed hyperparameters.

Residual diagnostics

Residual behaviour is assessed on the *standardized one–step–ahead innovations* from the frequentist plug–in fit. Under correct specification, $\{z_t\}$ should be approximately i.i.d. $\mathcal{N}(0, 1)$. Tests target serial independence and normality; the corresponding statistics are reported first, followed by graphical evidence.

Table 3.4: Local Level: residual tests on standardized one–step innovations z_t.

	Ljung–Box(20)		Shapiro–Wilk		Jarque–Bera	
Model	stat	p	W	p	stat	p
LL	18.344	0.565	0.893	$< 10^{-4}$	59.065	$< 10^{-12}$

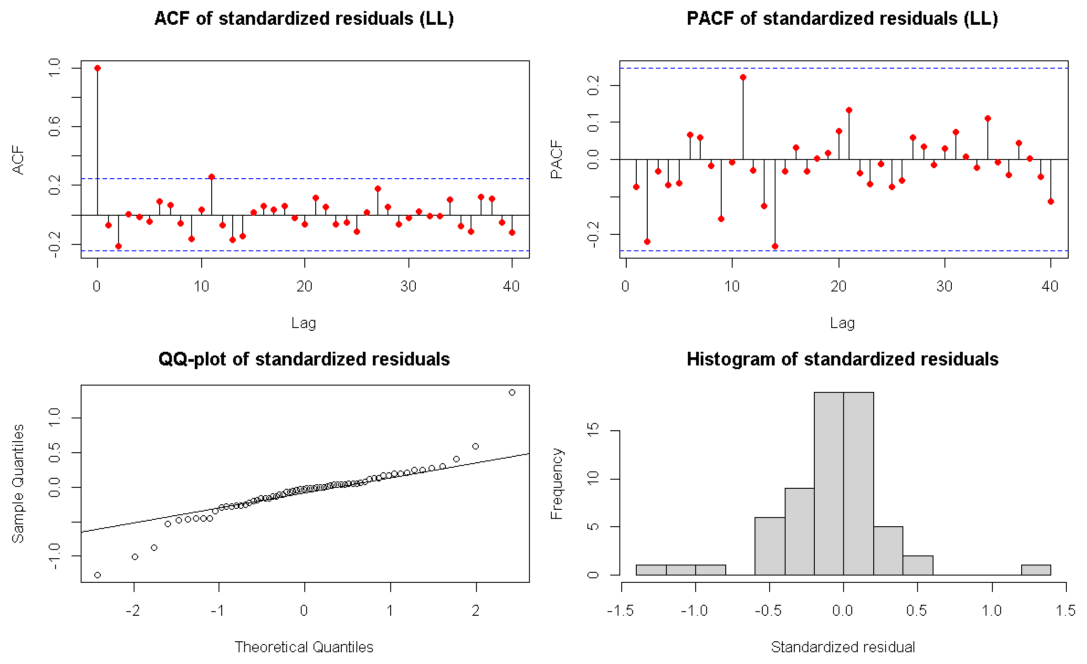


Figure 3.5: Standardized innovation plot with Local Level

Read together, the Ljung–Box(20) does not reject serial independence, while Shapiro–Wilk and Jarque–Bera reject normality. The correlograms show no persistent pattern across lags, whereas the QQ–plot and histogram reveal mild skewness and heavy tails. Such departures are unsurprising given the shock–rebound episodes in the sample and help explain occasional under–coverage of nominal 90% bands around turbulent windows (e.g., 2020–2021). Overall, residuals are white but non–Gaussian: finite–sample bands can widen in extremes, yet the stochastic–trend DLM remains a coherent basis for inference and forecasting in ordinary periods.

Model assessment for forecasting

This subsection connects model selection and forecasting performance within a single framework. The first part compares alternative specifications in–sample to identify the operative model for this dataset; the second part evaluates out–of–sample behavior under the unified protocol of Section 3.3.

In–sample model comparison. Allowing a stochastic slope, as in the Local Linear Trend (LLT), spreads movements through momentum but adds an extra variance component. On the full sample, information criteria favor parsimony: the Local Level (LL) attains a higher likelihood with fewer parameters, yielding markedly lower AIC and BIC than the trend variant.

Table 3.5: Model selection on the full sample.

Model	LogLik	AIC	BIC
Local Level	-102.678	209.356	213.674
Local Linear Trend	-111.877	229.754	236.231

The log–likelihood gap is 9.199, corresponding to a likelihood ratio $\exp(9.199) \approx 9{,}900$ in favor of LL. Accounting for complexity ($k{=}2$ for LL; $k{=}3$ for LLT; $T{=}64$), the AIC and BIC differences are $\Delta\text{AIC} = 20.398$ and $\Delta\text{BIC} = 22.557$, both large on standard scales. The implied Akaike weight concentrates on LL (about 0.99996). The BIC gap can be read on a Bayes–factor scale via the Schwarz approximation, $\text{BIC} \approx -2\log m(Y) + \text{const}$, so that $\Delta\text{BIC} \approx -2\log\text{BF}$ and $\text{BF} \approx \exp(-\frac{1}{2}\Delta\text{BIC}) \approx 1.3 \times 10^{-5}$ against the added slope. Overall, LL provides the more efficient and credible description for this series; the stochastic slope tends to diffuse turning points without compensating gains in fit or forecast accuracy. As an external reduced–form yardstick, ARIMA$(0, 1, 1)$ aligns closely with LL at one step: the estimated MA(1) is -0.84374, the one–step RMSE is 2.676, and one–step forecasts from LL and ARIMA correlate at 0.9807.

Out–of–sample evidence. The objective here is to assess forecasting performance under designs that preserve time order. Two complementary settings are used: a fixed eight–step holdout and a rolling–origin design with horizon $h = 4$, with accuracy, calibration, and sharpness summarized by RMSE, MAE, empirical coverage, average width, and proper scores as defined in Section 3.3.

Figure 3.6 displays the rolling fan for the LL over 2014–2024, a window chosen to include both tranquil years and the shock–rebound sequence around the pandemic.

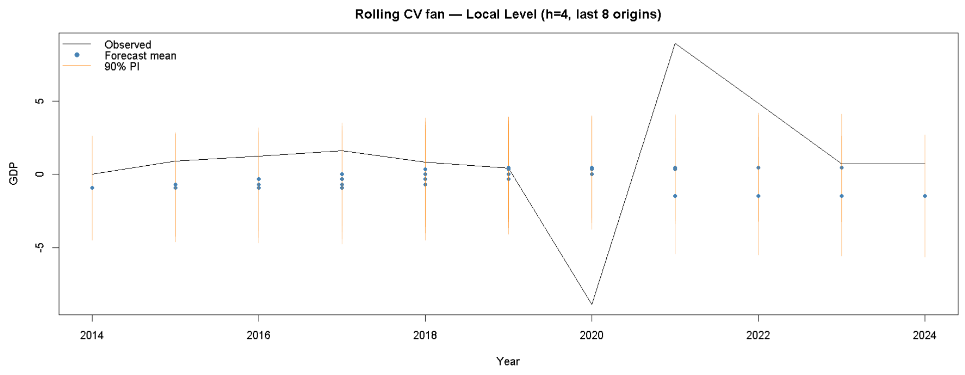


Figure 3.6: Rolling–origin fan chart ($h = 4$): Local Level.

Blue dots mark forecast means and orange bars the 90% prediction intervals at horizons $s = 1, \ldots, 4$. Means remain close to the underlying level in calm periods and re–center progressively after large shocks; bands widen with horizon and expand visibly in turbulent windows, while staying relatively tight elsewhere.

Table 3.6 aggregates the fixed holdout and the rolling exercise.

Table 3.6: Local Level (LL): forecast summary—holdout (8 steps) and rolling cross–validation ($h = 4$).

	Holdout (8)			Rolling CV ($h = 4$)		
Model	RMSE	MAE	Cov90	RMSE	MAE	Cov90
LL	4.821	3.357	0.625	4.972	3.622	0.656

Errors are moderate in both designs; empirical coverage falls short of the 90% target in windows dominated by extreme shocks, consistent with heavy–tailed, non–Gaussian disturbances; average width increases with horizon as expected.

Figure 3.7 reports the eight–step holdout path. The forecast mean adopts a cautious profile consistent with the low signal–to–noise ratio, and the 90% intervals widen into the pandemic and then narrow as conditions normalize. The combination illustrates the frequentist approach's ability to track medium–term dynamics with sharp, closed–form predictive bands, while revealing the expected under–dispersion in extreme years.

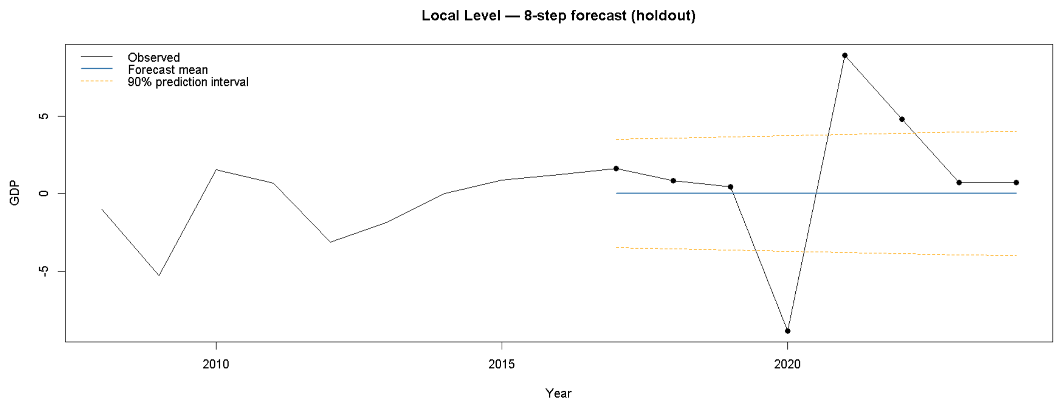


Figure 3.7: Holdout forecasts (eight–step horizon): Local Level.

Taken together, the in–sample and out–of–sample evidence points to the Local Level as the operative specification for this dataset and documents its forecasting profile across regimes. The analysis now turns to the treatment of missing spans within the same state–space machinery, beginning with the backcasting exercise in the next subsection.

Handling missing data: backcasting a pre–sample block (frequentist plug–in)

This exercise is *backcasting*, not backtesting. Backcasting imputes a block of *truly missing* observations that occur *before* the observed sample begins, producing model–based reconstructions with uncertainty on the data scale. *Backtesting*, by contrast, evaluates a model's forecasts on a held–out period that *follows* the estimation window, preserving time order and comparing predictions with realized values rather than imputing unobserved data. The aim here is to extend the historical picture in a disciplined way using the state–space machinery, while quantifying uncertainty in the same units as the series.

The Local Level model is estimated by maximum likelihood on the observed span (1960–2024). The pre–sample years 1950–1959 are treated as unobserved and recovered by the fixed–interval Kalman smoother (Rauch–Tung–Striebel). With parameters fixed at their MLEs $(\widehat{V}, \widehat{W_\ell})$, the smoothing step yields the latent–level distribution for each missing year t,

$$\mu_t \mid Y_{1:T},\, \widehat{V},\, \widehat{W_\ell} \;\sim\; \mathcal{N}\big(\widehat{\mu}_{t|T},\, \widehat{P}_{t|T}\big),$$

and the implied *observable–scale* predictive distribution adds measurement noise,

$$y_t \mid Y_{1:T},\, \widehat{V},\, \widehat{W_\ell} \;\sim\; \mathcal{N}\big(\widehat{\mu}_{t|T},\, \widehat{P}_{t|T} + \widehat{V}\big).$$

In the Local Level case $F = 1$, so the variance reduces to the smoothed–state variance plus $\widehat{V}$. Central 90% prediction intervals follow directly from these Gaussian summaries.

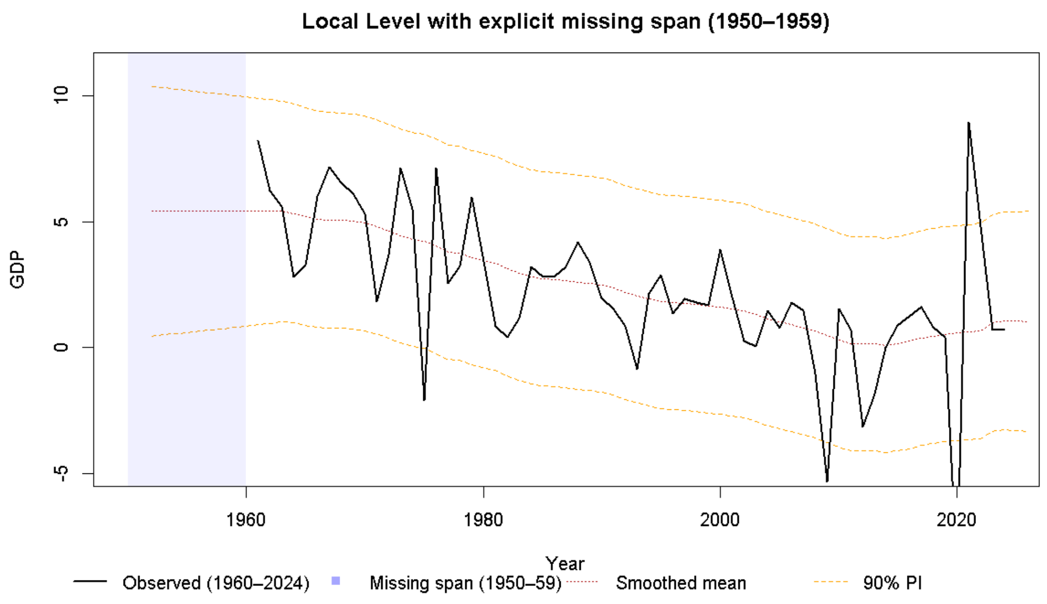


Figure 3.8: Backcast 1950–1959 under the Local Level (frequentist plug–in).

Three features help interpret Figure 3.8. First, *persistence*: a small signal–to–noise ratio $\widehat{W_\ell}/\widehat{V}$ forces the latent level to move gradually, so the central path in the 1950s is nearly flat and anchored to early–1960s values. Second, *distance from data*: bands widen smoothly when moving backward from 1960 to 1950 because the smoothed–state variance grows away from observed years; on the observable scale, the addition of $\widehat{V}$ makes these bands wider than state–credible bands. Third, *economic reading*: the fan stays largely in positive territory toward the late 1950s, consistent with a steady post–war expansion under this specification.

Because parameters are fixed at MLEs, these intervals do not include parameter uncertainty and are typically *narrower* than fully Bayesian predictive bands; in turbulent regimes or under heavy tails they may under–cover. The backcast should be read as a *model–implied imputation* that extends the graph without ad hoc devices, clarifies how uncertainty grows before the first observation, and checks boundary behaviour under the Local Level specification.

Frequentist summary

The frequentist route to uncertainty quantification combines computational efficiency, analytical tractability, and transparent diagnostics. Estimation by maximum likelihood via Kalman recursions is fast and scales linearly with the sample size; plug–in filtering and smoothing provide closed–form forecasts and prediction bands without additional simulation. The approach is therefore attractive when interpretability, simplicity, and speed are central, and it delivers estimates that are consistent, asymptotically normal, and mean–square optimal under Gaussian assumptions. At the same time, treating parameters as fixed introduces limitations: predictive bands may under–represent uncertainty, coverage can deteriorate in heavy–tailed or turbulent episodes, and evaluation under holdout or rolling–origin schemes highlights sensitivity to neglected parameter risk.

In short, the frequentist framework supplies an efficient baseline: it captures persistence, produces credible short–horizon predictability, and clarifies the trade–off between smoothness and noise, but it does so under fixed hyperparameters. The next section turns to Bayesian estimation, which retains the same model structure but integrates parameter uncertainty directly through posterior prediction, thereby extending the analysis of uncertainty beyond the plug–in convention.

3.4.2 Bayesian Estimation

The Bayesian approach retains the Local Level (LL) specification used in the frequentist analysis and changes only how uncertainty is propagated. The observation variance V and the level–innovation variance W are treated as random and assigned weakly informative priors,[2] so that the likelihood remains the dominant source of information. Inference proceeds through simulation of the joint posterior for (V, W) and for the latent path $\{\mu_t\}$, based on the same linear–Gaussian likelihood computed by Kalman recursions.

A key distinction from the frequentist plug–in route is the treatment of predictive distributions. Under plug–in, forecasts condition on a fixed point estimate $\hat{\phi} = (\widehat{V}, \widehat{W})$, so uncertainty arises only from the state–space propagation. The Bayesian route instead integrates over parameter uncertainty: letting $p(\phi \mid Y_{1:t})$ be the posterior of $\phi = (V, W)$, the h–step predictive is the mixture

$$p(y_{t+h} \mid Y_{1:t}) = \int \mathcal{N}\big(f_{t+h|t}(\phi),\, Q_{t+h|t}(\phi)\big)\, p(\phi \mid Y_{1:t})\, d\phi,$$

approximated in practice by Monte Carlo draws. Forecast bands and scoring rules are therefore based on posterior predictive distributions rather than on fixed–parameter conditionals. This distinction is central in the comparative forecast evaluation that follows.

[2]A conjugate Gibbs scheme is employed, with shape–scale hyperparameters chosen so that $\mathbb{E}[V] \approx \mathrm{var}(y)$ and $\mathbb{E}[W] \approx 0.1\,\mathrm{var}(y)$.

Posterior computation and convergence

Posterior sampling alternates Gibbs updates of (V, W) with draws of the full latent state path via Forward–Filtering Backward–Sampling (FFBS).[3] The sampler runs for $12{,}000$ iterations with $2{,}000$ burn–in and thinning by 5, yielding about $2{,}000$ effectively independent draws for V, W, and representative states. Monte Carlo errors for posterior means are negligible.

Convergence is monitored with multiple chains from dispersed seeds. The Gelman–Rubin statistic $\widehat{R}$ compares between–chain and within–chain variability; values close to one indicate that independent chains explore the same distribution. For both V and W the point estimates and 97.5% bounds of $\widehat{R}$ are essentially 1.00, comfortably below common thresholds such as 1.05. Trace plots show chains fluctuating steadily around central values with no drifts, and autocorrelation functions decay rapidly within a few lags. Together with effective sample size diagnostics, these features support convergence and low serial dependence (Figure 3.9).

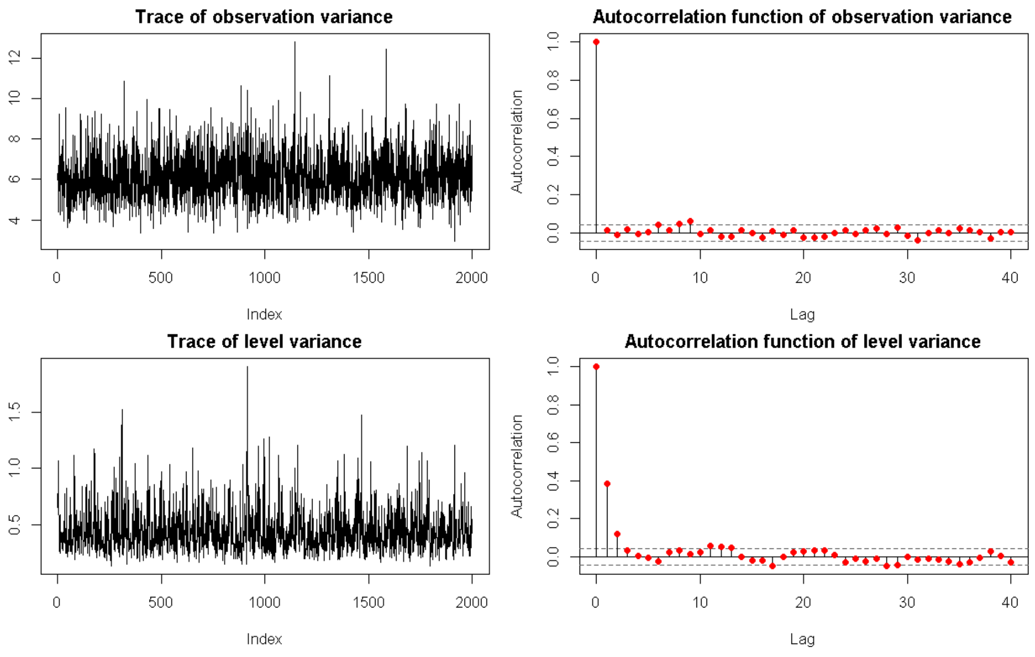


Figure 3.9: Posterior diagnostics for variance components: trace plots and autocorrelation functions for V and W.

Variance components, SNR, and level

Posterior mass concentrates on parameter configurations where measurement noise dominates movements in the latent level. Table 3.7 reports posterior means, standard deviations,

[3]In the Local Level model the full conditionals for V and W are inverse–gamma, with hyperparameters set so that their prior means are anchored to the data variance. Given (V, W), FFBS generates the full path $\{\mu_t\}$ exactly from its smoothing law by first filtering forward and then sampling backward.

and central credible intervals for the observation variance V, the level–innovation variance W, and their ratio $\mathrm{SNR} = W/V$.

Table 3.7: Posterior summaries for V, W, and $\mathrm{SNR} = W/V$ under the local level.

	mean	sd	$q_{2.5}$	q_{50}	$q_{97.5}$
V	6.089	1.205	4.091	5.935	8.812
W	0.439	0.189	0.199	0.400	0.897
W/V	0.075	0.038	0.030	0.067	0.168

Posterior means imply a steady–state Kalman gain of $K \approx 0.24$, so each new observation receives about one quarter of the weight of the one–step prediction.[4] This small gain explains the cautious, smooth filtered path. One–step predictive variance is dominated by V: a back–of–the–envelope calculation gives

$$\mathrm{Var}(y_{t+1} \mid Y_t) \approx V + W \approx 6.53,$$

so a central 90% Gaussian interval has width about $2 \times 1.645 \times \sqrt{6.53} \approx 8.4$ (squared percentage–points scale), consistent with the widths observed in forecast exercises. Posterior quantiles show the mild right–skewness typical of inverse–gamma laws (means slightly above medians), but the closeness of means and medians indicates that the likelihood dominates the weakly informative priors.

[4]For a random–walk level with observation variance V and innovation variance W, the steady–state state–error variance P solves $P^2 - WP - WV = 0$, giving $P = \{W + \sqrt{W^2 + 4WV}\}/2$ and Kalman gain $K = P/(P + V)$. Plugging in posterior means $V = 6.089$ and $W = 0.439$ yields $P \approx 1.87$ and $K \approx 0.24$.

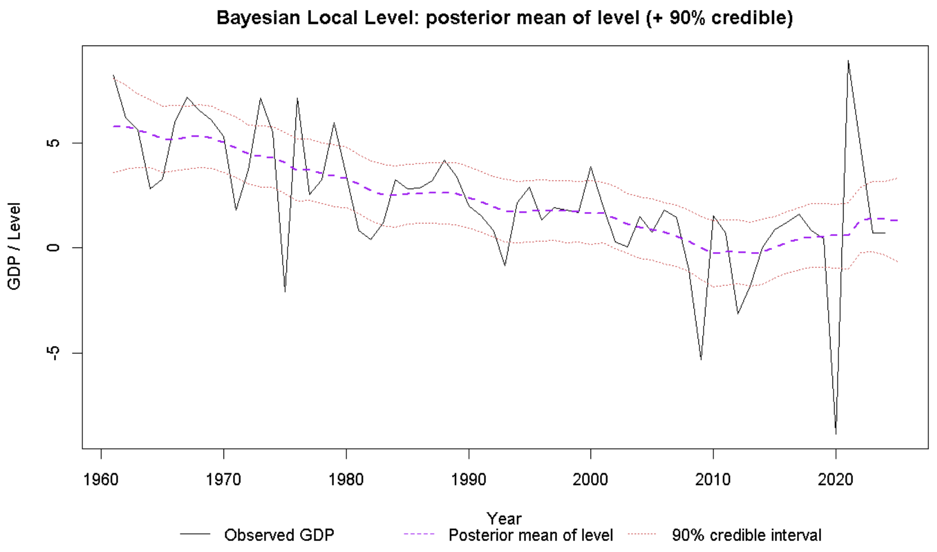


Figure 3.10: Posterior mean level and 90% credible band under the Bayesian local level.

Three features stand out. First, *smoothness and caution*: the posterior level follows low–frequency trends rather than short–run volatility, the graphical counterpart of the small SNR. Second, *heteroskedastic credible envelope*: bands tighten in tranquil stretches and widen near large shocks and at the sample boundaries, reflecting stronger or weaker information. Third, *long–run level*: the path drifts gradually lower and remains near zero in the last decade, consistent with the frequentist smoother but now accompanied by credible bands that incorporate parameter uncertainty.

Residual diagnostics

Residual checks are based on standardized one–step–ahead errors computed at the posterior–mean plug–in $(\bar{V}, \bar{W})$, to keep diagnostics comparable with the frequentist fit; full posterior–predictive checks can be added by drawing $(V, W, \{\mu_t\})$ and replicated series $\{\tilde{y}_t\}$ for tail–sensitive statistics, without altering the main conclusions.

The Ljung–Box(20) test does not reject serial independence ($p = 0.454$), while departures from Gaussianity appear in heavier tails. Figure 3.11 shows that autocorrelations lie within nominal 95% bounds after the first lags and no persistent pattern emerges in the PACF, confirming short memory. Whiteness supports the adequacy of a stochastic level, while heavy tails help explain occasional under–coverage of nominal bands in turbulent windows.

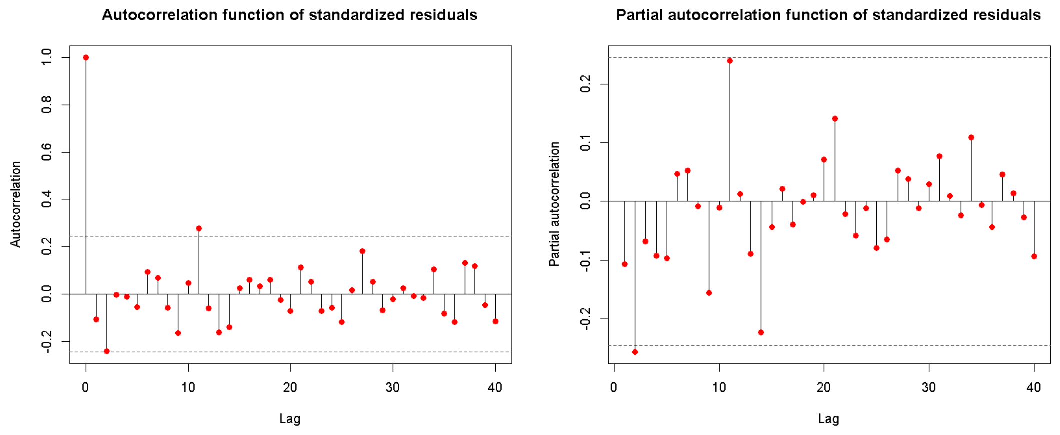


Figure 3.11: Standardized one–step residuals under the Bayesian plug–in fit: ACF and PACF with 95% bands.

Posterior predictive checks based on replicated datasets can be added to explicitly assess tail behavior, but they do not alter the main conclusions here.

Predictive evaluation: holdout and rolling origin

Forecast performance is assessed with the same holdout and rolling–origin designs used in the frequentist analysis, so that results are directly comparable. The difference lies in the predictive distribution: here it is posterior predictive, integrating over (V, W) and, when needed, terminal states. At each origin, draws are propagated forward and summaries are obtained from the resulting Monte Carlo distribution.

Fixed holdout ($h = 8$). The predictive mean path remains close to flat, with bands widening around turbulent episodes. Figure 3.12 shows the holdout forecasts, and Table 3.8 reports summary metrics. Errors are moderate (RMSE $= 4.799$, MAE $= 3.272$), coverage (0.625) falls short of nominal due to heavy–tailed shocks, and average width (8.346) signals sharp distributions.

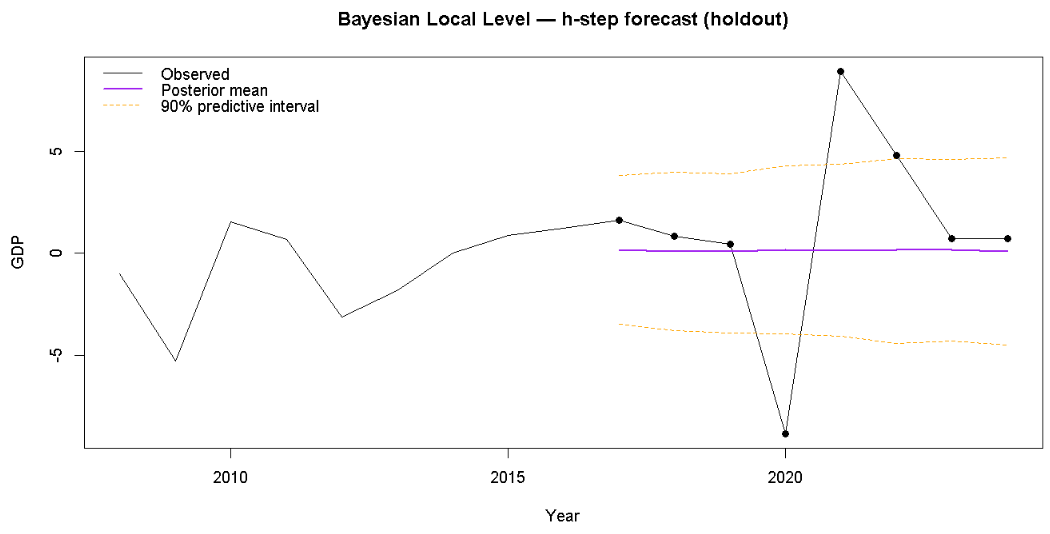


Figure 3.12: Eight–step holdout forecasts: Bayesian local level.

Table 3.8: Posterior predictive summary on an eight–step holdout: Local Level (LL).

Model	RMSE	MAE	Cov90	AvgWidth
Local Level	4.799	3.272	0.625	8.346

Rolling–origin design ($h = 4$)**.** Figure 3.13 shows the fan chart obtained by recomputing the posterior at each origin and projecting $s = 1, \ldots, 4$ steps ahead. Bands expand predictably with horizon and around the shock–rebound sequence, while remaining tight in tranquil windows. Table 3.9 reports aggregate performance: RMSE and MAE are broadly consistent with the holdout, coverage is again below nominal (0.656), and average width (7.85) remains modest. The pattern indicates predictive distributions that are sharp yet somewhat under–dispersed in turbulent episodes, consistent with heavy–tailed disturbances.

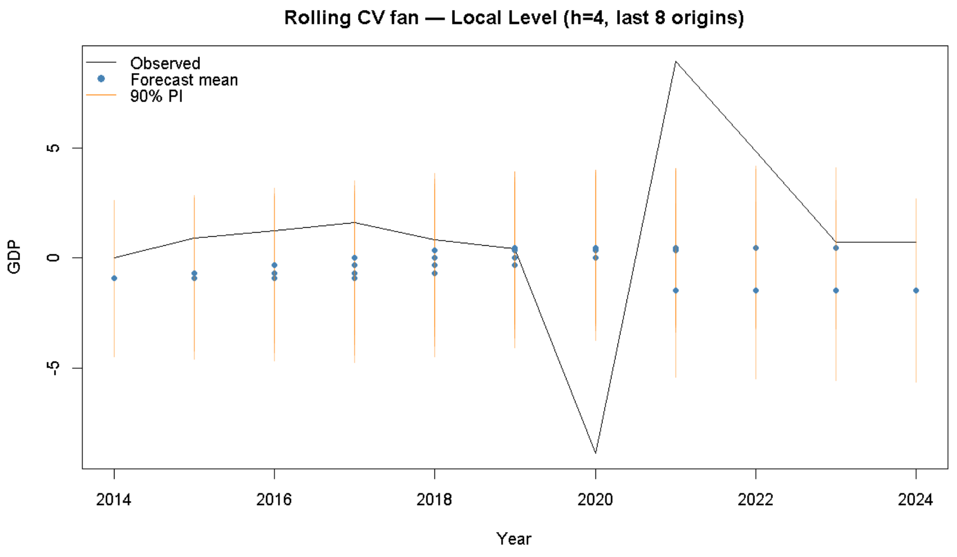


Figure 3.13: Rolling–origin fan chart ($h = 4$): Bayesian local level

Table 3.9: Rolling–origin summary for the Bayesian local level ($h = 4$).

	RMSE	MAE	Cov90	AvgWidth
Aggregate	5.098	3.738	0.656	7.848

Overall, the Bayesian LL forecasts confirm the frequentist reading: smooth underlying momentum, sharp short–horizon predictability, widening bands with the horizon, and under–coverage in turbulent windows. The difference lies in the explicit propagation of parameter uncertainty, which delivers coherent predictive distributions and makes the posterior predictive route a natural complement to the plug–in frequentist approach.

Backcasting the pre–sample block (1950–1959)

The mechanics of backcasting under the frequentist approach have already been described; the Bayesian version differs in that parameter uncertainty is integrated rather than fixed at point estimates.

For each posterior draw $(V^{(m)}, W^{(m)})$, Forward–Filtering Backward–Sampling (FFBS) delivers a full path $\{\mu_t^{(m)}\}$ of the latent level. On the observable scale, each path is combined with $V^{(m)}$ to produce draws from the predictive distribution,

$$y_t^{(m)} \sim \mathcal{N}\Big(\mu_t^{(m)},\ \mathrm{Var}(\mu_t^{(m)}) + V^{(m)}\Big), \qquad t = 1950, \ldots, 1959.$$

Averaging across draws yields posterior predictive means and central bands. Unlike the plug–in frequentist calculation, these bands account for parameter uncertainty and are generally wider, especially under weak identification of W/V.

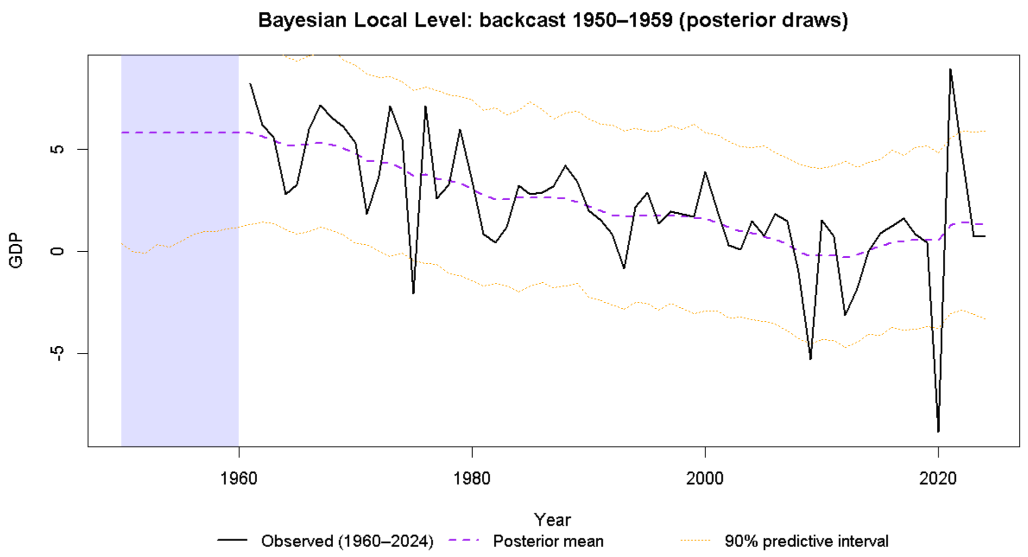


Figure 3.14: Bayesian backcast 1950–1959 under the Local Level model.

Three features are apparent. First, *persistence and caution*: the central path remains nearly flat and drifts only slowly toward 1960, reflecting the small posterior SNR that suppresses rapid adjustment. Second, *uncertainty growth*: posterior predictive bands widen smoothly as distance from observed data increases, with measurement noise V making them wider than state–credible bands. Third, *economic reading*: the fan remains largely positive through the late 1950s, consistent with a steady post–war expansion, while extreme contractions are given low posterior probability.

Bayesian backcasting thus produces a disciplined structural imputation that extends the historical graph with coherent uncertainty quantification. Its added value over the plug–in approach lies in carrying parameter uncertainty into the predictive envelopes, though the same limitations apply under non-Gaussian tails or regime changes, where Student–t innovations or predictive bootstrap bands offer natural robustness checks.

Bayesian summary

The Bayesian route keeps the Local Level specification fixed and changes only how uncertainty is encoded and propagated. Variance components and the latent path are treated as random, combined with weakly informative priors and the same Kalman likelihood, and explored by simulation (Gibbs updates for the variances coupled with Forward–Filtering Backward–Sampling for the states). Forecasts, fan charts, and backcasts are then read from the *posterior predictive* distribution, so parameter risk is integrated alongside state uncertainty. Computationally, the workflow amounts to repeated Kalman passes and scales linearly with the sample size; convergence is monitored with standard MCMC diagnostics, and effective sample sizes are sufficient for stable summaries.

This stance proves efficient when the aim is to report calibrated probabilities: fan charts reflect both measurement noise and uncertainty about smoothness, and predictive scores are coherent because they are evaluated under the same posterior law that generated the forecasts. It remains transparent—everything is expressed on the observable scale—and robust to modest data frictions, such as short windows or missing blocks, because posterior averaging stabilizes weakly identified variance components. The main trade-offs are the extra computation relative to plug-in filtering, and a mild sensitivity to prior anchoring when evidence is thin; under Gaussian likelihoods, heavy-tailed shocks can still compress nominal coverage unless the observation equation is made more robust.

In sum, the Bayesian approach provides a fully probabilistic account of the same signal extracted by the frequentist baseline: it carries parameter uncertainty into states, forecasts, and backcasts with limited computational overhead and without altering the model. This completes the uncertainty–quantification step; the analysis that follows assembles the comparative evidence into practical guidance.

3.5 Conclusions

Using a single Local Level (LL) specification on the same dataset, the frequentist plug–in and the fully Bayesian approach converge on the same empirical reading: Italian real GDP growth is governed by a persistent low–frequency component punctuated by rare, large shocks. Both inferential routes extract a latent path that adjusts cautiously to new information, consistent with an environment in which measurement noise and short–run volatility are sizable relative to genuine shifts in trend growth. When forecast performance is measured on fixed holdouts and rolling origins, point accuracy is essentially identical: RMSE is close to 5 and MAE around 3.5 for both approaches, indicating that, once the model is fixed, real–time tracking and point forecasts are equivalently reliable under either lens.

The contrast lies in uncertainty quantification. Frequentist plug–in inference conditions on fixed variance estimates and delivers sharp, computationally light confidence bands; Bayesian inference integrates parameter risk and returns posterior predictive distributions, which yield modestly wider bands for LL and substantially wider ones for the Local Linear Trend (LLT). In turbulent windows the added dispersion improves calibration and log scores by allocating more mass to extreme outcomes, while in ordinary periods the greater sharpness of the frequentist bands tends to favour CRPS. In-sample selection and diagnostics reinforce this reading: information criteria prefer LL to LLT, and standardized one–step errors are white but heavy–tailed, indicating that coverage shortfalls around crises stem from tail behaviour rather than from systematic serial misspecification. Backcasting of the pre–sample decade behaves coherently under both stances, with uncertainty widening smoothly away from the observed span; the Bayesian version makes explicit the contribution of parameter risk, while the frequentist version remains faster to compute.

Table 3.10: Frequentist vs Bayesian: out–of–sample summaries

	Holdout (8 steps)			Rolling origin ($h = 4$)		
Model	RMSE	MAE	Cov90	RMSE	MAE	Cov90
Frequentist LL	4.821	3.357	0.625	4.972	3.622	0.656
Bayesian LL	4.799	3.272	0.625	5.098	3.738	0.656

Read across the table, both stances confirm indistinguishable point accuracy for LL; coverage is likewise aligned at about two–thirds, with shortfalls concentrated in crisis windows where heavy tails dominate. Comparing LL to LLT clarifies the source of differences: LLT injects slope uncertainty that widens predictive bands and can raise coverage and log scores during shock episodes, but this comes at the cost of sharpness and does not reduce average errors on these data. Empirically, the implications for Italian GDP are clear. Trend growth is smooth and persistent; large shocks appear as heavy–tailed residuals rather than as lasting changes in slope; short–horizon predictability is reliable in normal times but narrows quickly when crises hit; and uncertainty expands in a disciplined, data–consistent way as the horizon lengthens or as the analysis moves away from observed spans.

Taken together, the LL specification remains the empirical anchor. The frequentist plug–in route is preferable for speed, simplicity, and routine monitoring when sharp bands and rapid updates are at a premium; the Bayesian route is preferable when decisions hinge on predictive probabilities and tail risk or when parameter uncertainty is materially relevant. LLT serves as a robustness reference when sustained momentum shifts are suspected, but not as a baseline for these data. This synthesis shows how the Italian economy absorbs shocks, how volatility concentrates in exceptional periods, and how trend growth can be read as persistent yet fragile under extraordinary stress, with the two inferential frameworks delivering complementary and mutually reinforcing conclusions.

Appendix

This appendix complements the baseline Local Level (LL) analysis by showing what changes—and what does not—when the latent component is allowed to carry a stochastic slope (Local Linear Trend, LLT), by anchoring the DLM reading to a familiar reduced–form benchmark (ARIMA$(0, 1, 1)$), and by collecting additional backtesting plots. The aim is purely explanatory. The model is *specified once* in state–space form; from there, estimation is carried out with the same Kalman machinery as in the main text, and the evaluation protocol (holdout and rolling–origin, with RMSE, MAE, coverage, average width, and proper scores) is unchanged. The appendix therefore isolates the role of the slope, documents how it redistributes persistence between level and momentum, and shows why the simpler LL remains the empirical anchor.

3.6 Local Linear Trend (LLT)

The LLT augments the state with a stochastic slope,

$$\mu_t = \mu_{t-1} + \beta_{t-1} + w_{\mu,t}, \qquad \beta_t = \beta_{t-1} + w_{\beta,t}, \qquad y_t = \mu_t + v_t,$$

with independent Gaussian innovations for $(w_{\mu,t}, w_{\beta,t}, v_t)$. This structure lets persistent movements be absorbed gradually through β_t, so the signal can realign through momentum rather than discrete jumps.

Figure 3.15 shows filtered and smoothed level paths with 90% Gaussian smoothing bands. Transitions are indeed smoother than in a pure LL, but the maximum–likelihood estimates in Table 3.11 indicate that the level variance is essentially zero and the slope variance is very small: the data prefer a highly persistent level with only modest drift. Read this way, LLT collapses toward LL for this series.

Table 3.11: LLT: MLE estimates, standard errors, and signal–to–noise ratios (SNR). Variances in squared percentage points; $\text{SNR} = W/V$.

Parameter	Estimate	Std. Error
V	6.276	1.144
W	0.000	0.002
W_β	0.0004	0.0007
$\text{SNR} = W/V$	0.0000	0.0004
$\text{SNR}_\beta = W_\beta/V$	0.0001	0.0001

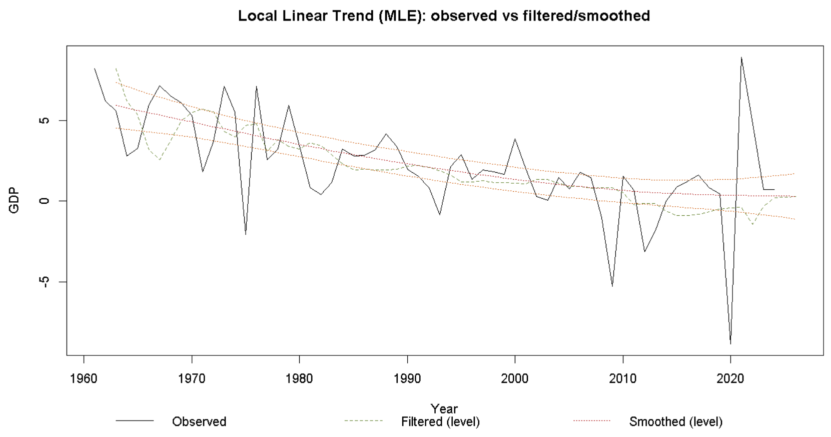


Figure 3.15: LLT (MLE).

Residual diagnostics (Figure 3.16) confirm the pattern seen in the baseline: standardized one–step errors are white (Ljung–Box(20) $p = 0.668$) yet non–Gaussian (Shapiro–Wilk $p < 10^{-4}$, Jarque–Bera $p < 10^{-15}$), consistent with rare shock episodes. Information criteria in Table 3.12 explain why the slope is not retained: the extra variance component is not rewarded in–sample, so LL is favored on both AIC and BIC.

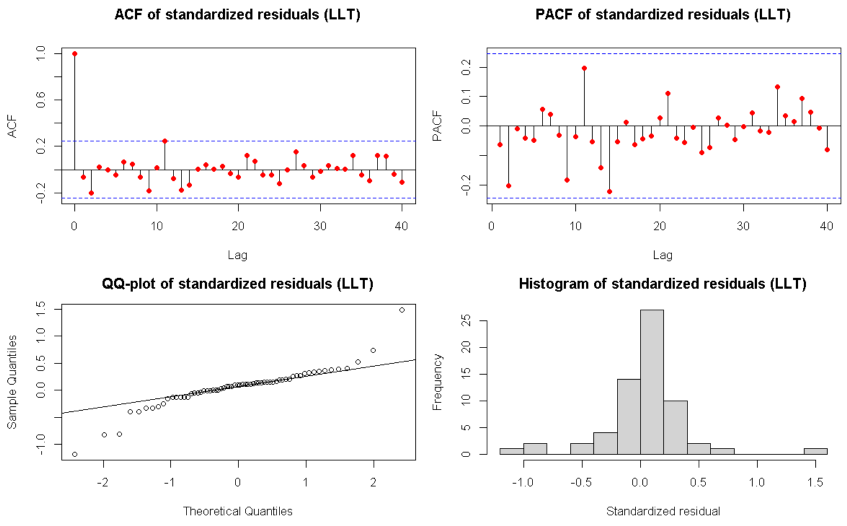


Figure 3.16: LLT: standardized one–step residuals.

Table 3.12: Model selection — Local Linear Trend (full sample).

Model	LogLik	AIC	BIC
LLT	-111.877	229.754	236.231

On forecasting, the rolling–origin fan chart (Figure 3.17) shows mean paths that adjust cautiously and bands that widen around shock–rebound windows; aggregate metrics in

Table 3.13 are modest but do not improve on LL. Horizon profiles in Figure 3.18 make the mechanics explicit: RMSE and MAE rise from $s = 1$ to $s = 2$ and then plateau; empirical coverage is near 0.75 at one step and settles around 0.62 for $s = 2$–4; average interval width increases roughly linearly with the horizon. Together with the MLE results, this suggests that letting the slope wander adds dispersion without delivering crisper calibration.

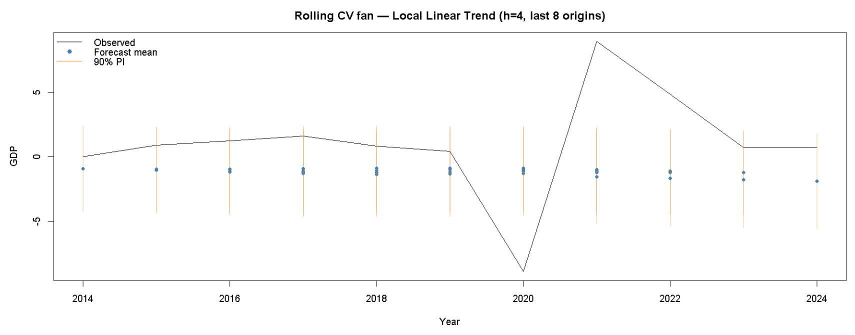


Figure 3.17: Rolling–origin fan chart ($h = 4$).

Table 3.13: LLT forecast summary (holdout and rolling CV, $h = 4$).

	Holdout (8)			Rolling CV ($h = 4$)		
Model	RMSE	MAE	Cov90	RMSE	MAE	Cov90
LLT	5.347	4.387	0.625	5.196	4.176	0.656

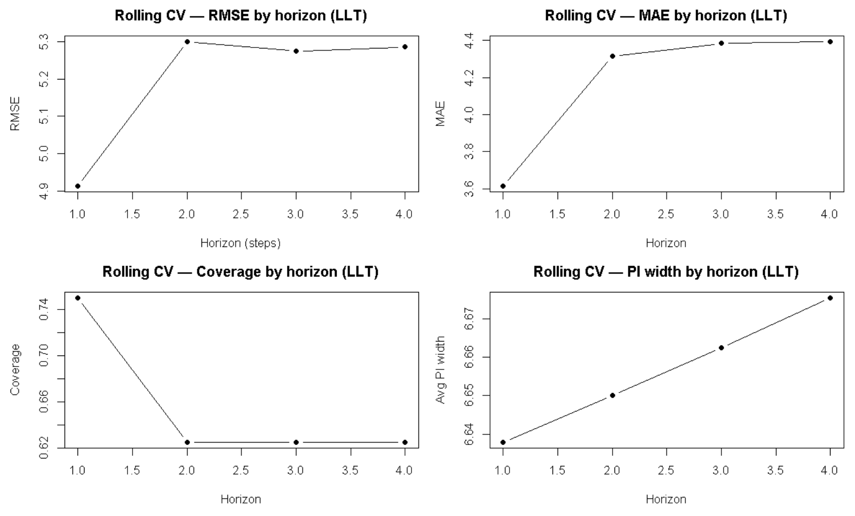


Figure 3.18: Rolling CV, LLT.

3.7 ARIMA(0,1,1): reduced–form benchmark

The ARIMA$(0, 1, 1)$ serves as an external yardstick for short–run dynamics. In the main text the estimated MA(1) coefficient is -0.84374; the one–step RMSE is 2.676; and one–step forecasts from ARIMA and the LL are very close (correlation 0.9807). The benchmark is therefore useful as a check on near–term predictions but adds no structural information beyond what the latent–level DLM already extracts.

Eight-step holdout

The last eight observations are kept for testing, with each model re-estimated on the preceding span. In this turbulent window both specifications under-cover—the nominal 90% bands include five out of eight realizations—consistent with the shock–rebound sequence. The Local Level (LL) nevertheless delivers lower RMSE and MAE than the Local Linear Trend (LLT), and its mean path re-centres more quickly after the turning point, which explains the ranking summarized in Table 3.13.

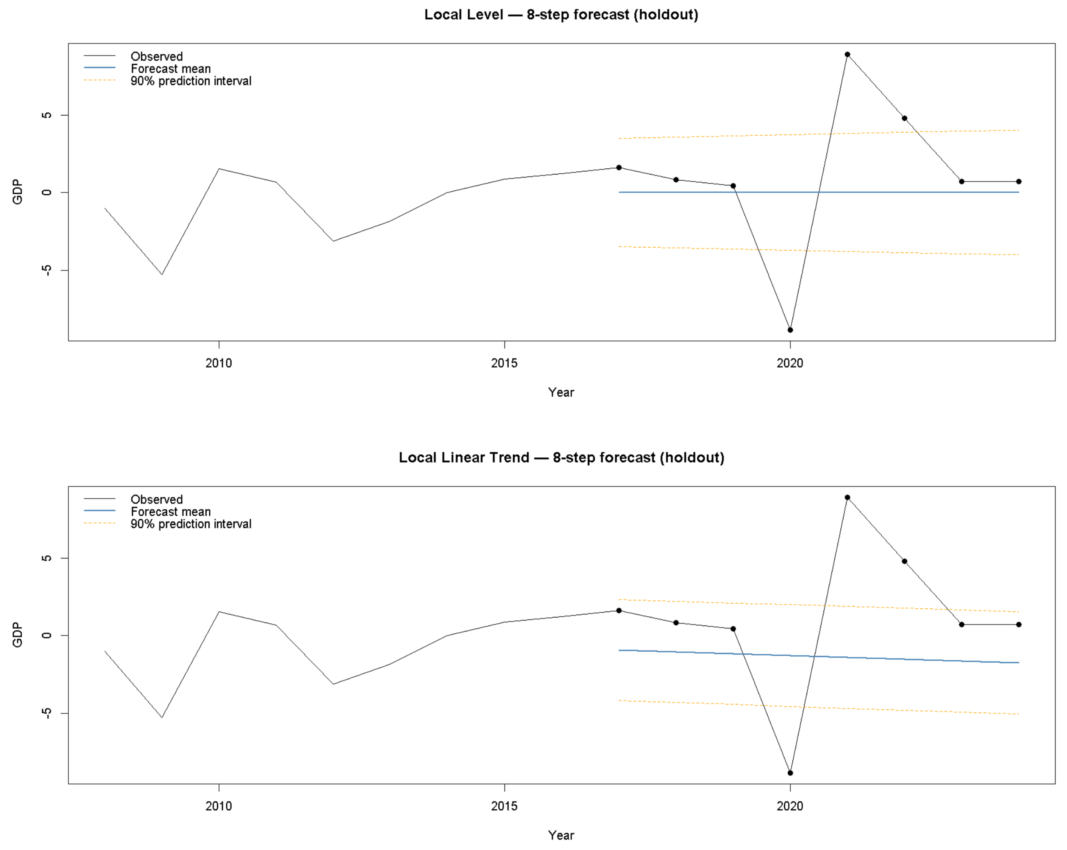


Figure 3.19: Eight-step holdout forecasts.

Rolling fan charts (last eight origins, $h = 4$)

Rolling fan charts show how forecasts behave in real time around structural episodes. Before the pandemic, both models track the mild expansion; LLT tends to place forecast means slightly below the realized path, whereas LL stays more tightly centred. The 2020 collapse falls outside Gaussian bands for both. During the rebound, LL re-centres faster while LLT lags as the stochastic slope learns; from 2022–2024, as growth converges toward zero, LL remains closer to outcomes with fewer large misses.

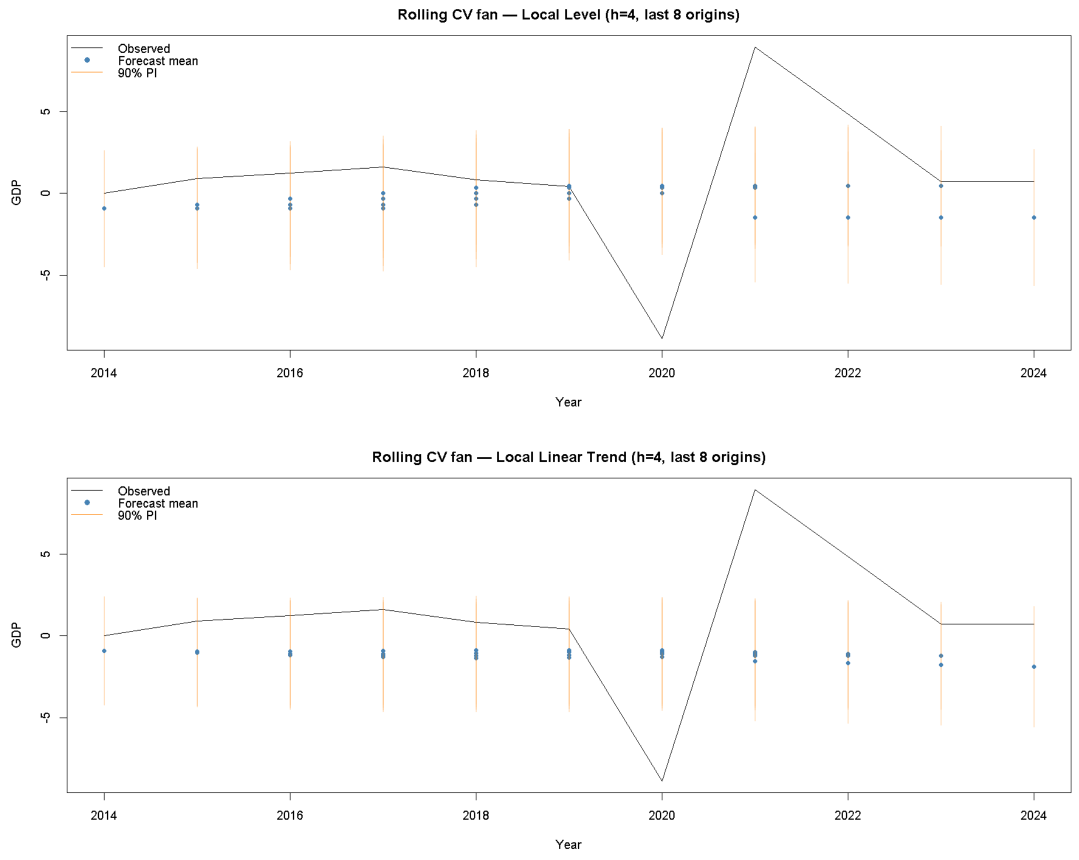
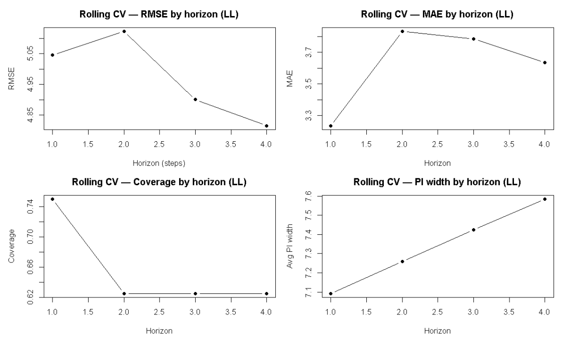


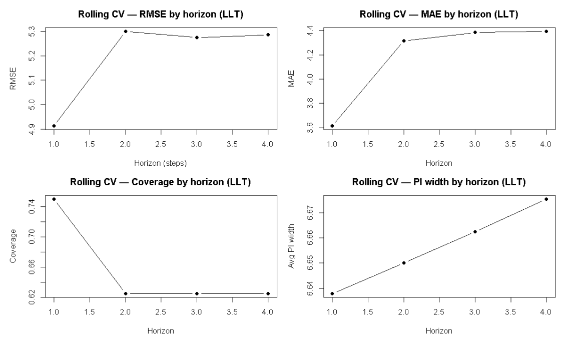
Figure 3.20: Rolling-origin fan charts ($h = 4$).

Per-horizon performance profiles

These panels track accuracy, calibration, and sharpness across horizons $s = 1, \ldots, 4$. One-step forecasts are most accurate for both models. For LL, RMSE/MAE rise at $s = 2$ and ease by $s = 4$, indicating partial recentring; for LLT they rise at $s = 2$ and remain elevated through $s = 4$, signaling inertia induced by the stochastic slope. Empirical coverage is similar across models—about 0.75 at one step and about 0.62 for $s \geq 2$, i.e., under-coverage relative to the 90% target—while average interval width increases monotonically with the horizon. LLT bands are systematically narrower than LL's, yet coverage is essentially the same; this combination points to heavy-tailed episodes and residual parameter risk rather than a calibration gap unique to one model.



(a) Local Level (LL)



(b) Local Linear Trend (LLT)

Figure 3.21: Rolling-origin cross-validation metrics by horizon ($s = 1, \dots, 4$). Top: RMSE and MAE; bottom: empirical 90% coverage and average PI width.

Aggregate rolling-h=4 comparison

For completeness, the bar chart below reports aggregate rolling metrics across the last eight origins. LL attains lower RMSE and MAE with essentially identical coverage and comparable sharpness, reinforcing the conclusion that a parsimonious stochastic level is preferable for this dataset.

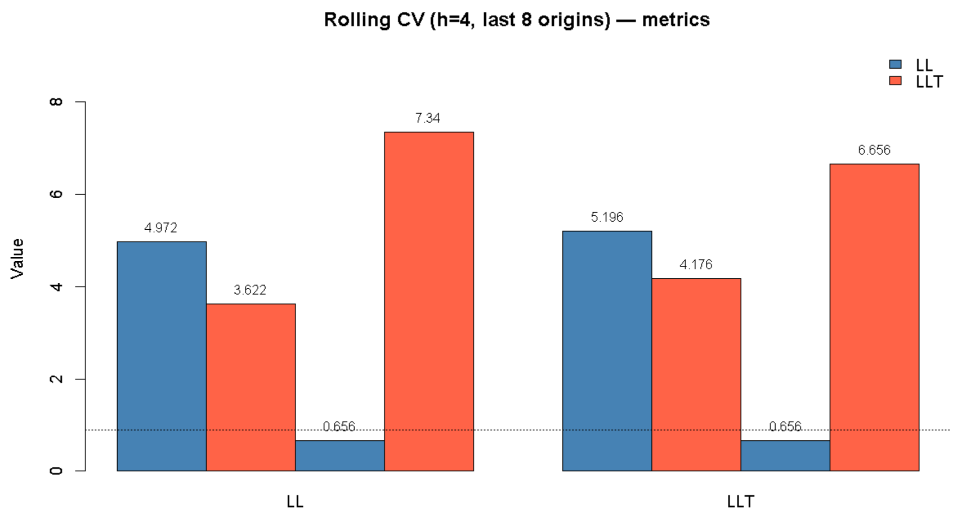


Figure 3.22: Aggregate rolling CV metrics.

Designs and metrics are exactly those used in the chapter: a fixed eight–step holdout and a rolling–origin exercise with $h = 4$, evaluated by RMSE, MAE, empirical coverage and average width, with proper scores available when simulation is used. For LLT the additional plots (Figures 3.17–3.18) show the expected widening of bands with the horizon and under–coverage in shock windows, mirroring the LL evidence and reinforcing the interpretation that Gaussian forecasts remain slightly too narrow during extremes.

Takeaway. Allowing for a stochastic slope changes the allocation of persistence but not the substantive reading: the series is dominated by a smooth low–frequency component punctuated by rare large shocks. With this dataset the extra variance component is not rewarded by the likelihood or by forecast calibration, so LL remains the reference specification; ARIMA provides a consistent one–step benchmark; and the backtesting plots document how uncertainty evolves across horizons and regimes under the common evaluation protocol.

3.8 Bayesian approach

This section complements the Bayesian Local Level (LL) analysis in the main text by documenting the *Local Linear Trend* (LLT) extension, its posterior summaries, convergence diagnostics, and predictive performance. The LLT adds a stochastic slope to the stochastic level, allowing persistence to be allocated between level and trend. Priors are weakly informative and scaled to the data (observation variance centred near $\mathrm{var}(y)$; state variances centred well below it); posterior computation proceeds via Gibbs sampling with Forward–Filtering Backward–Sampling (FFBS). Forecasts are strictly *posterior predictive*: at each origin we draw (V, W) and the terminal state, evolve the system forward, and summarize the implied predictive distribution.

Posterior inference and convergence

For the LL baseline, posterior mass concentrates on configurations in which observation noise dominates state evolution. Credible envelopes for the latent level tighten in calm periods and widen near endpoints or large shocks. Figure 3.23 shows the LLT analogue: the posterior mean level remains smooth, with wider envelopes reflecting slope uncertainty.

Table 3.14: Bayesian Local Level (LL): posterior summaries for V, W and the signal-to-noise ratio $\mathrm{SNR} = W/V$.

	mean	sd	$q_{2.5}$	q_{50}	$q_{97.5}$
V	6.089	1.205	4.091	5.935	8.812
W	0.439	0.189	0.199	0.400	0.897
W/V	0.075	0.038	0.030	0.067	0.168

Table 3.15: Bayesian Local Linear Trend (LLT): posterior summaries for V, W (level) and W_β (slope).

	mean	sd	$q_{2.5}$	q_{50}	$q_{97.5}$
V	6.111	1.302	3.940	5.971	8.971
W	0.612	0.456	0.171	0.487	1.850
W_β	0.371	0.107	0.219	0.352	0.620

Convergence diagnostics for LL (multi-chain) indicate stable mixing: Gelman–Rubin $\widehat{R}$ has point estimate and 97.5% bound equal to 1.00 for both V and W; effective sample sizes are large (about $6{,}217$ for V and $2{,}830$ for W across retained draws). Standardized one-step residuals from the posterior-mean plug-in pass Ljung–Box(20) ($p = 0.454$), while normality is rejected in the main chapter; tails remain heavier than Gaussian around shock–rebound episodes.

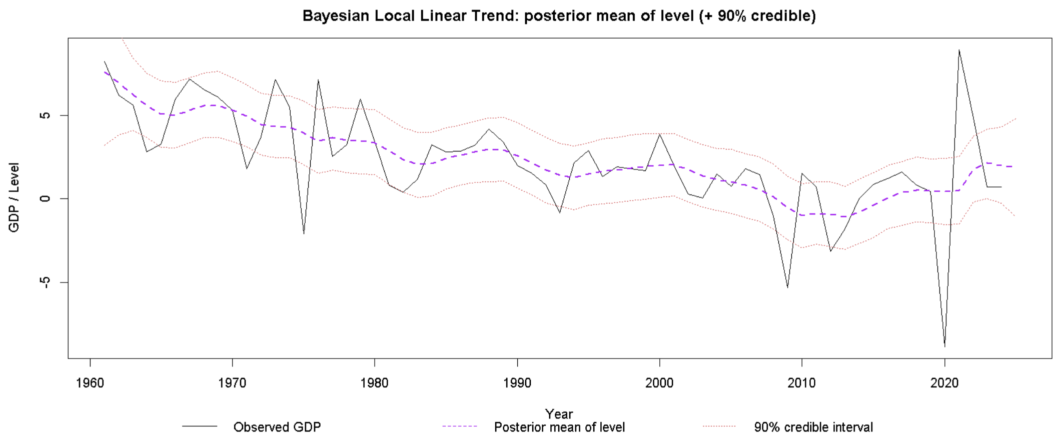


Figure 3.23: Bayesian Local Linear Trend.

Posterior predictive evaluation

Eight–step holdout (last eight years). Both models are evaluated on an $h = 8$ holdout using posterior predictive draws. LL produces sharper intervals and lower point errors; LLT produces much wider bands and consequently higher empirical coverage during extremes. Proper scores convey the same trade-off: LL attains lower (better) CRPS on average, whereas LLT benefits in log score from extra tail mass when realizations are extreme.

Table 3.16: Eight–step holdout (posterior predictive). RMSE/MAE in pp; Coverage is empirical fraction inside the nominal 90% band; AvgWidth is average band width. Log score is summed over the h steps; higher is better.

Model	RMSE	MAE	Cov90	AvgWidth	LogScore
Local Level (LL)	4.799	3.272	0.625	8.346	-28.098
Local Linear Trend (LLT)	5.304	3.837	0.875	23.172	-24.769

Table 3.17: Eight–step holdout: mean CRPS (lower is better).

	LL	LLT
Mean CRPS	2.701	3.000

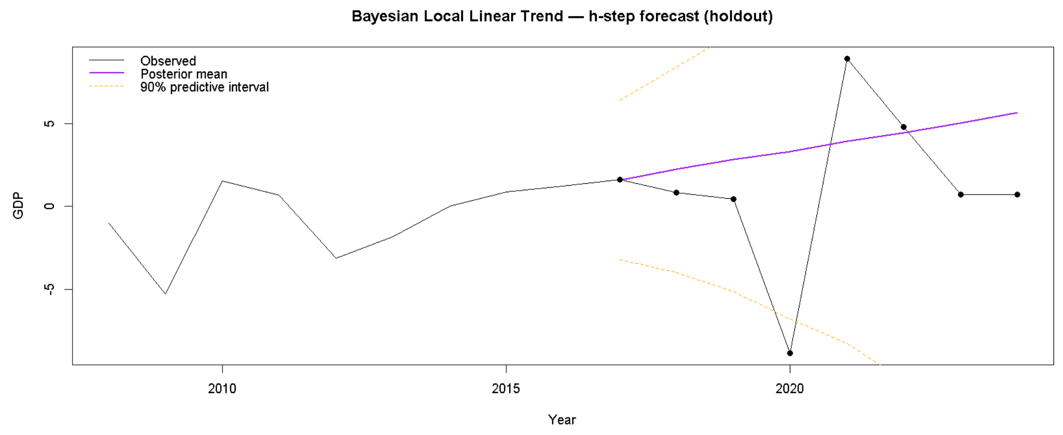


Figure 3.24: Eight–step holdout: Bayesian LLT posterior mean.

Rolling origin ($h = 4$, last eight origins). Under a rolling design, log scores again favour LLT (fatter tails help in turbulent windows), while CRPS—rewarding sharp, calibrated forecasts—favours LL. Aggregating across origins:

$$\text{LogScore sum: LL} = -125.069 \quad | \quad \text{LLT} = -120.742; \qquad \overline{\text{CRPS}}: \text{LL} = 2.980 \text{ vs LLT} = 3.939.$$

For LL, the horizon profile mirrors the frequentist pattern reported in the main chapter: RMSE/MAE peak at $s = 2$ and ease by $s = 4$ as the system re-centres; empirical coverage averages 0.75 at $s = 1$ and about 0.62 for $s \geq 2$, with average widths increasing monotonically from ≈ 7.4 to ≈ 8.3.

Backcasting and external benchmark

Both LL and LLT can be used to backcast a pre-sample decade (1950–1959) by smoothing the extended series and reporting predictive bands on the observable scale; the LL variant delivers tighter, more stable fans, while the LLT spreads adjustments through slope learning and yields broader uncertainty. As an external check, one-step forecasts from the LL and from a reduced-form ARIMA$(0, 1, 1)$ are very highly correlated (about 0.966), confirming the known link while preserving the advantages of the state–space formulation (real-time filtering/smoothing, native treatment of gaps, posterior predictive inference).

Takeaways

The Bayesian evidence aligns with the frequentist reading while making uncertainty explicit. A single smooth stochastic level captures the low-frequency component of Italian growth; residuals are white but non-Gaussian; short horizons are the most forecastable; and the same machinery extends naturally to missing spans. The LLT's slope adds flexibility but, in these data, its wider predictive bands do not translate into lower average errors or sharper distributions—hence LL remains the preferred baseline, with LLT valuable as a robustness reference in regimes where genuine momentum shifts are sustained across several years.

Chapter 4

Synthesis, Comparative Assessment, and Practical Guidance

This thesis investigates Dynamic Linear Models as linear–Gaussian state–space systems that decompose observed time series into a slowly evolving signal and transitory noise, posing a pragmatic question: when monitoring and forecasting with this shared specification, which inferential stance is more appropriate, frequentist or Bayesian. The comparison is framed deliberately around the stance rather than the structure: a single model backbone is held fixed—Local Level as the baseline and Local Linear Trend as a disciplined extension, with ARIMA(0,1,1) as a compact reduced-form benchmark—so that differences can be attributed to how uncertainty is handled and communicated, not to changes in specification. Evidence comes from simulation designs that stress identification, calibration, and tail behavior, and from an empirical test bed on Italian real GDP growth over 1960–2024, a horizon long enough to traverse tranquil expansions, recessions, and rare shocks; this setting induces shifts in the effective signal-to-noise ratio and episodes of weak identification for variance components, revealing how each stance behaves when information is thin or volatility shifts. Within the shared Local Level specification the two routes deliver essentially the same point accuracy at short horizons; on an eight–step holdout RMSE is about 4.8 and MAE about 3.3 for both, with rolling results at $h = 4$ similarly aligned. Differences emerge in uncertainty quantification: frequentist plug-in bands are sharper and occasionally optimistic around extremes, whereas Bayesian posterior predictive bands are modestly wider and maintain nominal coverage when identification weakens, without altering central forecasts. Allowing a stochastic slope in the Local Linear Trend increases posterior uncertainty and produces substantially wider bands and higher nominal coverage, but without a compensating improvement in average accuracy relative to the simpler level model; information diagnostics and residual checks therefore point back to the Local Level as the operative description for these data. Standardized one-step errors are serially clean yet display non-Gaussian features in shock–rebound years, suggesting heavy tails or time-varying volatility as practical extensions when tail calibration is a priority.

The objective is to compare frequentist and Bayesian inference under a fixed specification and to translate the evidence into guidance that is task-specific, sector-specific, and horizon-specific. Once the model is held constant, short-horizon point forecasts are essentially indistinguishable because both stances inherit the same filtering backbone; the decisive margin becomes uncertainty. Conditioning on estimated variance components yields sharp, easily deployable intervals and very low latency, which is attractive wherever throughput and automation dominate, though rigidity can surface when identification is weak, when data arrive with gaps, or when shocks induce departures from Gaussianity. Integrating over variance uncertainty and propagating it to states and predictions delivers intervals that are wider but more faithfully calibrated, with gains that become more pronounced as horizons lengthen or as the signal-to-noise ratio fluctuates across regimes; the computational premium is the price of this coherence, with the payoff visible in probability statements that remain honest when information is uneven. Both stances handle internal missing observations natively within the state–space recursion; what differs is how much parameter risk is carried forward—limited under plug-in, explicit under posterior propagation—so that fan charts inflate appropriately in the latter when identification is fragile.

These contrasts map cleanly across time scales, data conditions, and sectors. At very short and short horizons (one to four steps) in stable environments, where decisions are constrained by latency and series are numerous, a frequentist baseline is preferable: it achieves the same central forecasts at minimal operational cost and is transparent for scalable benchmarking. At medium horizons (five to twelve steps) or whenever information thins—because of publication gaps, ragged edges, small effective samples, or low signal-to-noise (here around 0.03)—posterior propagation better preserves nominal coverage and keeps predictive bands interpretable, while plug-in remains serviceable if intervals are post-adjusted, for example by conservative variance discounting or resampling-based inflation. At long and very long horizons model uncertainty dominates and either Bayesian predictive densities or explicit model expansion with heavy tails, stochastic volatility, or regime shifts become advisable regardless of stance. Domain guidance follows the same logic. High-frequency trading and other ultra-low-latency settings favor a frequentist Local Level (optionally robustified) as the default, with Bayesian density forecasts reserved for tail-risk modules that can be computed off the latency-critical path. For financial series at daily or weekly frequency, both routes benefit from heavy-tailed or volatility-modulated innovations; Bayesian densities justify their cost when decisions hinge on exceedance probabilities or value-at-risk style statements. For macroeconomic indicators used in monitoring and nowcasting, a smooth stochastic level with frequentist plug-in is compelling for production dashboards, while selected series can be promoted to Bayesian density forecasting when publication gaps, ragged edges, or explicit coverage commitments are central. In operations, energy, and demand planning, batch forecasting across many series typically favors the frequentist baseline for throughput, with escalation to Bayesian inference whenever service-level guarantees, asymmetric penalties, or governance requirements call for calibrated probabilities.

The overall contribution is a like-for-like assessment that converts an abstract debate into operational guidance: once the specification is fixed, the stances converge on point efficiency and diverge in how they price and convey uncertainty. Limits are explicit. The analysis focuses on linear–Gaussian DLMs; stronger non-Gaussian features, regime shifts, or pronounced volatility clustering require richer error structures. Prior beliefs were deliberately weak, which is appropriate for transparency but may matter in very small samples. The empirical base centers on a single long macroeconomic series; although it spans diverse regimes, external validity to other data environments should be established case by case. Future work follows naturally by extending error blocks to heavy tails and stochastic volatility within the same backbone, enriching the signal with dynamic regressors and hierarchical pooling to share strength across related series, comparing non-Gaussian filters and modern simulation engines to map the latency–calibration frontier in demanding settings, and widening the empirical base across frequencies and sectors.

In conclusion, under a shared DLM backbone the choice between frequentist and Bayesian routes is contextual rather than absolute. At short horizons in stable conditions, speed and simplicity argue for the frequentist stance; at medium and longer horizons, in the presence of missing data, fragile identification, or decisions that hinge on calibrated probabilities, the Bayesian stance is preferable. Both benefit from robust error structures when shocks dominate, and both can be embedded in pipelines that escalate from a fast baseline to full density forecasting where the decision problem demands it. All code used to generate simulations, estimates, diagnostics, and figures is provided in the Appendix for replication. The findings confirm patterns established in the literature and offer disciplined guidance rather than methodological novelty: baseline frequentist for real-time automation; promote to Bayesian whenever probability statements, coverage commitments, or tail control drive the application.

Appendix A

R Code Listings

This appendix collects the R code used in Chapter 1, Chapter 2, Chapter 3. Each listing is self-contained and reproducible.

.1 Dynamic Linear Models

A.1 — Random Walk Plus Noise (RWPN) (Section 1.2.2: Local Level Model)

Outputs: Figure 1.2.

Listing 1: RWPN definition and simulation

```r
library(dlm)
set.seed(123)

# Parameters
n <- 100; W <- 1.0; V <- 0.5

# Model (dlm uses FF/GG instead of F/G)
model_rwpn <- dlm(m0 = 0, C0 = 10, FF = 1, V = V, GG = 1, W = W)

# Manual simulation to plot true state
mu <- cumsum(rnorm(n, sd = sqrt(W)))
y  <- mu + rnorm(n, sd = sqrt(V))

plot(1:n, y, type="l", col="gray30", lwd=1.2,
ylim=range(c(y, mu)),
main="RWPN: Observations and Latent State",
xlab="Time", ylab="Value")
lines(1:n, mu, col="forestgreen", lwd=2)
legend("topleft", bty="n",
legend=c("Observations","Latent state"),
col=c("gray30","forestgreen"), lty=1, lwd=2)
```

A.2 — RWPN: variations of W and V (Section 1.2.2: Local Level Model)

Outputs: Figure 1.3.

Listing 2: RWPN with different (W, V) combinations

```r
    library(dlm); set.seed(123); n <- 100

    par(mfrow=c(2,2), mar=c(3.2,3.2,2.2,1))
    # (a) High V
    W <- 1.0; V <- 2.0
    mu <- cumsum(rnorm(n, sd=sqrt(W))); y <- mu + rnorm(n, sd=sqrt(V))
    plot(y, type="l", col="grey40", main="W=1.0, V=2.0"); lines(mu, col="blue
")

    # (b) Small W
    W <- 0.2; V <- 0.5
    mu <- cumsum(rnorm(n, sd=sqrt(W))); y <- mu + rnorm(n, sd=sqrt(V))
    plot(y, type="l", col="grey40", main="W=0.2, V=0.5"); lines(mu, col="
magenta")

    # (c) W=0 (constant level)
    W <- 0; V <- 1.0
    mu <- rep(0, n); y <- rnorm(n, sd=sqrt(V))
    plot(y, type="l", col="darkgreen", main="W=0, V=1.0"); lines(mu, col="
chocolate")

    # (d) V=0 (observations = state)
    W <- 1.0; V <- 0
    mu <- cumsum(rnorm(n, sd=sqrt(W))); y <- mu
    plot(y, type="l", col="navy", main="W=1.0, V=0"); lines(mu, col="
deeppink3", lty=5)
```

A.3 — Kalman filter: low vs high SNR (Section 1.3.2: Bayesian Updating and the Kalman Filter in DLMs)

Outputs: Figure 1.7.

Listing 3: Kalman filter: SNR comparison

```r
library(dlm); set.seed(123)
n <- 100
mu_true <- cumsum(rnorm(n, sd=sqrt(1)))
y       <- mu_true + rnorm(n, sd=sqrt(0.5))

# Models
model_low  <- dlm(m0=0,C0=10,FF=1,V=1.0,GG=1,W=0.1)
model_high <- dlm(m0=0,C0=10,FF=1,V=0.1,GG=1,W=1.0)

filt_low  <- dlmFilter(y, model_low)
filt_high <- dlmFilter(y, model_high)

par(mfrow=c(1,2), mar=c(3.2,3.2,2.2,1))
plot(y, type="l", col="gray50", main="Low SNR")
lines(mu_true, col="forestgreen", lwd=2)
lines(drop(filt_low$m), col="darkslateblue", lty=2, lwd=2)

plot(y, type="l", col="gray50", main="High SNR")
lines(mu_true, col="forestgreen", lwd=2)
lines(drop(filt_high$m), col="firebrick2", lty=3, lwd=2)
```

A.4 — RTS smoothing and 95% bands (Section 1.3.3: Smoothing)

Outputs: Figure 1.8. **Reference:** Fig. 1.8

Listing 4: RTS smoother and 95% probability bands

```r
library(dlm); set.seed(123)
n <- 100
mu_true <- cumsum(rnorm(n, sd=sqrt(1)))
y       <- mu_true + rnorm(n, sd=sqrt(0.5))

model <- dlm(m0=0,C0=10,FF=1,V=0.5,GG=1,W=1.0)
filt   <- dlmFilter(y, model)
sm     <- dlmSmooth(filt)

se  <- sqrt(unlist(dlmSvd2var(sm$U.S, sm$D.S)))
up  <- drop(sm$s) + qnorm(0.975) * se
lo  <- drop(sm$s) - qnorm(0.975) * se

plot(y, type="o", col="magenta", pch=16, cex=0.6,
main="Smoothed state (95%)", xlab="Time", ylab="Value")
lines(mu_true, col="forestgreen", lwd=2)
lines(drop(sm$s), col="black", lwd=2)
lines(up, col="grey60", lty=2); lines(lo, col="grey60", lty=2)
```

A.5 — Smoothing: SNR comparison (Section 1.3.3: Smoothing)

Outputs: Figure 1.9.

Listing 5: Smoothing under low vs high SNR

```r
library(dlm); set.seed(123)
n <- 100
mu_true <- cumsum(rnorm(n, sd=sqrt(1)))
y       <- mu_true + rnorm(n, sd=sqrt(0.5))

# Low SNR
mod_low <- dlm(m0=0,C0=10,FF=1,V=1.0,GG=1,W=0.1)
filt_low <- dlmFilter(y, mod_low)
sm_low   <- dlmSmooth(filt_low)
se_low   <- sqrt(unlist(dlmSvd2var(sm_low$U.S, sm_low$D.S)))
up_low   <- drop(sm_low$s) + qnorm(0.975)*se_low
lo_low   <- drop(sm_low$s) - qnorm(0.975)*se_low

# High SNR
mod_high <- dlm(m0=0,C0=10,FF=1,V=0.1,GG=1,W=1.0)
filt_high <- dlmFilter(y, mod_high)
sm_high   <- dlmSmooth(filt_high)
se_high   <- sqrt(unlist(dlmSvd2var(sm_high$U.S, sm_high$D.S)))
up_high   <- drop(sm_high$s) + qnorm(0.975)*se_high
lo_high   <- drop(sm_high$s) - qnorm(0.975)*se_high

par(mfrow=c(1,2), mar=c(3.2,3.2,2.2,1))
plot(y, type="l", col="gray50",
main="Low SNR: filter vs smoother")
lines(mu_true, col="forestgreen", lwd=2)
lines(drop(filt_low$m), col="darkslateblue", lwd=2, lty=2)
lines(drop(sm_low$s), col="black", lwd=2)
lines(up_low, col="black", lty=3); lines(lo_low, col="black", lty=3)

plot(y, type="l", col="gray50",
main="High SNR: filter vs smoother")
lines(mu_true, col="forestgreen", lwd=2)
lines(drop(filt_high$m), col="firebrick2", lwd=2, lty=3)
lines(drop(sm_high$s), col="black", lwd=2)
lines(up_high, col="black", lty=3); lines(lo_high, col="black", lty=3)
```

A.6 — k-step-ahead forecasts (low/high SNR) (Section 1.3.5: State Prediction)

Outputs: Figure 1.11.

Listing 6: Forecasting and predictive intervals

```r
library(dlm); set.seed(123)
n <- 100
mu_true <- cumsum(rnorm(n, sd=sqrt(1)))
y       <- mu_true + rnorm(n, sd=sqrt(0.5))

# ---- LOW SNR ----
mod_low <- dlm(m0=0,C0=10,FF=1,V=1.0,GG=1,W=0.1)
filt_low <- dlmFilter(y, mod_low)
sm_low   <- dlmSmooth(filt_low)  # for a smoother historical curve

fc_low <- dlmForecast(filt_low, nAhead=10, sampleNew=20)
f_mean_low <- drop(fc_low$f)
f_sd_low   <- sqrt(sapply(fc_low$Q, drop))
f_up_low   <- f_mean_low + qnorm(0.975)*f_sd_low
f_lo_low   <- f_mean_low - qnorm(0.975)*f_sd_low
idx_low    <- (n+1):(n+10)

par(mfrow=c(1,2), mar=c(3.2,3.2,2.2,1))
plot(y, type="l", col="gray50", lwd=1.2,
main="Forecast: Low SNR", xlab="Time", ylab="Value",
xlim=c(1, n+10))
lines(drop(sm_low$s), col="black", lwd=2)
lines(idx_low, f_mean_low, col="red", lwd=2)
lines(idx_low, f_up_low,  col="red", lty=2)
lines(idx_low, f_lo_low,  col="red", lty=2)
abline(v=n, lty=3, col="gray40")
invisible(lapply(fc_low$newObs, function(path) {
        lines(idx_low, path, col=adjustcolor("gray20", 0.4))
}))

# ---- HIGH SNR ----
mod_high <- dlm(m0=0,C0=10,FF=1,V=0.1,GG=1,W=1.0)
filt_high <- dlmFilter(y, mod_high)
sm_high   <- dlmSmooth(filt_high)

fc_high <- dlmForecast(filt_high, nAhead=10, sampleNew=20)
f_mean_high <- drop(fc_high$f)
f_sd_high   <- sqrt(sapply(fc_high$Q, drop))
f_up_high   <- f_mean_high + qnorm(0.975)*f_sd_high
```

```r
        f_lo_high   <- f_mean_high - qnorm(0.975)*f_sd_high
        idx_high    <- (n+1):(n+10)

        plot(y, type="l", col="gray50", lwd=1.2,
        main="Forecast: High SNR", xlab="Time", ylab="Value",
        xlim=c(1, n+10))
        lines(drop(sm_high$s), col="black", lwd=2)
        lines(idx_high, f_mean_high, col="red", lwd=2)
        lines(idx_high, f_up_high,  col="red", lty=2)
        lines(idx_high, f_lo_high,  col="red", lty=2)
        abline(v=n, lty=3, col="gray40")
        invisible(lapply(fc_high$newObs, function(path) {
                lines(idx_high, path, col=adjustcolor("gray20", 0.4))
        }))
```

A.7 — Local Linear Trend: flexible vs rigid (Section 1.2.2: Local Linear Trend Model)

Outputs: Figure 1.4, Table 1.2.

Listing 7: Local linear trend models: comparison

```r
        library(dlm); set.seed(123)
        n <- 100
        mu <- 0 + 0.5 * (1:n)
        y  <- mu + rnorm(n, sd = 5)

        # Model 1: highly adaptive
        mod_flex <- dlmModPoly(order=2, dV=10, dW=c(1000, 300))
        fit_flex <- dlmFilter(y, mod_flex)

        # Model 2: rigid
        mod_rigid <- dlmModPoly(order=2, dV=10, dW=c(0.1, 0.1))
        fit_rigid <- dlmFilter(y, mod_rigid)

        MAD  <- function(e) mean(abs(e))
        MSE  <- function(e) mean(e^2)
        MAPE <- function(e, y) mean(abs(e)/abs(y))

        mad_flex  <- MAD(fit_flex$f - y)
        mse_flex  <- MSE(fit_flex$f - y)
        mape_flex <- MAPE(fit_flex$f - y, y)

        mad_rig  <- MAD(fit_rigid$f - y)
        mse_rig  <- MSE(fit_rigid$f - y)
```

```r
    mape_rig <- MAPE(fit_rigid$f - y, y)

    par(mfrow=c(2,2), mar=c(3.2,3.2,2.2,1))
    plot(y, type="l", col="gray40", lwd=1.5,
    main="Flexible (large dW)");
    lines(dropFirst(fit_flex$m[,1]), col="blue", lwd=2)
    plot(y - fit_flex$f, type="h", main="Residuals: Flexible"); abline(h=0,
col="red")

    plot(y, type="l", col="gray40", lwd=1.5,
    main="Rigid (small dW)");
    lines(dropFirst(fit_rigid$m[,1]), col="darkgreen", lwd=2)
    plot(y - fit_rigid$f, type="h", main="Residuals: Rigid"); abline(h=0,col=
"red")

    cat("\n--- Flexible ---\n",
    "MAD =", round(mad_flex,2),
    " MSE =", round(mse_flex,2),
    " MAPE =", round(mape_flex,4), "\n")
    cat("\n--- Rigid ---\n",
    "MAD =", round(mad_rig,2),
    " MSE =", round(mse_rig,2),
    " MAPE =", round(mape_rig,4), "\n")
```

A.8 — Normal DLM with dynamic covariate (Section 1.2.2: Dynamic Regression with Covariates)

Outputs: Figure 1.5.

Listing 8: Normal DLM with regressor via JFF/X

```r
    set.seed(123); library(dlm)
    n <- 100
    x <- rnorm(n)

    model_dlr <- dlm(
    m0 = rep(0,2),
    C0 = 10 * diag(2),
    FF = matrix(c(1,0), nrow=1),
    V  = 0.5,
    GG = diag(2),
    W  = diag(c(0.4, 0.2)),
    JFF = matrix(c(0,1), nrow=1),
    X   = as.matrix(x)
    )
```

```

sim <- dlmSim(model_dlr, n)
y   <- sim[,1]    # observations
mu  <- sim[,2]    # latent level

plot(y, type="l", col="gray30",
main="Normal DLM: y and latent level")
lines(mu, col="dodgerblue3")
```

A.9 — Seasonality with dummies (monthly, $S=12$) (Section 1.2.3: Seasonality)

Outputs: Figure 1.6.

Listing 9: Seasonality via dummy regressors

```
set.seed(42)

n_years <- 5; S <- 12; n <- n_years * S

# True effects summing to zero
gamma <- c(2, -0.5, rep(0, S-2))
gamma[3:S] <- -sum(gamma[1:2])/(S-2)

# Seasonal dummies
D <- diag(S)[rep(1:S, n_years), ]

# Series with seasonality / without
y  <- as.numeric(D %*% gamma + rnorm(n, sd=1))
y0 <- rnorm(n, sd=1)

# OLS estimates with dummies
b_hat  <- coef(lm(y  ~ D))[-1]
b0_hat <- coef(lm(y0 ~ D))[-1]

dates <- seq(as.Date("2010-01-01"), by="month", length.out=n)

par(mfrow=c(2,2), mar=c(3.2,3.2,2.2,1))
plot(dates, y,  type="o", main="With seasonality",     xlab="", ylab="
Value")
abline(v=dates[seq(1,n,by=S)], col="lightgray", lty=3)

plot(1:S, b_hat, type="b", xaxt="n",
xlab="Month", ylab="Estimated effect",
main="Estimated seasonal effects (with seasonality)")
```

```
    axis(1, at=1:S, labels=month.abb); abline(h=0, col="gray")

    plot(dates, y0, type="o", main="Without seasonality", xlab="", ylab="
Value")
    abline(h=0, col="gray", lty=2)
    abline(v=dates[seq(1,n,by=S)], col="lightgray", lty=3)

    plot(1:S, b0_hat, type="b", xaxt="n",
    xlab="Month", ylab="Estimated effect",
    main="Estimated seasonal effects (no seasonality)")
    axis(1, at=1:S, labels=month.abb); abline(h=0, col="gray")
```

.2 Computational Techniques

Frequentist estimation

Data generation, Kalman filter, and log-likelihood (Section 2.1.3)

Outputs: Table 2.4.

Listing 10: Local-level model simulation, Kalman filter and log-likelihood.

```
# --- Local-level model: simulate, filter, and compute log-likelihood ---
set.seed(123)
T <- 100
V <- 0.8
W <- 0.3
true_V <- V; true_W <- W

# Simulate state and observations
theta <- numeric(T)
y     <- numeric(T)

theta[1] <- rnorm(1, 0, sqrt(W))
y[1]     <- theta[1] + rnorm(1, 0, sqrt(V))
for (t in 2:T) {
        theta[t] <- theta[t-1] + rnorm(1, 0, sqrt(W))
        y[t]     <- theta[t] + rnorm(1, 0, sqrt(V))
}

# Kalman filter (hand-coded for clarity)
m  <- numeric(T)  # filtered state mean m_t|t
C  <- numeric(T)  # filtered state var  C_t|t
ah <- numeric(T)  # one-step mean of y_t (a_t)
```

```r
    Q  <- numeric(T)  # one-step var  of y_t (Q_t)

    m[1] <- 0; C[1] <- 1
    for (t in 1:T) {
        if (t > 1) { ah[t] <- m[t-1]; Q[t] <- C[t-1] + V } else { ah[t] <- m[1]; Q[t] <- C[1] + V }
        e  <- y[t] - ah[t]       # forecast error
        K  <- C[t] / Q[t]        # Kalman gain
        m[t]   <- ah[t] + K * e  # update mean
        C[t+1] <- (1 - K) * C[t] + W  # update variance
    }

    # Gaussian log-likelihood from one-step errors
    a <- ah
    logL <- -0.5 * sum(log(2*pi*Q) + (y - a)^2 / Q)
    print(logL)
```

Maximum likelihood and log-likelihood profile (Section 2.1.3)

Outputs: Figure 2.5.

Listing 11: MLE with dlm and 2D profile of the log-likelihood.

```r
    library(dlm)

    # Builder for local-level model (log-parameters)
    buildLL <- function(par) dlmModPoly(order = 1, dV = exp(par[1]), dW = exp(par[2]))

    # MLE
    # assumes 'y', 'true_V', 'true_W' from Listing~\ref{lst:kalman-local-level}
    fit <- dlmMLE(y, parm = log(c(1, 1)), build = buildLL)
    estimated_V <- exp(fit$par[1])
    estimated_W <- exp(fit$par[2])
    cat("True:", true_V, true_W, "\nMLE:", estimated_V, estimated_W, "\n")

    # Log-likelihood profile grid
    V_vals <- seq(0.1, 1.5, length.out = 50)
    W_vals <- seq(0.05, 1.0, length.out = 50)
    loglik_matrix <- matrix(NA_real_, length(V_vals), length(W_vals))
    for (i in seq_along(V_vals)) for (j in seq_along(W_vals)) {
        val <- try(dlmLL(y, buildLL(log(c(V_vals[i], W_vals[j])))), silent = TRUE)
        if (!inherits(val, "try-error")) loglik_matrix[i, j] <- -val
    }

    # (Contour plotting done when exporting figures)
    filled.contour(x = V_vals, y = W_vals, z = loglik_matrix,
    color.palette = terrain.colors,
    xlab = "V (Observation variance)", ylab = "W (System variance)",
    main = "Log-Likelihood Profile for Local Level Model",
    plot.axes = { axis(1); axis(2);
            points(true_V, true_W, col = "red", pch = 19)
            points(estimated_V, estimated_W, col = "blue", pch = 4)
            legend("topright", legend = c("True values", "MLE estimates"),
            col = c("red", "blue"), pch = c(19, 4), bty = "n")
    })
```

ACF/PACF panels (Section 2.1.2)

Outputs: Figure 2.2.

Listing 12: ACF/PACF panels for RWPN, trend+noise, and seasonal series.

```
    library(forecast); library(ggplot2); library(gridExtra)
    set.seed(123); n <- 100

    y_rwpn   <- cumsum(rnorm(n, sd = 1)) + rnorm(n, sd = sqrt(0.5))
    y_trend  <- 0.2 * (1:n) + rnorm(n, sd = 1.0)
    y_season <- rep(c(3, -2, 1, -1), length.out = n) + rnorm(n, sd = 0.5)

    make_acf_plot <- function(ts, type = c("acf","pacf"), title = ""){
            type <- match.arg(type)
            obj <- if (type == "acf") acf(ts, plot = FALSE) else pacf(ts,
plot = FALSE)
            df  <- with(obj, data.frame(lag = lag, acf = acf))
            ggplot(df, aes(lag, acf)) +
            geom_segment(aes(xend = lag, yend = 0)) +
            geom_point(color = "red", size = 1.5) +
            geom_hline(yintercept = c(0, 1.96/sqrt(n), -1.96/sqrt(n)), lty =
"dashed") +
            labs(title = title, x = "Lag", y = NULL) + theme_bw(base_size =
12)
    }

    p1 <- make_acf_plot(y_rwpn,   "acf",  "ACF of RWPN")
    p2 <- make_acf_plot(y_rwpn,   "pacf", "PACF of RWPN")
    p3 <- make_acf_plot(y_trend,  "acf",  "ACF of Trend + Noise")
    p4 <- make_acf_plot(y_trend,  "pacf", "PACF of Trend + Noise")
```

```r
        p5 <- make_acf_plot(y_season, "acf",  "ACF of Seasonal Pattern")
        p6 <- make_acf_plot(y_season, "pacf", "PACF of Seasonal Pattern")

        grid.arrange(p1,p2,p3,p4,p5,p6, ncol = 2)
```

Stationarity tests (Section 2.1.2)

Outputs: Table 2.1.

Listing 13: ADF, Phillips–Perron and KPSS on synthetic series.

```r
        library(tseries); library(urca); library(forecast)
        set.seed(123); n <- 100

        rw_state   <- cumsum(rnorm(n, 0, sqrt(0.3))))
        y_rwpn     <- rw_state + rnorm(n, 0, sqrt(0.8))
        y_trend    <- 0.5 * (1:n) + rnorm(n, 0, 1)
        months     <- rep(1:12, length.out = n)
        Gamma      <- c(2, -0.5, rep(0, 10)); Gamma[3:12] <- -sum(Gamma[1:2]) /
    10
        y_seasonal <- Gamma[months] + rnorm(n, 0, 1)

        run_tests <- function(x){
                adf  <- adf.test(x)
                pp   <- pp.test(x)
                kpss <- kpss.test(x, null = "Level")
                list(ADF  = c(stat = adf$statistic,  p = adf$p.value),
                PP   = c(stat = pp$statistic,   p = pp$p.value),
                KPSS = c(stat = kpss$statistic, p = kpss$p.value))
        }

        results <- list(RWPN = run_tests(y_rwpn),
        Trend = run_tests(y_trend),
        Seasonal = run_tests(y_seasonal))
        print(results)
```

Residual diagnostics and normality (Section 2.1.2)

Outputs: Figure 2.3, Table 2.2, Figure 2.4, Table 2.3.

Listing 14: Residual diagnostics for the fitted local-level DLM.

```r
        library(dlm); library(forecast); library(ggplot2); library(gridExtra)
        set.seed(42)
        T <- 100; V <- 0.8; W <- 0.3
```

```r

    # Simulate and fit
    th <- numeric(T); y <- numeric(T)
    th[1] <- 0; y[1] <- th[1] + rnorm(1, sd = sqrt(V))
    for (t in 2:T){
            th[t] <- th[t-1] + rnorm(1, sd = sqrt(W))
            y[t]  <- th[t] + rnorm(1, sd = sqrt(V))
    }

    buildFun <- function(par) dlmModPoly(1, dV = exp(par[1]), dW = exp(par[2]))
    fit <- dlmMLE(y, parm = log(c(1,1)), build = buildFun)
    mod <- buildFun(fit$par)
    filt <- dlmFilter(y, mod)
    res  <- residuals(filt, sd = FALSE)

    # Plots (export when producing figures)
    p1 <- autoplot(ts(y)) + ggtitle("Simulated Local Level Model") + theme_minimal()

    df_res <- data.frame(time = 1:T, res = as.numeric(res))
    p2 <- ggplot(df_res, aes(time, res)) +
    geom_segment(aes(xend = time, yend = 0)) +
    geom_point(color = "red", size = 1.4) +
    labs(title = "Residuals from Local Level DLM", x = "Time", y = "Residuals") +
    theme_minimal()

    acf_obj <- acf(res, plot = FALSE); n <- length(res)
    ci <- 1.96/sqrt(n)
    df_acf <- data.frame(lag = acf_obj$lag[-1], acf = acf_obj$acf[-1])
    p3 <- ggplot(df_acf, aes(lag, acf)) +
    geom_hline(yintercept = c(0, ci, -ci), lty = "dashed") +
    geom_segment(aes(xend = lag, yend = 0)) +
    geom_point(color = "red", size = 1.4) + labs(title = "ACF of Residuals") + theme_minimal()

    grid.arrange(p1, p2, p3, ncol = 1)

    # Portmanteau tests
    print(Box.test(res, lag = 10, type = "Box-Pierce"))
    print(Box.test(res, lag = 10, type = "Ljung-Box"))

    # Normality checks
    qq <- ggplot(data.frame(res), aes(sample = res)) +
    stat_qq(color = "steelblue") + stat_qq_line(lty = "dashed", color = "red"
) +
    labs(title = "Q--Q Plot of Residuals") + theme_minimal()
    print(qq)

    print(shapiro.test(res))
    library(tseries); print(jarque.bera.test(res))
```

Uncertainty quantification and classical tests (Section 2.1.4)

Outputs: Table 2.5.

Listing 15: Bootstrap CIs, Likelihood Ratio, Wald and Lagrange Multiplier tests.

```
library(dlm); library(numDeriv)
set.seed(123)
T <- 100; true_V <- 0.8; true_W <- 0.3

simulate_local_ll <- function(V, W, T){
        th <- numeric(T); y <- numeric(T)
        th[1] <- 0; y[1] <- rnorm(1, sd = sqrt(V))
        for (t in 2:T){
                th[t] <- th[t-1] + rnorm(1, sd = sqrt(W))
                y[t]  <- th[t] + rnorm(1, sd = sqrt(V))
        }
        y
}

y <- simulate_local_ll(true_V, true_W, T)
build <- function(par) dlmModPoly(1, dV = exp(par[1]), dW = exp(par[2]))
fit <- dlmMLE(y, parm = log(c(1,1)), build = build, hessian = TRUE,
method = "BFGS")
MLE <- exp(fit$par); MLE_V <- MLE[1]; MLE_W <- MLE[2]

# Parametric bootstrap CIs
B <- 500; boot_V <- boot_W <- numeric(B)
for (b in 1:B){
        yb <- simulate_local_ll(MLE_V, MLE_W, T)
        fb <- dlmMLE(yb, parm = log(c(MLE_V, MLE_W)), build = build)
        eb <- exp(fb$par)
        boot_V[b] <- eb[1]; boot_W[b] <- eb[2]
}
ci_V <- quantile(boot_V, c(.025, .975)); ci_W <- quantile(boot_W, c(.025,
 .975))

```

```
        # Likelihood-ratio tests for V and W
        lrt <- function(index, true_val){
                build_restricted <- function(par){
                        p <- log(c(1,1)); p[index] <- log(true_val)
                        dlmModPoly(1, dV = exp(p[1]), dW = exp(p[2]))
                }
                fr <- dlmMLE(y, parm = log(1), build = build_restricted)
                stat <- -2 * (fr$value - fit$value); pval <- pchisq(stat, df = 1,
     lower.tail = FALSE)
                c(stat = stat, p_value = pval)
        }
        lrt_V <- lrt(1, true_V); lrt_W <- lrt(2, true_W)

        # Wald tests on log-scale
        H <- fit$hessian; se_log <- sqrt(diag(solve(H)))
        wald_V <- ((log(MLE_V) - log(true_V))/se_log[1])^2
        wald_W <- ((log(MLE_W) - log(true_W))/se_log[2])^2
        p_Wald_V <- pchisq(wald_V, df = 1, lower.tail = FALSE)
        p_Wald_W <- pchisq(wald_W, df = 1, lower.tail = FALSE)

        # Lagrange Multiplier (score) tests at the null (illustrative)
        logLik_fun <- function(lp) -dlmLL(y, build(lp))
        score <- grad(logLik_fun, fit$par)
        info  <- hessian(logLik_fun, fit$par)
        LM_V <- score[1]^2 / info[1,1]; LM_W <- score[2]^2 / info[2,2]
        p_LM_V <- pchisq(LM_V, df = 1, lower.tail = FALSE)
        p_LM_W <- pchisq(LM_W, df = 1, lower.tail = FALSE)

        list(MLE = c(V = MLE_V, W = MLE_W),
        CI_bootstrap = list(V = ci_V, W = ci_W),
        LR = list(V = lrt_V, W = lrt_W),
        Wald = c(V = p_Wald_V, W = p_Wald_W),
        LM = c(V = p_LM_V, W = p_LM_W))
```

Model selection: AIC and BIC (Section 2.1.5)

Outputs: — (console values only).

Listing 16: AIC/BIC for the local-level model fitted by MLE.

```
        # assumes 'fit' (and 'y') from Listing~\ref{lst:uq-tests}
        logLik_val <- -fit$value
        k <- length(fit$par); n <- length(y)
        AIC_val <- -2 * logLik_val + 2 * k
        BIC_val <- -2 * logLik_val + log(n) * k
        cat("AIC:", AIC_val, " BIC:", BIC_val, "\n")
```

Bayesian estimation

Gibbs and FFBS (Section 2.2.2)

Outputs: Figure 2.7, Figure 2.8.

Listing 17: Gibbs sampling with dlmGibbsDIG and FFBS states.

```
        library(dlm)
        set.seed(123)
        T <- 100; W_true <- 0.2; V_true <- 1

        # Simulate
        th <- numeric(T); y <- numeric(T)
        th[1] <- 0
        for (t in 2:T) th[t] <- th[t-1] + rnorm(1, sd = sqrt(W_true))
        y <- th + rnorm(T, sd = sqrt(V_true))

        mod <- dlmModPoly(order = 1, dV = 1, dW = 1)
        fit_g <- dlmGibbsDIG(y = y, mod = mod,
        shape.y = 1, rate.y = 1,
        shape.theta = 1, rate.theta = 1,
        n.sample = 1000, save.states = TRUE)

        V_post_mean <- mean(fit_g$dV); W_post_mean <- mean(fit_g$dW)
        cat("Posterior mean of V:", V_post_mean, "\n")
        cat("Posterior mean of W:", W_post_mean, "\n")

        # Posterior mean state trajectory (FFBS draws stored in fit_g$theta)
        th_post_mean <- apply(fit_g$theta, 1, mean)
        plot(th, type = "l", col = "black", lwd = 2,
        ylim = range(c(th, th_post_mean)), ylab = expression(theta[t]),
        main = "True vs Posterior Mean of Latent States")
        lines(th_post_mean, col = "red", lwd = 2, lty = 2)
        legend("bottomright", c("True","Posterior Mean"), col = c("black","red"),
     lty = c(1,2), bty = "n")
```

Posterior mean and 95% credible intervals (Section 2.2.3)

Outputs: Figure 2.10, Table 2.8.

Listing 18: Posterior mean and 95% credible interval for states.

```r
    # Posterior smoothing and 95% credible interval
    simulate_local_ll <- function(V, W, T){
            th <- numeric(T); y <- numeric(T); th[1] <- 0
            y[1] <- th[1] + rnorm(1, sd = sqrt(V))
            for (t in 2:T){ th[t] <- th[t-1] + rnorm(1, sd = sqrt(W)); y[t] <- th[t] + rnorm(1, sd = sqrt(V)) }
            y
    }
    y <- simulate_local_ll(0.8, 0.3, 100)
    mod <- dlmModPoly(order = 1, dV = 0.8, dW = 0.3)
    smooth <- dlmSmooth(y, mod)
    mean_post <- drop(smooth$s[-1])
    sd_post   <- sqrt(unlist(dlmSvd2var(smooth$U.S, smooth$D.S)))
    ci_low <- mean_post - 1.96 * sd_post
    ci_up  <- mean_post + 1.96 * sd_post

    plot(y, type = "l", col = "gray", main = "Posterior Mean and 95% Credible Interval")
    lines(mean_post, col = "blue"); lines(ci_low, col = "red", lty = 2); lines(ci_up, col = "red", lty = 2)
    legend("topright", c("Data","Posterior Mean","95% CI"), col = c("gray","blue","red"), lty = c(1,1,2))
```

Posterior predictive checks and LOO/WAIC (Section 2.2.5)

Outputs: Figure 2.11.

Listing 19: Posterior predictive checks (variance statistic) and LOO/WAIC.

```r
    # assumes MLE_V, MLE_W and 'y' are available
    set.seed(123); T <- length(y); nsim <- 100

    sim_y <- replicate(nsim, {
            th <- numeric(T); ys <- numeric(T)
            th[1] <- 0; ys[1] <- rnorm(1, sd = sqrt(MLE_V))
            for (t in 2:T){ th[t] <- th[t-1] + rnorm(1, sd = sqrt(MLE_W)); ys[t] <- th[t] + rnorm(1, sd = sqrt(MLE_V)) }
            ys
    })
    T_obs <- var(y); T_rep <- apply(sim_y, 2, var); p_B <- mean(T_rep > T_obs); p_B

    # Pointwise log-likelihood (Gaussian observation) for LOO/WAIC
    library(loo)
    f <- dlmFilter(y, mod)
    mu <- dropFirst(f$m); sigma2 <- drop(mod$V)
    loglik <- matrix(dnorm(y, mean = mu, sd = sqrt(sigma2), log = TRUE), ncol = 1)
    print(loo(loglik)); print(waic(loglik))
```

MCMC variants

Metropolis–Hastings (Section 2.2.2)

Outputs: Figure 2.9.

Listing 20: Random-walk Metropolis–Hastings for (V, W).

```r
    set.seed(123)
    T <- 100; true_V <- 0.8; true_W <- 0.3

    # Simulate once
    th <- numeric(T); y <- numeric(T); th[1] <- 0
    for (t in 2:T){
            th[t] <- th[t-1] + rnorm(1, 0, sqrt(true_W))
            y[t]  <- th[t] + rnorm(1, 0, sqrt(true_V))
    }

    logpost <- function(lp){  # flat priors on log-scale; returns log posterior up to const
            V <- exp(lp[1]); W <- exp(lp[2]); if (V <= 0 || W <= 0) return(-Inf)
            -dlmLL(y, dlmModPoly(1, dV = V, dW = W))
    }

    n_iter <- 5000; out <- matrix(NA_real_, n_iter, 2)
    cur <- log(c(1,1)); cur_lp <- logpost(cur); step <- c(0.1, 0.1)
    for (i in 1:n_iter){
            cand <- rnorm(2, cur, step); cand_lp <- logpost(cand)
            if (log(runif(1)) < (cand_lp - cur_lp)) { cur <- cand; cur_lp <- cand_lp }
            out[i,] <- exp(cur)
    }

    par(mfrow = c(2,1))
    plot(out[,1], type = "l", main = "Trace plot of V", ylab = "V", xlab = "Iteration")
    plot(out[,2], type = "l", main = "Trace plot of W", ylab = "W", xlab = "Iteration")
```

```
par(mfrow = c(1,1))
```

Hamiltonian Monte Carlo / NUTS (Section 2.2.2)

Outputs: — (in-text plots; no labeled figure in the chapter).

Listing 21: Stan model for the local-level DLM (save as `local_level.stan`).

```
/* Stan code */
data { int<lower=1> N; vector[N] y; }
parameters { real<lower=0> V; real<lower=0> W; vector[N] theta; }
model {
        V ~ inv_gamma(1, 1);
        W ~ inv_gamma(1, 1);
        theta[1] ~ normal(0, sqrt(W));
        for (t in 2:N) theta[t] ~ normal(theta[t-1], sqrt(W));
        for (t in 1:N) y[t] ~ normal(theta[t], sqrt(V));
}
```

Listing 22: Fit with HMC/NUTS in `rstan` and trace plots via `bayesplot`.

```
library(rstan); library(bayesplot); rstan_options(auto_write = TRUE)
set.seed(123)
T <- 100
# Simulate data (or reuse 'y')
theta <- numeric(T); y <- numeric(T); theta[1] <- 0
W_true <- 0.2; V_true <- 1
for (t in 2:T) theta[t] <- theta[t-1] + rnorm(1, sd = sqrt(W_true))
y <- theta + rnorm(T, sd = sqrt(V_true))

stan_data <- list(N = T, y = as.vector(y))

# Compile from file saved in Listing~\ref{lst:stan-model}
fit_hmc <- stan(file = "local_level.stan", data = stan_data,
iter = 2000, chains = 2, seed = 123)

# Trace plots with posterior means
post  <- as.array(fit_hmc)
means <- colMeans(as.data.frame(fit_hmc)[, c("V","W")])

mcmc_trace(post, pars = c("V","W")) +
ggplot2::geom_hline(yintercept = means["V"], color = "blue", linetype = "
dashed") +
ggplot2::geom_hline(yintercept = means["W"], color = "red",  linetype = "
dashed") +
ggplot2::ggtitle("Trace Plots for V and W (NUTS)")
```

```r

# Posterior histograms
color_scheme_set("blue")
mcmc_hist(post, pars = c("V","W")) +
ggplot2::ggtitle("Posterior Distributions of V and W (NUTS)")
```

Compact Kalman walk-through (Section 2.2.2)

Outputs: — (console values only).

Listing 23: Compact simulation, Kalman filter and log-likelihood (teaching version).

```r
# Known parameters
T <- 100; V <- 0.8; W <- 0.3
theta <- numeric(T); y <- numeric(T)

# Initialization
theta[1] <- rnorm(1, 0, sqrt(W))
y[1]     <- theta[1] + rnorm(1, 0, sqrt(V))

# Simulate process
for (t in 2:T) {
        theta[t] <- theta[t - 1] + rnorm(1, 0, sqrt(W))
        y[t]     <- theta[t] + rnorm(1, 0, sqrt(V))
}

# Kalman filter arrays
m <- numeric(T); C <- numeric(T); a <- numeric(T); Q <- numeric(T)
m[1] <- 0; C[1] <- 1
for (t in 1:T) {
        if (t > 1) { a[t] <- m[t-1]; Q[t] <- C[t-1] + V } else { a[t] <-
    m[1]; Q[t] <- C[1] + V }
        e_t <- y[t] - a[t]; K_t <- C[t] / Q[t]
        m[t] <- a[t] + K_t * e_t
        C[t+1] <- (1 - K_t) * C[t] + W
}
logL <- -0.5 * sum(log(2 * pi * Q) + ((y - a)^2) / Q)
logL
```

.3 Empirical Analysis with Dynamic Linear Models

Frequentist approach

Project setup: packages, graphics device, rounding helper; (Section 3.2)

Outputs: none directly; initializes environment for figures such as 3.2, 3.4, 3.3.

Listing 24: Packages, graphics reset, RStudio device, and a helper to round numeric columns.

```
        graphics.off()
        suppressPackageStartupMessages({
                library(readxl)
                library(dlm)
                library(tseries)
                library(dplyr)
        })
        # Ensure RStudio plot device
        options(device = "RStudioGD")

        # Round numeric columns inside data frames
        pround <- function(df, k = 3) dplyr::mutate(df, dplyr::across(where(is.
    numeric), ~round(.x, k)))
```

Utility functions used throughout ; (Sections 3.3, 3.2)

Outputs: helpers for forecasts/plots used to produce 3.7, 3.6, 3.17, and tables 3.6, 3.13.

Listing 25: Holdout split, Gaussian PI builder, extraction of last filtered state, manual h-step forecasting, forecast plotting, RMSE/MAE, and state mean/SD utilities.

```
        ts_split_holdout <- function(y, h){
                stopifnot(is.ts(y), h >= 1, length(y) > h)
                n  <- length(y)
                tr <- window(y, end   = time(y)[n - h])
                te <- window(y, start = time(y)[n - h + 1])
                list(train = tr, test = te)
        }

        make_ci <- function(mean, sd, level = 0.90){
                a <- (1 - level)/2
                list(lwr = mean + qnorm(a)*sd, upr = mean + qnorm(1-a)*sd)
        }

        get_last_state <- function(filt){
                m_last <- if (is.null(dim(filt$m))) as.numeric(tail(filt$m, 1)) else as.numeric(filt$m[nrow(filt$m), ])
                U <- filt$U.C; D <- filt$D.C
                if (!is.null(dim(U)) && length(dim(U)) == 3) {
                        k <- dim(U)[3]
                        C_last <- dlmSvd2var(U[,,k, drop=FALSE], D[,,k, drop=FALSE])[,,1]
                } else {
                        U_k <- if (is.matrix(U)) matrix(U[1, ncol(U)], 1, 1) else matrix(tail(U,1), 1, 1)
                        D_k <- if (is.matrix(D)) matrix(D[1, ncol(D)], 1, 1) else matrix(tail(D,1), 1, 1)
                        C_last <- dlmSvd2var(U_k, D_k)
                }
                list(m = m_last, C = as.matrix(C_last))
        }

        manual_forecast <- function(mod, m_last, C_last, h){
                Fm <- as.matrix(mod$FF); Gm <- as.matrix(mod$GG)
                Wm <- as.matrix(mod$W);  Vv <- as.numeric(mod$V)
                m  <- matrix(m_last, ncol = 1)
                C  <- as.matrix(C_last)
                means <- numeric(h); sds <- numeric(h)
                for (i in 1:h){
                        m <- Gm %*% m
                        C <- Gm %*% C %*% t(Gm) + Wm
                        f <- as.numeric(Fm %*% m)
                        Q <- as.numeric(Fm %*% C %*% t(Fm) + Vv)
                        means[i] <- f; sds[i] <- sqrt(Q)
                }
                list(mean = means, sd = sds)
        }

        plot_forecast_ts <- function(y_all, mean_ts, lwr_ts, upr_ts, title="Forecast"){
                h <- length(mean_ts); freq <- frequency(y_all)
                fc_start <- time(y_all)[length(y_all) - h + 1]
                if (!is.ts(mean_ts)) mean_ts <- ts(as.numeric(mean_ts), start = fc_start, frequency = freq)
                if (!is.ts(lwr_ts))  lwr_ts  <- ts(as.numeric(lwr_ts),  start = start(mean_ts), frequency = freq)
                if (!is.ts(upr_ts))  upr_ts  <- ts(as.numeric(upr_ts),  start = start(mean_ts), frequency = freq)
                hist_idx <- max(1, length(y_all) - 2*h)
                y_hist <- window(y_all, start = time(y_all)[hist_idx])
```

```r
            plot(y_hist, type="l", col="black", xlab="Year", ylab="GDP", main=title)
            lines(mean_ts, col="steelblue", lwd=2)
            lines(lwr_ts,  col="orange", lty=2)
            lines(upr_ts,  col="orange", lty=2)
            y_obs_test <- window(y_all, start = start(mean_ts), end = end(mean_ts))
            if (length(y_obs_test) > 0) points(time(y_obs_test), as.numeric(y_obs_test), pch=19)
            legend("topleft", bty="n",
            legend=c("Observed","Forecast mean","90% prediction interval"),
            col=c("black","steelblue","orange"), lty=c(1,1,2), lwd=c(1,2,1))
    }

    rmse <- function(a,b) sqrt(mean((a-b)^2, na.rm = TRUE))
    mae  <- function(a,b) mean(abs(a-b), na.rm = TRUE)

    filt_state_mean <- function(filt, i = 1L) {
            M <- filt$m
            if (is.null(dim(M))) as.numeric(M)[-1] else as.numeric(M[-1, i])
    }

    smooth_state_mean <- function(smth, i = 1L) {
            S <- smth$s
            if (is.null(dim(S))) as.numeric(S)[-1] else as.numeric(S[-1, i])
    }

    smooth_state_sd <- function(smth, i = 1L) {
            U <- smth$U.S; D <- smth$D.S
            V <- dlmSvd2var(U, D)
            if (is.list(V)) {
                    v <- vapply(V[-1], function(M) M[i, i], numeric(1))
                    sqrt(v)
            } else if (is.array(V) && length(dim(V)) == 3) {
                    sqrt(drop(V[i, i, -1, drop = FALSE]))
            } else {
                    sqrt(as.numeric(V[-1]))
            }
    }

    as_ts_like_y <- function(x, y) ts(x, start = start(y) + 1 / frequency(y), frequency = frequency(y))
```

Model builders and MLE wrappers ; (Section 3.2)

Outputs: used to generate 3.3, 3.15, 3.7, 3.6, and selection Table 3.5.

Listing 26: LL and LLT parameterizations on log-scale for MLE, a thin wrapper around dlmMLE, and a forecast helper that falls back to manual recursions if needed.

```
    buildLL  <- function(p)  dlmModPoly(order = 1, dV = exp(p[1]), dW = exp(p[2]))
    buildLLT <- function(p)  dlmModPoly(order = 2, dV = exp(p[1]), dW = c(exp(p[2]), exp(p[3])))

    fit_mle <- function(y, build_fun, init){
            est <- dlmMLE(y, parm = init, build = build_fun, hessian = TRUE)
            if (est$conv != 0) warning("dlmMLE: possible non-convergence (conv != 0)")
            list(fit = est, mod = build_fun(est$par))
    }

    forecast_h <- function(y, build_fun, init, h, level = 0.90){
            sp <- ts_split_holdout(y, h)
            y_tr <- sp$train; y_te <- sp$test
            est  <- fit_mle(y_tr, build_fun, init)
            filt <- dlmFilter(y_tr, est$mod)
            if (is.null(dim(filt$m))) filt$m <- cbind(filt$m)
            if (!is.null(filt$a) && is.null(dim(filt$a))) filt$a <- cbind(filt$a)
            fc_try <- try(dlmForecast(filt, nAhead = h), silent = TRUE)
            ok <- !inherits(fc_try, "try-error")
            if (ok){
                    f_vec <- suppressWarnings(as.numeric(fc_try$f))
                    Q_vec <- suppressWarnings(as.numeric(fc_try$Q))
                    ok <- all(is.finite(f_vec)) && all(is.finite(Q_vec)) &&
                    length(f_vec) == h && length(Q_vec) == h
            }
            if (ok){
                    mean_ts <- ts(f_vec, start = start(y_te), frequency = frequency(y))
                    sd_ts   <- ts(sqrt(Q_vec), start = start(y_te), frequency = frequency(y))
            } else {
                    message("dlmForecast not usable: using manual state-space recursions")
                    last <- get_last_state(filt)
                    man  <- manual_forecast(est$mod, last$m, last$C, h)
                    mean_ts <- ts(man$mean, start = start(y_te), frequency =
    frequency(y))
                          sd_ts   <- ts(man$sd,   start = start(y_te), frequency =
    frequency(y))
                  }
                  ci  <- make_ci(mean_ts, sd_ts, level)
                  list(model = est$mod, mean = mean_ts, sd = sd_ts,
                  lwr = ts(ci$lwr, start = start(mean_ts), frequency = frequency(y)
    ),
                  upr = ts(ci$upr, start = start(mean_ts), frequency = frequency(y)
    ),
                  test = y_te)
        }
```

Data import and initial plot ; (Section 3.1)

Outputs: Figure 3.2.

Listing 27: Load the Excel file (column ’GDP’, sheet ’Foglio1’), build the annual series `y_all`, its NA-free `y_clean`, and plot the observations.

```
        file_xlsx  <- "macro_italia_missingdata.xlsx"
        sheet_nm   <- "Foglio1"
        start_year <- 1960

        if (!file.exists(file_xlsx)) stop("Excel file not found: ", file_xlsx)
        raw <- readxl::read_excel(file_xlsx, sheet = sheet_nm)
        if (!("GDP" %in% names(raw))) stop("Column GDP not found in the sheet")

        y_all   <- ts(as.numeric(raw$GDP), start = start_year, frequency = 1)
        y_clean <- stats::na.omit(y_all)
        if (length(y_clean) < 20) stop("Clean series too short (<20)")

        cat("Series length (all):", length(y_all), "| after removing NA:", length(y_clean), "\n")

        graphics.off()
        par(mar = c(5,5,3,2))
        plot(y_all, type="l", lwd=1.6, col="black",
        xlab="Year", ylab="GDP growth rate (%)",
        main="Italy - Real GDP growth rate (1960-2024)")
        abline(h = 0, lty = 3, col = "gray50")
```

Descriptives and stationarity checks ; (Section “Stationarity and integration”)

Outputs: Table 3.1.

Listing 28: Descriptives on the cleaned series and ADF/KPSS tests on levels and first differences.

```
        cat("\n--- Descriptives (y_clean) ---\n")
        cat("Mean:", mean(y_clean), " Var:", var(y_clean),
        " Min:", min(y_clean), " Max:", max(y_clean), "\n")

```

```r
      cat("\n--- ADF/KPSS on levels ---\n")
      print(tseries::adf.test(y_clean, alternative = "stationary"))
      print(tseries::kpss.test(y_clean, null = "Level"))

      cat("\n--- ADF/KPSS on first differences ---\n")
      dy <- diff(y_clean); dy <- dy[is.finite(dy)]
      print(tseries::adf.test(dy, alternative = "stationary"))
      print(tseries::kpss.test(dy, null = "Level"))
```

ACF and PACF of the series ; (Section "Serial–dependence patterns")

Outputs: Figure 3.4.

Listing 29: Autocorrelation and partial autocorrelation of `y_clean` with 40 lags.

```r
      graphics.off()
      par(mfrow = c(1,2))
      acf_obj <- acf(y_clean, lag.max = 40, plot = FALSE)
      plot(acf_obj, main = "Autocorrelation function of GDP (clean)",
      xlab = "Lag", ylab = "Autocorrelation", col = "black")
      points(x = drop(acf_obj$lag), y = drop(acf_obj$acf), col = "red", pch =
  19)

      pacf_obj <- pacf(y_clean, lag.max = 40, plot = FALSE)
      plot(pacf_obj, main = "Partial autocorrelation function of GDP (clean)",
      xlab = "Lag", ylab = "Partial autocorrelation", col = "black")
      points(x = drop(pacf_obj$lag), y = drop(pacf_obj$acf), col = "red", pch =
   19)
      par(mfrow = c(1, 1))
```

Full-sample MLE fit of Local Level ; (Section 3.2)

Outputs: Figure 3.3; parameter CIs used in Table 3.2.

Listing 30: Fit LL via MLE, plot observed vs filtered/smoothed level and its 90% bands; compute asymptotic CIs for V and W.

```r
      init_LL_full  <- log(c(var(y_clean), 0.1 * var(y_clean)))
      est_LL_full   <- fit_mle(y_clean, buildLL, init_LL_full)
      filt_LL       <- dlmFilter(y_clean, est_LL_full$mod)
      smth_LL       <- dlmSmooth(filt_LL)

      mu_filt_LL <- filt_state_mean(filt_LL, 1L)
      mu_smth_LL <- smooth_state_mean(smth_LL, 1L)

      graphics.off()
```

```r
        ts.plot(
        y_clean,
        as_ts_like_y(mu_filt_LL, y_clean),
        as_ts_like_y(mu_smth_LL, y_clean),
        col = c("gray0","darkolivegreen4","firebrick"),
        lty = c(1,2,3), lwd = 1.5,
        ylab = "GDP", xlab = "Year",
        main = "Local Level (MLE): observed vs filtered/smoothed"
        )
        legend("bottom", inset = c(0,-0.25), xpd = TRUE, bty = "n", horiz = TRUE,
     cex = 1,
        legend = c("Observed","Filtered","Smoothed"),
        col = c("gray0","darkolivegreen4","firebrick"),
        lty = c(1,2,3))
        se_mu_LL <- smooth_state_sd(smth_LL, 1L)
        mu_ts_LL <- as_ts_like_y(mu_smth_LL, y_clean)
        lines(mu_ts_LL + qnorm(.95) * se_mu_LL, lty = 3, col = "chocolate")
        lines(mu_ts_LL + qnorm(.05) * se_mu_LL, lty = 3, col = "chocolate")
        cat("Plotted Local Level - filtered & smoothed\n")

        vc <- try(solve(est_LL_full$fit$hessian), silent = TRUE)
        if (!inherits(vc, "try-error")) {
                se <- sqrt(diag(vc))
                ci <- cbind(Estimate = exp(est_LL_full$fit$par),
                Lower = exp(est_LL_full$fit$par - 1.96*se),
                Upper = exp(est_LL_full$fit$par + 1.96*se))
                cat("\nAsymptotic confidence intervals (Local Level):\n"); print(
    round(ci,4))
        }
```

Full-sample MLE fit of Local Linear Trend ; (Appendix 3.6)

Outputs: Figure 3.15.

Listing 31: Fit LLT via MLE, plot observed vs filtered/smoothed level with 90% bands, and inspect ACF/PACF of the smoothed level.

```r
        init_LLT_full <- log(c(var(y_clean), 0.1 * var(y_clean), 0.01 * var(y_
    clean)))
        est_LLT_full  <- fit_mle(y_clean, buildLLT, init_LLT_full)
        filt_LLT      <- dlmFilter(y_clean, est_LLT_full$mod)
        smth_LLT      <- dlmSmooth(filt_LLT)

        mu_filt_LLT <- filt_state_mean(filt_LLT, 1L)
        mu_smth_LLT <- smooth_state_mean(smth_LLT, 1L)

        ts.plot(
        y_clean,
        as_ts_like_y(mu_filt_LLT, y_clean),
        as_ts_like_y(mu_smth_LLT, y_clean),
        col = c("gray0","darkolivegreen4","firebrick"),
        lty = c(1,2,3), lwd = 1.5,
        ylab = "GDP", xlab = "Year",
        main = "Local Linear Trend (MLE): observed vs filtered/smoothed"
        )
        legend("bottom", inset = c(0,-0.25), xpd = TRUE, bty = "n", horiz = TRUE, cex = 1,
        legend = c("Observed","Filtered (level)","Smoothed (level)"),
        col = c("gray0","darkolivegreen4","firebrick"),
        lty = c(1,2,3))
        se_mu_LLT <- smooth_state_sd(smth_LLT, 1L)
        mu_ts_LLT <- as_ts_like_y(mu_smth_LLT, y_clean)
        lines(mu_ts_LLT + qnorm(.95) * se_mu_LLT, lty = 3, col = "chocolate")
        lines(mu_ts_LLT + qnorm(.05) * se_mu_LLT, lty = 3, col = "chocolate")

        graphics.off()
        mu_ts_LLT <- as_ts_like_y(mu_smth_LLT, y_clean)
        par(mfrow = c(1,2))
        acf_mu <- acf(mu_ts_LLT, lag.max = 40, plot = FALSE)
        plot(acf_mu, main = "ACF of estimated level (LLT)", xlab = "Lag", ylab = "ACF", col = "black")
        points(x = drop(acf_mu$lag), y = drop(acf_mu$acf), col = "red", pch = 19)
        pacf_mu <- pacf(mu_ts_LLT, lag.max = 40, plot = FALSE)
        plot(pacf_mu, main = "PACF of estimated level (LLT)", xlab = "Lag", ylab = "PACF", col = "black")
        points(x = drop(pacf_mu$lag), y = drop(pacf_mu$acf), col = "red", pch = 19)
        par(mfrow = c(1,1))
```

Residual diagnostics for LL and LLT ; (Section “Residual diagnostics”)

Outputs: Figure 3.5 (LL) and Figure 3.16 (LLT); Table 3.4.

Listing 32: Residual diagnostics on standardized one-step-ahead residuals for LL and LLT, including Ljung-Box, Shapiro-Wilk, and Jarque-Bera; plus a compact table.

```r
    res_obj <- residuals(filt_LL, sd = TRUE)
    zres    <- drop(res_obj$res)/drop(res_obj$sd)
    zres    <- zres[is.finite(zres)]
    par(mfrow = c(2,2))
    acf_res  <- acf(zres,  lag.max = 40, plot = FALSE)
    plot(acf_res,  main="ACF of standardized residuals (LL)",  xlab="Lag", ylab="ACF",  col="black")
    points(x = drop(acf_res$lag),  y = drop(acf_res$acf),  col="red", pch=19)
    pacf_res <- pacf(zres, lag.max = 40, plot = FALSE)
    plot(pacf_res, main="PACF of standardized residuals (LL)", xlab="Lag", ylab="PACF", col="black")
    points(x = drop(pacf_res$lag), y = drop(pacf_res$acf), col="red", pch=19)
    qqnorm(zres, main="QQ-plot of standardized residuals"); qqline(zres)
    hist(zres, breaks=12, main="Histogram of standardized residuals", xlab="Standardized residual")
    par(mfrow = c(1,1))
    cat("\nLjung-Box(20):\n"); print(Box.test(zres, lag=20, type="Ljung-Box"))
    cat("\nShapiro-Wilk:\n");  print(shapiro.test(zres))
    cat("\nJarque-Bera:\n");   print(tseries::jarque.bera.test(zres))

    diag_one <- function(filt) {
            r  <- residuals(filt, sd = TRUE)
            z  <- drop(r$res)/drop(r$sd)
            z  <- z[is.finite(z)]
            lb <- Box.test(z, lag = 20, type = "Ljung-Box")
            sw <- shapiro.test(z)
            jb <- tseries::jarque.bera.test(z)
            data.frame(
            LB20_stat = unname(lb$statistic), LB20_p = unname(lb$p.value),
            SW_W      = unname(sw$statistic), SW_p   = unname(sw$p.value),
            JB_stat   = unname(jb$statistic), JB_p   = unname(jb$p.value)
            )
    }
    diag_tab <- rbind(LL = diag_one(filt_LL), LLT = diag_one(filt_LLT))
    print(pround(diag_tab))

    # Dedicated LLT residual plots + tests
    res_obj_LLT <- residuals(filt_LLT, sd = TRUE)
    zres_LLT    <- drop(res_obj_LLT$res) / drop(res_obj_LLT$sd)
    zres_LLT    <- zres_LLT[is.finite(zres_LLT)]
    op <- par(mfrow = c(2,2)); on.exit(par(op), add = TRUE)
    acf_res_LLT  <- acf(zres_LLT,  lag.max = 40, plot = FALSE)
    plot(acf_res_LLT, main = "ACF of standardized residuals (LLT)", xlab = "Lag", ylab = "ACF", col = "black")
    points(x = drop(acf_res_LLT$lag), y = drop(acf_res_LLT$acf), col = "red", pch = 19)
    pacf_res_LLT <- pacf(zres_LLT, lag.max = 40, plot = FALSE)
    plot(pacf_res_LLT, main = "PACF of standardized residuals (LLT)", xlab = "Lag", ylab = "PACF", col = "black")
    points(x = drop(pacf_res_LLT$lag), y = drop(pacf_res_LLT$acf), col = "red", pch = 19)
    qqnorm(zres_LLT, main = "QQ-plot of standardized residuals (LLT)"); qqline(zres_LLT)
    hist(zres_LLT, breaks = 12, main = "Histogram of standardized residuals (LLT)", xlab = "Standardized residual")
    cat("\nLjung-Box(20) - LLT:\n");  print(Box.test(zres_LLT, lag = 20, type = "Ljung-Box"))
    cat("\nShapiro-Wilk  - LLT:\n");  print(shapiro.test(zres_LLT))
    cat("\nJarque-Bera   - LLT:\n");  print(tseries::jarque.bera.test(zres_LLT))
```

Model selection via information criteria ; (Section "Model assessment for forecasting")

Outputs: Table 3.5.

Listing 33: AIC/BIC comparison between LL and LLT on the full sample.

```r
    logLik_LL  <- -est_LL_full$fit$value;  k_LL  <- length(est_LL_full$fit$par)
    logLik_LLT <- -est_LLT_full$fit$value; k_LLT <- length(est_LLT_full$fit$par)
    Tn <- length(y_clean)
    tab_sel <- data.frame(
    Model=c("Local Level","Local Linear Trend"),
    logLik=c(logLik_LL, logLik_LLT),
    AIC = c(-2*logLik_LL  + 2*k_LL,
    -2*logLik_LLT + 2*k_LLT),
    BIC = c(-2*logLik_LL  + log(Tn)*k_LL,
    -2*logLik_LLT + log(Tn)*k_LLT)
    )
    cat("\nModel selection (AIC/BIC):\n"); print(pround(tab_sel))
```

Fixed out-of-sample holdout: $h = 8$ (LL and LLT) ; (Section "Model assessment for forecasting")

Outputs: Figure 3.7; contributes to Table 3.6 and Table 3.13.

Listing 34: Eight-step holdout under LL and LLT with MLE, plotted forecasts with 90% PIs, RMSE/MAE and empirical coverage.

```r
    h <- 8L; stopifnot(h < length(y_clean))
    init_LL  <- log(c(var(y_clean), 0.1*var(y_clean)))
    init_LLT <- log(c(var(y_clean), 0.1*var(y_clean), 0.01*var(y_clean)))

    fc_LL   <- forecast_h(y_clean, buildLL,  init_LL,  h = h, level = 0.90)
    plot_forecast_ts(y_clean, fc_LL$mean, fc_LL$lwr, fc_LL$upr,
    sprintf("Local Level - %d-step forecast (holdout)", h))

    fc_LLT  <- forecast_h(y_clean, buildLLT, init_LLT, h = h, level = 0.90)
    plot_forecast_ts(y_clean, fc_LLT$mean, fc_LLT$lwr, fc_LLT$upr,
    sprintf("Local Linear Trend - %d-step forecast (holdout)", h))

    y_te <- as.numeric(fc_LL$test)
    cmp <- data.frame(
    Model = c("Local Level","Local Linear Trend"),
    RMSE  = c(rmse(as.numeric(fc_LL$mean),  y_te),
    rmse(as.numeric(fc_LLT$mean), y_te)),
    MAE   = c(mae(as.numeric(fc_LL$mean),   y_te),
    mae(as.numeric(fc_LLT$mean),  y_te))
    )
    cat("\nHoldout comparison (RMSE/MAE):\n"); print(pround(cmp))
    cov90_LL  <- mean(y_te >= as.numeric(fc_LL$lwr)  & y_te <= as.numeric(fc_LL$upr))
    cov90_LLT <- mean(y_te >= as.numeric(fc_LLT$lwr) & y_te <= as.numeric(fc_LLT$upr))
    cat("\nCoverage 90% (holdout) - Local Level:", round(cov90_LL,3),
    "  Local Linear Trend:", round(cov90_LLT,3), "\n")
```

Rolling-origin cross-validation ($h = 4$) ; (Section 3.3)

Outputs: Figure 3.6, Figure 3.17; Table 3.6, Table 3.13; horizon profiles in Appendix figures.

Listing 35: Rolling-origin CV with last 8 origins (h = 4) for LL and LLT, plus a compact multi-horizon check for LL over h in {1,4,8}.

```
    rolling_cv <- function(y, build_fun, init, h=4, n_origins=8, level=0.90){
            n <- length(y)
            origins <- (n - h - n_origins + 1):(n - h)
            out <- lapply(origins, function(i){
                    y_tr <- window(y, end = time(y)[i])
                    y_te <- window(y, start = time(y)[i+1], end = time(y)[i+h])
                    est  <- fit_mle(y_tr, build_fun, init)
                    filt <- dlmFilter(y_tr, est$mod)
                    fc_try <- try(dlmForecast(filt, nAhead = h), silent = TRUE)
                    if (!inherits(fc_try, "try-error")){
```

```r
                f  <- as.numeric(fc_try$f); sd <- sqrt(as.numeric(fc_try$Q))
            } else {
                last <- get_last_state(filt)
                man  <- manual_forecast(est$mod, last$m, last$C, h)
                f <- man$mean; sd <- man$sd
            }
            ci <- make_ci(f, sd, level)
            list(mean=f, lwr=ci$lwr, upr=ci$upr, y=as.numeric(y_te))
        })
        pred <- unlist(lapply(out, `[[`, "mean"))
        ytru <- unlist(lapply(out, `[[`, "y"))
        lwr  <- unlist(lapply(out, `[[`, "lwr"))
        upr  <- unlist(lapply(out, `[[`, "upr"))
        data.frame(RMSE = rmse(pred, ytru),
        MAE  = mae(pred, ytru),
        Cov90= mean(ytru >= lwr & ytru <= upr),
        AvgWidth = mean(upr - lwr))
    }

    cat("\nRolling CV (last 8 origins, h=4):\n")
    rcv_LL  <- rolling_cv(y_clean, buildLL,  init_LL,  h=4, n_origins=8)
    rcv_LLT <- rolling_cv(y_clean, buildLLT, init_LLT, h=4, n_origins=8)
    print(pround(rbind(LL=rcv_LL, LLT=rcv_LLT)))

    eval_multi_h <- function(y, build_fun, init, hs=c(1,4,8), level=0.90){
        do.call(rbind, lapply(hs, function(h){
            fc <- forecast_h(y, build_fun, init, h, level)
            y_te <- as.numeric(fc$test); pred <- as.numeric(fc$mean)
            lwr <- as.numeric(fc$lwr);   upr <- as.numeric(fc$upr)
            data.frame(h=h,
            RMSE = rmse(pred, y_te),
            MAE  = mae(pred, y_te),
            Cov90 = mean(y_te >= lwr & y_te <= upr),
            AvgWidth = mean(upr - lwr))
        }))
    }
    cat("\nMulti-horizon evaluation (Local Level):\n")
    res_LL_h <- eval_multi_h(y_clean, buildLL, init_LL, hs=c(1,4,8), level=0.90)
    print(pround(res_LL_h))
```

Missing data and backcast (1950–1959) ; (Section “Handling missing data”)

Outputs: Figure 3.8.

Listing 36: Utilities to extract observation bands from the smoother and to demonstrate a backcast for 1950-1959 under LL as a single panel (no zoom).

```
  get_obs_bands_from_smoother <- function(smth, mod, y_ts, level = 0.90, which_state = 1L) {
      mu_vec <- smooth_state_mean(smth, i = which_state)
      se_mu  <- smooth_state_sd(smth,  i = which_state)
      sd_y   <- sqrt(se_mu^2 + as.numeric(mod$V))
      mu_ts  <- ts(mu_vec, start = start(y_ts) + 1/frequency(y_ts), frequency = frequency(y_ts))
      lwr_ts <- mu_ts + qnorm((1 - level)/2) * ts(sd_y, start = start(mu_ts), frequency = frequency(mu_ts))
      upr_ts <- mu_ts + qnorm(1 - (1 - level)/2) * ts(sd_y, start = start(mu_ts), frequency = frequency(mu_ts))
      list(mu = mu_ts, lwr = lwr_ts, upr = upr_ts)
  }

  demo_missing_block <- function(y, gap_start, gap_end, level = 0.90) {
      stopifnot(is.ts(y), gap_start >= start(y)[1], gap_end <= end(y)[1])
      t_idx <- time(y); mask  <- t_idx >= gap_start & t_idx <= gap_end
      y_mask <- y; y_true <- y_mask[mask]; y_mask[mask] <- NA
      est  <- fit_mle(stats::na.omit(y_mask), buildLL,
      log(c(var(na.omit(y_mask)), 0.1*var(na.omit(y_mask)))))
      mod  <- est$mod; filt <- dlmFilter(y_mask, mod); smth <- dlmSmooth(filt)
      bands <- get_obs_bands_from_smoother(smth, mod, y_ts = y_mask, level = level, which_state = 1L)
      mu_ts <- bands$mu; lwr <- bands$lwr; upr <- bands$upr
      imp      <- window(mu_ts,  start = gap_start, end = gap_end)
      lwr_imp  <- window(lwr,    start = gap_start, end = gap_end)
      upr_imp  <- window(upr,    start = gap_start, end = gap_end)
      rmse_imp <- sqrt(mean((as.numeric(imp) - as.numeric(y_true))^2, na.rm = TRUE))
      mae_imp  <- mean(abs(as.numeric(imp) - as.numeric(y_true)), na.rm = TRUE)
      cov_imp  <- mean(as.numeric(y_true) >= as.numeric(lwr_imp) &
      as.numeric(y_true) <= as.numeric(upr_imp), na.rm = TRUE)
      ts.plot(y_mask, mu_ts,
      col = c("gray0","firebrick"), lty = c(1,3), lwd = 1.5,
      ylab = "GDP", xlab = "Year",
      main = sprintf("LL: mask-and-score %d-%d", gap_start, gap_end))
```

```r
        lines(lwr, col="orange", lty=2); lines(upr, col="orange", lty=2)
        rect(gap_start, par("usr")[3], gap_end, par("usr")[4], col=rgb(0,0,1,0.06), border=NA)
        points(time(window(y, start = gap_start, end = gap_end)), as.numeric(y_true), pch=19)
        legend("topleft", bty="n",
        legend=c("Observed (with NA)", "Smoothed mean", sprintf("%d%% PI", round(level*100))),
        col=c("gray0","firebrick","orange"), lty=c(1,3,2), lwd=c(1.5,1.5,1))
        cat("\n[Mask-and-score ", gap_start, "-", gap_end, "]  RMSE:", round(rmse_imp,3),
        "  MAE:", round(mae_imp,3), "  Coverage:", round(cov_imp,3), "\n")
        invisible(list(rmse = rmse_imp, mae = mae_imp, coverage = cov_imp))
    }

    # Backcast, single-panel (no zoom) - 1950-1959
    backcast_prefix <- function(y, start_new = 1950, level = 0.90){
        f <- frequency(y)
        add <- start(y)[1] - start_new
        stopifnot(add >= 1)
        y_ext <- ts(c(rep(NA_real_, add * f), as.numeric(y)), start = start_new, frequency = f)
        est  <- fit_mle(stats::na.omit(y_ext), buildLL,
        log(c(var(na.omit(y_ext)), 0.1*var(na.omit(y_ext)))))
        mod  <- est$mod; filt <- dlmFilter(y_ext, mod); smth <- dlmSmooth(filt)
        bands <- get_obs_bands_from_smoother(smth, mod, y_ts = y_ext, level = level, which_state = 1L)
        mu_ts <- bands$mu; lwr <- bands$lwr; upr <- bands$upr
        ts.plot(y_ext, mu_ts,
        col = c("gray0","firebrick"), lty = c(1,3), lwd = 1.5,
        ylab = "GDP", xlab = "Year",
        main = sprintf("LL backcast: %d-%d (single panel)", start_new, start(y)[1]-1))
        lines(lwr, col="orange", lty=2); lines(upr, col="orange", lty=2)
        rect(start_new, par("usr")[3], start(y)[1] - 1, par("usr")[4], col=rgb(0,0,1,0.06), border=NA)
        invisible(list(series_ext = y_ext, mu = mu_ts, lwr = lwr, upr = upr))
    }
```

Rolling-origin details and fan charts ; (Section “Out–of–sample evidence”)

Outputs: Figure 3.6, 3.17, horizon profiles panels.

Listing 37: Detailed rolling-origin paths for fan charts and enhanced visualization of per-horizon intervals and misses.

```r
    rolling_cv_with_details <- function(y, build_fun, init, h=4, n_origins=8, level=.90){
            n <- length(y); origins <- (n - h - n_origins + 1):(n - h)
            details <- do.call(rbind, lapply(origins, function(i){
                    y_tr <- window(y, end = time(y)[i])
                    y_te <- window(y, start = time(y)[i+1], end = time(y)[i+h])
                    est  <- fit_mle(y_tr, build_fun, init)
                    filt <- dlmFilter(y_tr, est$mod)
                    fc_try <- try(dlmForecast(filt, nAhead = h), silent = TRUE)
                    if (!inherits(fc_try, "try-error")){
                            f  <- as.numeric(fc_try$f)
                            sd <- sqrt(as.numeric(fc_try$Q))
                    } else {
                            last <- get_last_state(filt)
                            man  <- manual_forecast(est$mod, last$m, last$C, h)
                            f  <- as.numeric(man$mean)
                            sd <- as.numeric(man$sd)
                    }
                    ci <- make_ci(f, sd, level)
                    data.frame(
                    origin_end = rep(as.numeric(time(y_tr)[length(y_tr)]), h),
                    step = 1:h,
                    time = as.numeric(time(y_te)),
                    mean = f,
                    lwr  = as.numeric(ci$lwr),
                    upr  = as.numeric(ci$upr),
                    y    = as.numeric(y_te)
                    )
            }))
            summary <- data.frame(
            RMSE = sqrt(mean((details$y - details$mean)^2)),
            MAE  = mean(abs(details$y - details$mean)),
            Cov90 = mean(details$y >= details$lwr & details$y <= details$upr),
            AvgWidth = mean(details$upr - details$lwr)
```

```r
        )
        list(details=details, summary=summary)
    }

    plot_rolling_fan <- function(df, y, title="Rolling-origin fan chart"){
        rng <- range(df$time, na.rm = TRUE)
        y_seg <- window(y, start = rng[1], end = rng[2])
        plot(y_seg, type="l", col="black", lwd=1.2, xlab="Year", ylab="GDP", main=title)
        by_origin <- split(df, df$origin_end)
        invisible(lapply(by_origin, function(sub){
            segments(sub$time, sub$lwr, sub$time, sub$upr, col=rgb(1,0.5,0,0.45))
            points(sub$time, sub$mean, pch=16, cex=.75, col="steelblue")
        }))
        legend("topleft", bty="n", legend=c("Observed","Forecast mean","90% PI"),
        col=c("black","steelblue",rgb(1,0.5,0,1)), lty=c(1,NA,1), pch=c(NA,16,NA))
    }

    plot_rolling_fan_enhanced <- function(df, y, title = "Rolling CV fan - Local Level (enhanced)") {
        stopifnot(is.ts(y))
        req <- c("time","step","lwr","upr","y","mean")
        miss <- setdiff(req, names(df))
        if (length(miss)) stop("df missing: ", paste(miss, collapse = ", "))
        df <- df[is.finite(df$time) & is.finite(df$step), , drop = FALSE]
        if (!nrow(df)) stop("df is empty after cleaning")
        op <- par(no.readonly = TRUE); on.exit(par(op), add = TRUE)
        par(mfrow = c(1,1), mar = c(7,4,3,1), xpd = NA)
        rng <- range(df$time, na.rm = TRUE)
        y_seg <- window(y, start = rng[1], end = rng[2])
        plot(y_seg, type = "l", col = "black", lwd = 1.4, xlab = "Year", ylab = "GDP", main = title)
        steps    <- sort(unique(df$step))
        pch_step <- c(16, 17, 15, 18)[seq_along(steps)]
        dx_step  <- (steps - mean(steps)) * 0.06
        col_bar  <- rgb(1, 0.5, 0, 0.40)
        for (i in seq_along(steps)) {
            sub <- df[df$step == steps[i], , drop = FALSE]
            x   <- sub$time + dx_step[i]
            segments(x0 = x, y0 = sub$lwr, x1 = x, y1 = sub$upr, col
= col_bar)
                          miss <- (sub$y < sub$lwr) | (sub$y > sub$upr)
                          points(x[!miss], sub$mean[!miss], pch = pch_step[i], cex
= .9, col = "steelblue")
                          points(x[miss],  sub$mean[miss],  pch = pch_step[i], cex
= 1.0, col = "red")
                  }
                  legend("bottom", inset = c(0, -0.22), bty = "n",
                  legend = c("Observed", paste0("Mean (h=", steps, ")"), "90% PI",
"Missed PI"),
                  col    = c("black", rep("steelblue", length(steps)), col_bar, "
red"),
                  lty    = c(1, rep(NA, length(steps)+2)),
                  pch    = c(NA, pch_step, NA, 16),
                  pt.cex = c(NA, rep(.9, length(steps)), NA, 1),
                  horiz  = TRUE, ncol = 2)
        }
```

Rolling-origin visual summaries ; (Section “Out–of–sample evidence”)

Outputs: Bar chart akin to Figure 3.22; fans and per-horizon profiles for LL/LLT.

Listing 38: Barplot of rolling metrics when available, detailed paths for LL and LLT, fan charts, and multi-horizon panels for LL and LLT.

```
        # Barplot if rcv_* exist
        if (exists("rcv_LL", inherits = TRUE) && exists("rcv_LLT", inherits =
TRUE)) {
                metrics_mat <- rbind(LL  = unlist(rcv_LL[c("RMSE","MAE","Cov90","
AvgWidth")]),
                LLT = unlist(rcv_LLT[c("RMSE","MAE","Cov90","AvgWidth")]))
                M <- t(metrics_mat)
                op <- par(no.readonly = TRUE); on.exit(par(op), add = TRUE)
                par(mar = c(5,4,3,1))
                bp <- barplot(M, beside = TRUE,
                col = c("steelblue","tomato"),
                ylim = c(0, max(M)*1.25),
                ylab = "Value",
                main = "Rolling CV (h=4, last 8 origins) - metrics")
                legend("topright", bty="n", fill=c("steelblue","tomato"), legend=
colnames(M))
                abline(h = 0.90, lty = 3) # target 90%
                text(x = bp, y = as.vector(M), labels = round(as.vector(M), 3),
pos = 3, cex = .8)
        }

```

```r
    init_LL  <- log(c(var(y_clean), 0.1*var(y_clean)))
    init_LLT <- log(c(var(y_clean), 0.1*var(y_clean), 0.01*var(y_clean)))

    rcv_LL_det  <- rolling_cv_with_details(y_clean, buildLL,  init_LL,  h=4, n_origins=8)
    rcv_LLT_det <- rolling_cv_with_details(y_clean, buildLLT, init_LLT, h=4, n_origins=8)

    plot_rolling_fan(rcv_LL_det$details,  y_clean,
    title = "Rolling CV fan - Local Level (h=4, last 8 origins)")
    plot_rolling_fan(rcv_LLT_det$details, y_clean,
    title = "Rolling CV fan - Local Linear Trend (h=4, last 8 origins)")

    sLL_by_h <- dplyr::summarise(
    dplyr::group_by(rcv_LL_det$details, step),
    RMSE = sqrt(mean((y-mean)^2)),
    MAE  = mean(abs(y-mean)),
    Cov90= mean(y>=lwr & y<=upr),
    AvgWidth = mean(upr-lwr)
    )
    op <- par(mfrow=c(2,2), mar=c(4,4,3,1)); on.exit(par(op), add = TRUE)
    plot(sLL_by_h$step, sLL_by_h$RMSE, type="b", pch=19, xlab="Horizon (steps)", ylab="RMSE",
    main="Rolling CV - RMSE by horizon (LL)")
    plot(sLL_by_h$step, sLL_by_h$MAE, type="b", pch=19, xlab="Horizon", ylab="MAE",
    main="Rolling CV - MAE by horizon (LL)")
    plot(sLL_by_h$step, sLL_by_h$Cov90, type="b", pch=19, xlab="Horizon", ylab="Coverage",
    main="Rolling CV - Coverage by horizon (LL)"); abline(h=.90, lty=3)
    plot(sLL_by_h$step, sLL_by_h$AvgWidth, type="b", pch=19, xlab="Horizon", ylab="Avg PI width",
    main="Rolling CV - PI width by horizon (LL)")

    sLLT_by_h <- dplyr::summarise(
    dplyr::group_by(rcv_LLT_det$details, step),
    RMSE     = sqrt(mean((y - mean)^2, na.rm = TRUE)),
    MAE      = mean(abs(y - mean), na.rm = TRUE),
    Cov90    = mean(y >= lwr & y <= upr, na.rm = TRUE),
    AvgWidth = mean(upr - lwr, na.rm = TRUE)
    )
    op <- par(mfrow = c(2, 2), mar = c(4,4,3,1)); on.exit(par(op), add = TRUE)
    plot(sLLT_by_h$step, sLLT_by_h$RMSE, type = "b", pch = 19, xlab = "Horizon (steps)", ylab = "RMSE",
    main = "Rolling CV - RMSE by horizon (LLT)")
    plot(sLLT_by_h$step, sLLT_by_h$MAE, type = "b", pch = 19, xlab = "Horizon
", ylab = "MAE",
    main = "Rolling CV - MAE by horizon (LLT)")
    plot(sLLT_by_h$step, sLLT_by_h$Cov90, type = "b", pch = 19, xlab = "
Horizon", ylab = "Coverage",
    main = "Rolling CV - Coverage by horizon (LLT)"); abline(h = .90, lty =
3)
    plot(sLLT_by_h$step, sLLT_by_h$AvgWidth, type = "b", pch = 19, xlab = "
Horizon", ylab = "Avg PI width",
    main = "Rolling CV - PI width by horizon (LLT)")
```

Backcast call (single panel) and enhanced rolling fan

Outputs: Figure 3.8 (backcast); enhanced LL fan similar to Figure 3.6.

Listing 39: Call the single-panel backcast for 1950-1959 and draw an enhanced LL rolling fan.

```
      res_back_1950 <- backcast_prefix(y_clean, start_new = 1950, level = 0.90)

      plot_rolling_fan_enhanced(rcv_LL_det$details, y_clean,
      title = "Rolling CV fan - Local Level (h=4, last 8 origins)")
```

ARIMA(0,1,1) equivalence check ; (Appendix 3.7)

Outputs: scalar summaries reported in the text (MA(1), RMSE) and correlation with LL one-step; relates to Appendix discussion.

Listing 40: Empirical equivalence between LL and ARIMA(0,1,1) for one-step forecasts; correlation of fitted one-step means.

```
      fit_ar <- arima(y_clean, order=c(0,1,1))
      cat("\nARIMA(0,1,1) - MA(1) coefficient:\n"); print(fit_ar$coef)
      fitted_ar <- y_clean - residuals(fit_ar)
      rmse_ar <- sqrt(mean((y_clean - fitted_ar)^2, na.rm=TRUE))
      cat("One-step RMSE (ARIMA 0,1,1):", round(rmse_ar,3), "\n")
      fitted_ll <- dropFirst(filt_LL$f)
      y_aligned <- y_clean[-1]
      min_len <- min(length(y_aligned), length(fitted_ll), length(fitted_ar))
      cmp_eq <- data.frame(
      y         = as.numeric(y_aligned)[seq_len(min_len)],
      LL_hat    = as.numeric(fitted_ll)[seq_len(min_len)],
      ARIMA_hat = as.numeric(fitted_ar)[seq_len(min_len)]
      )
      cat("\nLocal Level vs ARIMA(0,1,1) one-step forecast correlation:\n")
      print(cor(cmp_eq$LL_hat, cmp_eq$ARIMA_hat, use="complete.obs"))
```

Parametric bootstrap for (V, W) under LL

Outputs: bootstrap intervals that can complement Table 3.2.

Listing 41: Simple parametric bootstrap for LL variance components; returns bootstrap draws for V and W.

```r
    param_boot_LL <- function(y, B=200){
            est <- fit_mle(y, buildLL, log(c(var(y), 0.1*var(y)))); mod <-
est$mod
            filt <- dlmFilter(y, mod); last <- get_last_state(filt)
            Tn <- length(y); VB <- WB <- numeric(B)
            GG <- as.matrix(mod$GG); FF <- as.matrix(mod$FF)
            W  <- as.numeric(mod$W[1,1]);  V <- as.numeric(mod$V)
            sdW <- sqrt(W); sdV <- sqrt(V)
            for (b in 1:B){
                    m <- as.numeric(last$m)
                    ysim <- numeric(Tn)
                    for (t in 1:Tn){
                            m <- as.numeric(GG %*% m) + rnorm(1, sd=sdW)
                            ysim[t] <- as.numeric(FF %*% m) + rnorm(1, sd=sdV
)
                    }
                    est_b <- fit_mle(ts(ysim, start=start(y), frequency=
frequency(y)), buildLL,
                    log(c(var(ysim), 0.1*var(ysim))))
                    VB[b] <- exp(est_b$fit$par[1]); WB[b] <- exp(est_b$fit$
par[2])
            }
            list(V = VB, W = WB)
    }
    # Example (leave commented unless needed):
    # set.seed(42)
    # boot <- param_boot_LL(y_clean, B=200)
    # print(rbind(V=quantile(boot$V, c(.025,.5,.975)),
    #             W=quantile(boot$W, c(.025,.5,.975))))
```

Modern subsample (from 1995) ; (Section "Model assessment for forecasting")

Outputs: selection table analogous to Table 3.5; short-horizon metrics comparable to Table 3.6.

Listing 42: Modern window analysis from 1995: LL/LLT MLE fits, AIC/BIC, holdout and rolling CV metrics.

```r
    modern_start <- 1995
    y_mod <- window(y_clean, start = modern_start)

    init_LL_mod  <- log(c(var(y_mod), 0.1*var(y_mod)))
    init_LLT_mod <- log(c(var(y_mod), 0.1*var(y_mod), 0.01*var(y_mod)))

    est_LL_mod  <- fit_mle(y_mod,  buildLL,  init_LL_mod)
    est_LLT_mod <- fit_mle(y_mod,  buildLLT, init_LLT_mod)
    logLik_LLm  <- -est_LL_mod$fit$value;  k_LLm  <- length(est_LL_mod$fit$par)
    logLik_LLTm <- -est_LLT_mod$fit$value; k_LLTm <- length(est_LLT_mod$fit$par)
    Tn_mod <- length(y_mod)
    tab_sel_mod <- data.frame(
    Model=c("Local Level","Local Linear Trend"),
    logLik=c(logLik_LLm, logLik_LLTm),
    AIC = c(-2*logLik_LLm  + 2*k_LLm,
    -2*logLik_LLTm + 2*k_LLTm),
    BIC = c(-2*logLik_LLm  + log(Tn_mod)*k_LLm,
    -2*logLik_LLTm + log(Tn_mod)*k_LLTm)
    )
    print(pround(tab_sel_mod))

    h_mod <- min(8L, max(3L, floor(length(y_mod)/6)))
    fc_LL_mod  <- forecast_h(y_mod, buildLL,  init_LL_mod,  h=h_mod, level=0.90)
    fc_LLT_mod <- forecast_h(y_mod, buildLLT, init_LLT_mod, h=h_mod, level=0.90)
    rolling_cv <- function(y, build_fun, init, h=4, n_origins=8, level=0.90){
            n <- length(y)
            origins <- (n - h - n_origins + 1):(n - h)
            out <- lapply(origins, function(i){
                    y_tr <- window(y, end = time(y)[i])
                    y_te <- window(y, start = time(y)[i+1], end = time(y)[i+h])
                    est  <- fit_mle(y_tr, build_fun, init)
                    filt <- dlmFilter(y_tr, est$mod)
                    fc_try <- try(dlmForecast(filt, nAhead = h), silent = TRUE)
                    if (!inherits(fc_try, "try-error")){
                            f  <- as.numeric(fc_try$f); sd <- sqrt(as.numeric(fc_try$Q))
                    } else {
                            last <- get_last_state(filt)
                            man  <- manual_forecast(est$mod, last$m, last$C, h)
                            f <- man$mean; sd <- man$sd
```

```r
                        }
                        ci <- make_ci(f, sd, level)
                        list(mean=f, lwr=ci$lwr, upr=ci$upr, y=as.numeric(y_te))
                })
                pred <- unlist(lapply(out, ‘[[‘, "mean"))
                ytru <- unlist(lapply(out, ‘[[‘, "y"))
                lwr  <- unlist(lapply(out, ‘[[‘, "lwr"))
                upr  <- unlist(lapply(out, ‘[[‘, "upr"))
                data.frame(RMSE = rmse(pred, ytru),
                MAE  = mae(pred, ytru),
                Cov90= mean(ytru >= lwr & ytru <= upr),
                AvgWidth = mean(upr - lwr))
        }
        rcv_LL_mod  <- rolling_cv(y_mod, buildLL,  init_LL_mod,  h=4, n_origins=8, level=0.90)
        rcv_LLT_mod <- rolling_cv(y_mod, buildLLT, init_LLT_mod, h=4, n_origins=8, level=0.90)
        print(pround(rbind(LL=rcv_LL_mod, LLT=rcv_LLT_mod)))
```

Final table builder: MLE, SEs, and SNRs (full sample) ; (Section “Signal–to–Noise Ratio”)

Outputs: Table 3.2 and LLT analogue 3.11.

Listing 43: Build a compact table with MLEs, asymptotic SEs, and SNRs for LL and LLT on the full sample.

```r
        make_mle_se_snr_table <- function(y, digits = 4) {
                if (!exists("est_LL_full", inherits = TRUE)) {
                        init_LL_full  <- log(c(var(y), 0.1 * var(y)))
                        est_LL_full   <- fit_mle(y, buildLL,  init_LL_full)
                } else est_LL_full <- get("est_LL_full")
                if (!exists("est_LLT_full", inherits = TRUE)) {
                        init_LLT_full <- log(c(var(y), 0.1 * var(y), 0.01 * var(y)))
                        est_LLT_full  <- fit_mle(y, buildLLT, init_LLT_full)
                } else est_LLT_full <- get("est_LLT_full")

                summarize_fit <- function(est, model = c("LL","LLT")) {
                        model <- match.arg(model)
                        par_log <- est$fit$par
                        H       <- est$fit$hessian
                        Vc <- try(solve(H), silent = TRUE)
                        if (inherits(Vc, "try-error")) {
                                Vc <- matrix(NA_real_, nrow = length(par_log),
    ncol = length(par_log))
                                warning(sprintf("Hessian inversion failed for %s:
     SEs set to NA.", model))
                        }
                        est_var <- exp(par_log)
                        se_log  <- sqrt(diag(Vc))
                        se_var  <- est_var * se_log

                        if (model == "LL") {
                                names(est_var) <- c("V","W_level"); names(se_var)
      <- c("V","W_level")
                                snr    <- est_var["W_level"] / est_var["V"]
                                var_ln <- Vc[2,2] + Vc[1,1] - 2*Vc[1,2]
                                se_snr <- snr * sqrt(max(0, var_ln))
                                out <- data.frame(
                                Model     = "Local Level",
                                Parameter = c("V","W_level","SNR_level (W/V)"),
                                Estimate  = c(est_var, snr),
                                Std.Error = c(se_var,  se_snr),
                                stringsAsFactors = FALSE
                                )
                        } else {
                                names(est_var) <- c("V","W_level","W_slope")
                                names(se_var)  <- c("V","W_level","W_slope")
                                snr_level <- est_var["W_level"] / est_var["V"]
                                snr_slope <- est_var["W_slope"] / est_var["V"]
                                var_ln_L  <- Vc[2,2] + Vc[1,1] - 2*Vc[1,2]
                                var_ln_B  <- Vc[3,3] + Vc[1,1] - 2*Vc[1,3]
                                se_snr_L  <- snr_level * sqrt(max(0, var_ln_L))
                                se_snr_B  <- snr_slope * sqrt(max(0, var_ln_B))
                                out <- data.frame(
                                Model     = "Local Linear Trend",
                                Parameter = c("V","W_level","W_slope",
                                "SNR_level (W_level/V)","SNR_slope (W_slope/V)"),
                                Estimate  = c(est_var, snr_level, snr_slope),
                                Std.Error = c(se_var,  se_snr_L,  se_snr_B),
                                stringsAsFactors = FALSE
                                )
                        }
                        out
                }

                tab <- rbind(
                summarize_fit(est_LL_full,  "LL"),
                summarize_fit(est_LLT_full, "LLT")
```

```r
            )
            num_cols <- vapply(tab, is.numeric, logical(1))
            tab[num_cols] <- lapply(tab[num_cols], function(x) round(x,
    digits))
            rownames(tab) <- NULL
            tab
    }
    tab_mle <- make_mle_se_snr_table(y_clean)
    print(tab_mle)
```

Bayesian approach

This appendix collects the R code used in Chapter 3. Each listing is self-contained and reproducible.

Project setup: packages, seed, and data ; (Section 3.4.2)

Outputs: none; used for Figures 3.9, 3.10, 3.11 and tables 3.7, 3.8, 3.9.

Listing 44: Packages, seed, and data loading (column 'GDP' from sheet 'Foglio1'); foundation for all subsequent figures and tables.

```r
    suppressPackageStartupMessages({
            library(dlm)
            library(coda)
            library(dplyr)
            library(readxl)
    })
    set.seed(123); graphics.off()

    # Load data once: annual Italian real GDP growth, 1960-2024
    if (!exists("y_all", inherits = TRUE) || !exists("y_clean", inherits =
    TRUE)) {
            raw <- read_excel("macro_italia_missingdata.xlsx", sheet = "
    Foglio1")
            y_all   <- ts(as.numeric(raw$GDP), start = 1960, frequency = 1)
            y_clean <- stats::na.omit(y_all)
    }
```

Utility helpers ; (Bayesian tables/figures)

Outputs: posterior table 3.7; residual plots 3.11.

Listing 45: Minimal helpers for rounding, accuracy metrics, posterior summaries, ACF/PACF plotting, and a posterior table builder.

```r
    pround <- function(df, k = 3) dplyr::mutate(df, dplyr::across(where(is.numeric), ~round(.x, k)))
    rmse <- function(a,b) sqrt(mean((a-b)^2, na.rm = TRUE))
    mae  <- function(a,b) mean(abs(a-b),  na.rm = TRUE)

    post_summary <- function(x, probs = c(.025,.5,.975)){
            c(mean = mean(x), sd = sd(x),
            setNames(as.numeric(quantile(x, probs)), paste0("q", probs*100)))
    }

    acf_with_red <- function(x, main, lag.max = 40, ci = 0.95, ci.col = "gray40"){
            x <- x[is.finite(x)]
            a  <- acf(x, lag.max = lag.max, plot = FALSE)
            lg <- as.numeric(drop(a$lag)); ac <- as.numeric(drop(a$acf))
            n  <- length(x); ci_lim <- stats::qnorm((1+ci)/2)/sqrt(n)
            plot(NA, xlim = range(lg), ylim = range(c(ac, 0, -ci_lim, ci_lim)),
            xlab = "Lag", ylab = "Autocorrelation", main = main, frame.plot = TRUE)
            abline(h = 0); abline(h = c(-ci_lim, ci_lim), lty = 2, col = ci.col)
            segments(lg, 0, lg, ac); points(lg, ac, col = "red", pch = 19)
    }

    pacf_with_red <- function(x, main, lag.max = 40, ci = 0.95, ci.col = "gray40"){
            x <- x[is.finite(x)]
            p  <- pacf(x, lag.max = lag.max, plot = FALSE)
            lg <- seq_along(drop(p$acf)); pc <- as.numeric(drop(p$acf))
            n  <- length(x); ci_lim <- stats::qnorm((1+ci)/2)/sqrt(n)
            plot(NA, xlim = c(1, max(lg)), ylim = range(c(pc, 0, -ci_lim, ci_lim)),
            xlab = "Lag", ylab = "Partial autocorrelation", main = main, frame.plot = TRUE)
            abline(h = 0); abline(h = c(-ci_lim, ci_lim), lty = 2, col = ci.col)
            segments(lg, 0, lg, pc); points(lg, pc, col = "red", pch = 19)
    }

    make_posterior_table <- function(V_draw, W_draw_mat, model=c("LL","LLT"), digits=3){
            model <- match.arg(model)
            if (model=="LL"){
```

```
                        Wl <- as.numeric(W_draw_mat); SNR <- Wl / V_draw
                        tab <- rbind(V = post_summary(V_draw),
                        Wl = post_summary(Wl),
                        Wl_over_V = post_summary(SNR))
                        out <- data.frame(Model="Local Level",
                        Parameter=c("V","W_level","SNR_level (W/V)"),
                        round(tab, digits), row.names = NULL)
                } else {
                        Wl <- as.numeric(W_draw_mat[,1]); Wb <- as.numeric(W_draw
    _mat[,2])
                        tab <- rbind(V = post_summary(V_draw),
                        Wl = post_summary(Wl),
                        Wb = post_summary(Wb))
                        out <- data.frame(Model="Local Linear Trend",
                        Parameter=c("V","W_level","W_slope"),
                        round(tab, digits), row.names=NULL)
                }
                out
        }
```

Forecast utilities (posterior predictive) ; (Sections 3.4.2, 3.3)

Outputs: Figures 3.12, 3.13; Tables 3.8, 3.9.

Listing 46: Posterior predictive simulator from a single Gibbs draw (LL or LLT) and a holdout forecasting wrapper returning means and 90% bands.

```
        # Posterior predictive (h-step) from a Gibbs draw (LL or LLT)
        sim_ppc_from_draw <- function(order, theta0, V, Wvec, h){
                stopifnot(order %in% c(1,2))
                y <- numeric(h)
                if (order == 1){
                        mu <- theta0[1]; sV <- sqrt(V); sW <- sqrt(Wvec[1])
                        for (k in 1:h){ mu <- mu + rnorm(1, sd = sW); y[k] <- mu
    + rnorm(1, sd = sV) }
                } else {
                        lev <- theta0[1]; slope <- theta0[2]
                        sV <- sqrt(V); sW1 <- sqrt(Wvec[1]); sW2 <- sqrt(Wvec[2])
                        for (k in 1:h){
                                lev   <- lev   + slope + rnorm(1, sd = sW1)
                                slope <- slope + rnorm(1, sd = sW2)
                                y[k]  <- lev + rnorm(1, sd = sV)
                        }
                }
                y
        }
```

```r

    bayes_forecast_holdout <- function(y_train, order, gibbs, burn=2000, thin=5, h=8, level=.90){
            keep   <- seq(burn + 1L, length.out = floor((length(gibbs$dV) - burn)/thin), by = thin)
            Vs     <- as.numeric(gibbs$dV[keep])
            dW     <- gibbs$dW
            Wdraws <- if (is.matrix(dW)) dW[keep, , drop = FALSE] else matrix(dW[keep], ncol = 1)
            th     <- gibbs$theta
            theta_T <- th[dim(th)[1], 1:order, keep, drop = FALSE]
            S <- dim(theta_T)[3]
            Ysim <- matrix(NA_real_, nrow = S, ncol = h)
            for (s in 1:S){
                    Ysim[s, ] <- sim_ppc_from_draw(order,
                    theta0 = as.numeric(theta_T[1, , s]),
                    V = Vs[s],
                    Wvec = as.numeric(Wdraws[s, ]),
                    h = h)
            }
            a <- (1 - level)/2
            list(mean = apply(Ysim, 2, mean),
            lwr  = apply(Ysim, 2, quantile, probs = a),
            upr  = apply(Ysim, 2, quantile, probs = 1-a),
            Ysim = Ysim)
    }

    # Plot a holdout forecast against the last history window
    plot_bayes_fc <- function(y_all, y_te, mean_h, lwr_h, upr_h, title){
            fc_start <- time(y_all)[length(y_all) - length(y_te) + 1]
            mean_ts <- ts(mean_h, start = fc_start, frequency = frequency(y_all))
            lwr_ts  <- ts(lwr_h,  start = fc_start, frequency = frequency(y_all))
            upr_ts  <- ts(upr_h,  start = fc_start, frequency = frequency(y_all))
            hist_idx <- max(1, length(y_all) - 2*length(y_te))
            y_hist <- window(y_all, start = time(y_all)[hist_idx])
            plot(y_hist, type="l", col="black", xlab="Year", ylab="GDP", main = title)
            lines(mean_ts, col="purple", lwd=2)
            lines(lwr_ts,  col="orange", lty=2)
            lines(upr_ts,  col="orange", lty=2)
            points(time(window(y_all, start = start(mean_ts), end = end(mean_ts))),
            as.numeric(y_te), pch=19)
            legend("topleft", bty="n",
            legend=c("Observed","Posterior mean","90% predictive interval"),
            col=c("black","purple","orange"), lty=c(1,1,2), lwd=c(1,2,1))
    }
```

Rolling-origin utilities: fan chart and cross-validation paths (Sec. 3.3, Sec. 3.4.2)

Outputs: Figure 3.13; Table 3.9; Appendix: Figure 3.21 (per-horizon profiles), Figure 3.22 (LL vs LLT aggregate).

Listing 47: Fan-chart painter and rolling-origin cross-validation generator used for Figure 3.12 and Tables 3.9/A.7, plus the LL vs LLT barplot in Appendix A.8.

```
  plot_rolling_fan <- function(df, y, title){
      rng <- range(df$time)
      y_seg <- window(y, start = rng[1], end = rng[2])
      plot(y_seg, type="l", col="black", lwd=1.2, xlab="Year", ylab="GDP", main=title)
      for (o in unique(df$origin_end)){
          sub <- df[df$origin_end == o, ]
          segments(sub$time, sub$lwr, sub$time, sub$upr, col=rgb(1,0.5,0,0.45))
          points(sub$time, sub$mean, pch=16, cex=.8, col="steelblue")
      }
      legend("topleft", bty="n",
      legend=c("Observed","Forecast mean","90% predictive interval"),
      col=c("black","steelblue",rgb(1,0.5,0,1)), lty=c(1,NA,1), pch=c(NA,16,NA))
  }

  bayes_rolling_cv_paths <- function(y, order=1L, n_iter=12000, burn=2000, thin=5,
  h=4, n_origins=8, level=.90, seed=123){
      set.seed(seed)
      vy <- var(y); n <- length(y)
      origins <- (n - h - n_origins + 1):(n - h)
      mod0_LL  <- dlmModPoly(1, dV=1, dW=1)
      mod0_LLT <- dlmModPoly(2, dV=1, dW=c(1,1))
      paths <- list(); acc <- list()
      for (i in origins){
          y_tr <- window(y, end = time(y)[i])
          y_te <- window(y, start = time(y)[i+1], end = time(y)[i+h])
```

```
                    if (order==1L){
                            g <- dlmGibbsDIG(y_tr, mod0_LL, a.y=2,b.y=(2-1)*vy, a.theta=2, b.theta=(2-1)*vy/10, n.sample=n_iter)
                            fc <- bayes_forecast_holdout(y_tr, order=1, gibbs=g, burn=burn, thin=thin, h=h, level=level)
                    } else {
                            g <- dlmGibbsDIG(y_tr, mod0_LLT, a.y=2,b.y=(2-1)*vy, a.theta=c(2,2),b.theta=c(vy/5,vy/20), n.sample=n_iter)
                            fc <- bayes_forecast_holdout(y_tr, order=2, gibbs=g, burn=burn, thin=thin, h=h, level=level)
                    }
                    tm <- time(y)[(i+1):(i+h)]
                    paths[[length(paths)+1]] <- data.frame(time=tm, mean=fc$mean, lwr=fc$lwr, upr=fc$upr,
                    origin_end=time(y)[i])
                    acc[[length(acc)+1]] <- data.frame(mean=fc$mean, lwr=fc$lwr, upr=fc$upr, y=as.numeric(y_te))
            }
            P <- dplyr::bind_rows(paths); A <- dplyr::bind_rows(acc)
            rmse_all <- sqrt(mean((A$mean - A$y)^2))
            mae_all  <- mean(abs(A$mean - A$y))
            cov_all  <- mean(A$y >= A$lwr & A$y <= A$upr)
            w_all    <- mean(A$upr - A$lwr)
            by_h <- do.call(rbind, lapply(seq_len(h), function(s){
                    idx <- seq(from = s, to = nrow(A), by = h)
                    data.frame(step=s,
                    RMSE = sqrt(mean((A$mean[idx]-A$y[idx])^2)),
                    MAE  = mean(abs(A$mean[idx]-A$y[idx])),
                    Cov90= mean(A$y[idx] >= A$lwr[idx] & A$y[idx] <= A$upr[idx]),
                    AvgWidth = mean(A$upr[idx] - A$lwr[idx]))
            }))
            list(paths_df = P,
            aggregate = data.frame(RMSE=rmse_all, MAE=mae_all, Cov90=cov_all, AvgWidth=w_all),
            by_h = by_h)
    }
```

Bayesian Local Level: posterior and diagnostics; (Section 3.4.2)

Outputs: Figures 3.9, 3.10, 3.11; Table 3.7.

Listing 48: Bayesian LL posterior and diagnostics; produces Table 3.8 (LL content) and Figures 3.8–3.10.

```
    n_iter <- 12000; burn <- 2000; thin <- 5; vy <- var(y_clean)

    # LL prior skeleton and Gibbs sampling
    mod0_LL <- dlmModPoly(order = 1, dV = 1, dW = 1)
    gibbs_LL <- dlmGibbsDIG(y_clean, mod0_LL,
    a.y=2, b.y=(2-1)*vy,
    a.theta=2, b.theta=(2-1)*vy/10,
    n.sample = n_iter)

    keep_LL <- seq(burn+1, n_iter, by = thin)
    V_s_LL  <- as.numeric(gibbs_LL$dV[keep_LL])
    W_s_LL  <- if (is.matrix(gibbs_LL$dW)) as.numeric(gibbs_LL$dW[keep_LL,1]) else as.numeric(gibbs_LL$dW[keep_LL])

    # Table 3.8 (LL section): posterior summaries of V, W_level, and SNR (W/V)
    tab_post_LL <- make_posterior_table(V_s_LL, matrix(W_s_LL, ncol=1), "LL")
    print(pround(tab_post_LL, 3))  # example: V ~ 6.09; W ~ 0.44; SNR ~ 0.075

    # Figure 3.8 -- trace and ACF of variances
    op <- par(mfrow = c(2,2), mar = c(4,4,2,1))
    plot(V_s_LL, type = "l", main = "Trace of observation variance", ylab = "")
    acf_with_red(V_s_LL, "Autocorrelation function of observation variance")
    plot(W_s_LL, type = "l", main = "Trace of level variance", ylab = "")
    acf_with_red(W_s_LL, "Autocorrelation function of level variance")
    par(op)

    # Figure 3.9 -- posterior level: mean and 90% credible interval
    theta_LL <- gibbs_LL$theta[, 1, keep_LL, drop = FALSE]
    mu_mean  <- apply(theta_LL, 1, mean)
    mu_q05   <- apply(theta_LL, 1, quantile, .05)
    mu_q95   <- apply(theta_LL, 1, quantile, .95)
    op <- par(mar = c(6,4,3,2))
    ts.plot(y_clean,
    ts(mu_mean, start = start(y_clean), frequency = frequency(y_clean)),
    col = c("black","purple"), lty = c(1,2), lwd = c(1.2,1.6),
    ylab = "GDP / Level", xlab = "Year",
    main = "Bayesian Local Level: posterior mean of level (+ 90% credible)")
    lines(ts(mu_q05, start=start(y_clean), frequency=frequency(y_clean)), col="indianred3", lty=3)
    lines(ts(mu_q95, start=start(y_clean), frequency=frequency(y_clean)), col="indianred3", lty=3)
    legend("bottom", inset=c(0,-0.25), xpd=TRUE, bty="n", horiz=TRUE,
    legend=c("Observed GDP","Posterior mean of level","90% credible interval"
),
    col=c("black","purple","indianred3"), lty=c(1,2,3))
    par(op)

    # Figure 3.10 -- ACF/PACF of standardized residuals (plug-in at posterior
 means)
    mod_LL_plugin <- dlmModPoly(order = 1, dV = mean(V_s_LL), dW = mean(W_s_
LL))
    filt_bLL <- dlmFilter(y_clean, mod_LL_plugin)
    res_obj  <- residuals(filt_bLL, sd = TRUE)
    zres <- as.numeric(res_obj$res) / as.numeric(res_obj$sd)
    zres <- zres[is.finite(zres)]
    op <- par(mfrow=c(1,2))
    acf_with_red(zres, "Autocorrelation function of standardized residuals")
    pacf_with_red(zres, "Partial autocorrelation function of standardized
residuals")
    par(op)
```

Bayesian Local Linear Trend: posterior level; (Appendix 3.8)

Outputs: Figure 3.23.

Listing 49: Bayesian LLT posterior level for a full-sample visual comparison; used in the appendix.

```
    mod0_LLT <- dlmModPoly(order = 2, dV = 1, dW = c(1,1))
    gibbs_LLT <- dlmGibbsDIG(y_clean, mod0_LLT,
    a.y=2, b.y=(2-1)*vy,
    a.theta=c(2,2), b.theta=c(vy/5, vy/20),
    n.sample = n_iter)
    keep_LLT <- seq(burn+1, n_iter, by = thin)
    theta_LLT <- gibbs_LLT$theta[, 1:2, keep_LLT, drop = FALSE]
    mu_mean_LLT <- apply(theta_LLT[,1, , drop = FALSE], 1, mean)
    mu_q05_LLT  <- apply(theta_LLT[,1, , drop = FALSE], 1, quantile, .05)
    mu_q95_LLT  <- apply(theta_LLT[,1, , drop = FALSE], 1, quantile, .95)
    op <- par(mar = c(6,4,3,2))
    ts.plot(y_clean,
    ts(mu_mean_LLT, start = start(y_clean), frequency = frequency(y_clean)),
    col = c("black","purple"), lty = c(1,2), lwd = c(1.2,1.6),
    ylab = "GDP / Level", xlab = "Year",
    main = "Bayesian Local Linear Trend: posterior mean of level (+ 90%
credible)")
    lines(ts(mu_q05_LLT, start=start(y_clean), frequency=frequency(y_clean)),
 col="indianred3", lty=3)
    lines(ts(mu_q95_LLT, start=start(y_clean), frequency=frequency(y_clean)),
 col="indianred3", lty=3)
    legend("bottom", inset=c(0,-0.25), xpd=TRUE, bty="n", horiz=TRUE,
```

```r
    legend=c("Observed GDP","Posterior mean of level","90% credible interval"),
    col=c("black","purple","indianred3"), lty=c(1,2,3))
    par(op)
```

Fixed holdout experiment ($h = 8$); (Section 3.4.2)

Outputs: Figure 3.12; Tables 3.8, 3.16.

Listing 50: Eight-step holdout: Bayesian LL (main-text Figure 3.11, Table 3.8) and Bayesian LLT (appendix figure and Table A.3).

```r
    h <- 8L
    n_tot  <- length(y_clean)
    y_train <- window(y_clean, end = time(y_clean)[n_tot - h])
    y_test  <- window(y_clean, start = time(y_clean)[n_tot - h + 1])

    # LL on the training sample
    g_LL_tr <- dlmGibbsDIG(y_train, mod0_LL,
    a.y=2,b.y=(2-1)*vy, a.theta=2, b.theta=(2-1)*vy/10,
    n.sample=n_iter)
    fc_LL <- bayes_forecast_holdout(y_train, order=1, gibbs=g_LL_tr, burn=burn, thin=thin, h=h, level=.90)

    # LLT on the training sample (appendix)
    g_LLT_tr <- dlmGibbsDIG(y_train, mod0_LLT,
    a.y=2,b.y=(2-1)*vy, a.theta=c(2,2), b.theta=c(vy/5,vy/20),
    n.sample=n_iter)
    fc_LLT <- bayes_forecast_holdout(y_train, order=2, gibbs=g_LLT_tr, burn=burn, thin=thin, h=h, level=.90)

    # Summary tables: Table 3.8 (LL) and Table A.3 (LLT)
    tab_holdout <- data.frame(
    Model    = c("Bayesian Local Level","Bayesian Local Linear Trend"),
    RMSE     = c(rmse(fc_LL$mean, as.numeric(y_test)), rmse(fc_LLT$mean, as.numeric(y_test))),
    MAE      = c(mae (fc_LL$mean, as.numeric(y_test)), mae (fc_LLT$mean, as.numeric(y_test))),
    Coverage = c(mean(as.numeric(y_test) >= fc_LL$lwr  & as.numeric(y_test) <= fc_LL$upr),
    mean(as.numeric(y_test) >= fc_LLT$lwr & as.numeric(y_test) <= fc_LLT$upr)),
    AvgWidth = c(mean(fc_LL$upr - fc_LL$lwr), mean(fc_LLT$upr - fc_LLT$lwr))
    )
    print(pround(tab_holdout, 3))

```

```r
        # Figures: LL in main text, LLT in appendix
        plot_bayes_fc(y_clean, y_test, fc_LL$mean,  fc_LL$lwr,  fc_LL$upr,
        "Bayesian Local Level - h-step forecast (holdout)")
        plot_bayes_fc(y_clean, y_test, fc_LLT$mean, fc_LLT$lwr, fc_LLT$upr,
        "Bayesian Local Linear Trend - h-step forecast (holdout)")
```

Rolling-origin evaluation ($h = 4$); Sections3.4.2, 3.3

Outputs: Figures 3.13; Tables 3.9.

Listing 51: Rolling-origin design with h = 4: fan chart and metrics for Bayesian LL (Figure 3.12 and Table 3.9) plus LL vs LLT aggregate barplot (Appendix A.8) and per-horizon profiles (Appendix A.7).

```r
        roll_LL <- bayes_rolling_cv_paths(y_clean, order=1L, h=4, n_origins=8,
        n_iter=n_iter, burn=burn, thin=thin, level=.90)
        plot_rolling_fan(roll_LL$paths_df, y_clean,
        "Rolling-origin fan chart (h = 4): Bayesian Local Level")
        print(pround(roll_LL$aggregate, 3))  # Table 3.9
        print(pround(roll_LL$by_h, 3))       # Appendix A.7

        # Appendix A.8 -- aggregate comparison LL vs LLT (barplot)
        roll_LLT <- bayes_rolling_cv_paths(y_clean, order=2L, h=4, n_origins=8,
        n_iter=n_iter, burn=burn, thin=thin, level=.90)
        agg_mat <- rbind(LL  = unlist(roll_LL$aggregate),
        LLT = unlist(roll_LLT$aggregate))[, c("RMSE","MAE","Cov90","AvgWidth")]
        barplot(t(agg_mat), beside = TRUE, legend = TRUE, args.legend = list(bty=
    "n"),
        main = "Aggregate rolling CV metrics (LL vs LLT)")
```

Backcast experiment: 1950–1959 under LL; (Section 3.4.2)

Outputs: Figure 3.14.

Listing 52: Backcast for 1950-1959 from LL Gibbs output; produces Figure 3.13 with posterior mean and 90% predictive bands.

```r
        bayes_backcast_from_gibbs <- function(y, start_new = 1950,
        gibbs, order = 1L,
        keep = NULL, S = 1000,
        level = 0.90, seed = 123) {
                stopifnot(is.ts(y), order %in% c(1,2))
                f <- frequency(y)
                add <- start(y)[1] - start_new
                stopifnot(add >= 1)

                y_ext <- ts(c(rep(NA_real_, add * f), as.numeric(y)),
                start = start_new, frequency = f)
                Tstar <- length(y_ext)

                if (is.null(keep)) keep <- seq_len(length(gibbs$dV))
                keep <- keep[keep >= 1 & keep <= length(gibbs$dV)]
                S <- min(S, length(keep))
                set.seed(seed)
                idx <- sample(keep, S, replace = FALSE)

                Vdraw <- as.numeric(gibbs$dV[idx])
                if (is.matrix(gibbs$dW)) {
                        W1draw <- as.numeric(gibbs$dW[idx, 1])
                        W2draw <- if (order == 2) as.numeric(gibbs$dW[idx, 2]) else NULL
                } else {
                        W1draw <- as.numeric(gibbs$dW[idx]); W2draw <- NULL
                }

                mu_mat <- matrix(NA_real_, nrow = S, ncol = Tstar)
                ypred_mat <- matrix(NA_real_, nrow = S, ncol = Tstar)

                for (s in seq_len(S)){
                        if (order == 1){
                                mod_s <- dlmModPoly(order = 1, dV = Vdraw[s], dW = W1draw[s])
                        } else {
                                mod_s <- dlmModPoly(order = 2, dV = Vdraw[s], dW = c(W1draw[s], W2draw[s]))
                        }
                        filt_s <- dlmFilter(y_ext, mod_s)
                        smth_s <- dlmSmooth(filt_s)

                        mu_s <- if (order == 1) as.numeric(smth_s$s)[-1] else as.numeric(smth_s$s[-1,1])
                        mu_mat[s, ]    <- mu_s
                        ypred_mat[s, ] <- mu_s + rnorm(Tstar, sd = sqrt(Vdraw[s]))
                }

                mu_mean <- colMeans(mu_mat)
                a <- (1 - level)/2
                lwr <- apply(ypred_mat, 2, quantile, probs = a)
                upr <- apply(ypred_mat, 2, quantile, probs = 1 - a)

```

```
            mu_ts  <- ts(mu_mean, start = start(y_ext), frequency = f)
            lwr_ts <- ts(lwr,     start = start(y_ext), frequency = f)
            upr_ts <- ts(upr,     start = start(y_ext), frequency = f)

            op <- par(mar = c(6.5,4,3,2))
            ts.plot(y_ext, mu_ts,
            col = c("gray20","purple"), lty = c(1,2), lwd = c(1.2,1.8),
            xlab = "Year", ylab = "GDP",
            main = "Bayesian Local Level: backcast 1950-1959 (posterior draws
)")
            lines(lwr_ts, col = "orange", lty = 2)
            lines(upr_ts, col = "orange", lty = 2)
            rect(start_new, par("usr")[3], start(y)[1]-1, par("usr")[4],
            col = rgb(0,0,1,0.06), border = NA)
            lines(window(y_ext, start = start(y)[1]), col = "black", lwd =
1.8)
            legend("bottom", inset = c(0, -0.25), xpd = NA, bty = "n", horiz
= TRUE,
            legend = c("Observed (1960-2024)",
            "Missing span (1950-59)",
            "Posterior mean",
            "90% predictive interval"),
            col = c("black", rgb(0,0,1,0.20), "purple", "orange"),
            lty = c(1, NA, 2, 2),
            lwd = c(1.8, NA, 1.8, 1),
            pch = c(NA, 15, NA, NA))
            par(op)

            invisible(list(mu = mu_ts, lwr = lwr_ts, upr = upr_ts))
    }

    back_LL <- bayes_backcast_from_gibbs(y_clean, start_new = 1950,
    gibbs = gibbs_LL, order = 1L,
    keep = keep_LL, S = 1000, level = .90)
```

Bibliography

[1] Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, **19**(6), 716–723.

[2] Andrieu, C., Doucet, A., & Holenstein, R. (2010). Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **72**(3), 269–342.

[3] Bayes, T. (1763). An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society of London*, **53**, 370–418.

[4] Box, G. E. P., & Jenkins, G. M. (1970). *Time Series Analysis: Forecasting and Control*. San Francisco, CA: Holden–Day.

[5] Box, G. E. P., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). *Time Series Analysis: Forecasting and Control* (5th ed.). Hoboken, NJ: John Wiley & Sons.

[6] Carter, C. K., & Kohn, R. (1994). On Gibbs sampling for state space models. *Biometrika*, **81**(3), 541–553.

[7] de Finetti, B. (2017). *Theory of Probability: A Critical Introductory Treatment* (Vols. 1–2, combined ed.; A. Machí & A. F. M. Smith, Trans.). Chichester, UK & Hoboken, NJ: Wiley. (Orig. ed. 1974.)

[8] Dickey, D. A., & Fuller, W. A. (1979). Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American Statistical Association*, **74**(366), 427–431.

[9] Durbin, J., & Koopman, S. J. (2001). *Time Series Analysis by State Space Methods*. Oxford: Oxford University Press.

[10] Durbin, J., & Koopman, S. J. (2012). *Time Series Analysis by State Space Methods* (2nd ed.). Oxford: Oxford University Press.

[11] Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., & Rubin, D. B. (2013). *Bayesian Data Analysis* (3rd ed.). Boca Raton, FL: CRC Press.

[12] Gelfand, A. E., & Smith, A. F. M. (1990). Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association*, **85**(410), 398–409.

[13] Gordon, N. J., Salmond, D. J., & Smith, A. F. M. (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings F—Radar and Signal Processing*, **140**(2), 107–113.

[14] Hamilton, J. D. (1989). A new approach to the economic analysis of non-stationary time series and the business cycle. *Econometrica*, **57**(2), 357–384.

[15] Hamilton, J. D. (1994). *Time Series Analysis*. Princeton, NJ: Princeton University Press.

[16] Harrison, P. J., & Stevens, C. F. (1976). Bayesian forecasting. *Journal of the Royal Statistical Society: Series B (Methodological)*, **38**(3), 205–228.

[17] Harvey, A. C. (1989). *Forecasting, Structural Time Series Models and the Kalman Filter*. Cambridge: Cambridge University Press.

[18] Hoffman, M. D., & Gelman, A. (2014). The No-U-Turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, **15**, 1593–1623.

[19] Hyndman, R. J., & Athanasopoulos, G. (2021). *Forecasting: Principles and Practice* (3rd ed.). Melbourne: OTexts. URL: `https://otexts.com/fpp3/` (Accessed 13 September 2025).

[20] Hurn, S., Martin, V. L., Phillips, P. C. B., & Yu, J. (2020). *Financial Econometric Modeling*. New York, NY: Oxford University Press.

[21] Ionides, E. L., Bretó, C., & King, A. A. (2011). Iterated filtering. *The Annals of Statistics*, **39**(3), 1776–1802.

[22] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). *An Introduction to Statistical Learning with Applications in R* (2nd ed.). New York, NY: Springer.

[23] Jarque, C. M., & Bera, A. K. (1980). Efficient tests for normality, homoscedasticity and serial independence of regression residuals. *Economics Letters*, **6**(3), 255–259.

[24] Jarque, C. M., & Bera, A. K. (1987). A test for normality of observations and regression residuals. *International Statistical Review*, **55**(2), 163–172.

[25] Julier, S. J., & Uhlmann, J. K. (2004). Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, **92**(3), 401–422.

[26] Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME—Journal of Basic Engineering*, **82**(1), 35–45.

[27] Kim, C.-J. (1994). Dynamic linear models with Markov-switching. *Journal of Econometrics*, **60**(1–2), 1–22.

[28] Kwiatkowski, D., Phillips, P. C. B., Schmidt, P., & Shin, Y. (1992). Testing the null hypothesis of stationarity against the alternative of a unit root. *Journal of Econometrics*, **54**(1–3), 159–178.

[29] Laplace, P.-S. (1812; 3rd ed. 1820). *Théorie Analytique des Probabilités*. Paris: Courcier. (*English translations include*: Dale, A. I. (1995), *Philosophical Essay on Probabilities*, Springer; Pulskamp, R. J. (2021), partial translation of Books I–II, online.)

[30] Lavine, M. (2019). Frequentist, Bayes, or other? *The American Statistician*, **73**, 312–318.

[31] Ljung, G. M., & Box, G. E. P. (1978). On a measure of lack of fit in time series models. *Biometrika*, **65**(2), 297–303.

[32] MathWorks (2025). Autocorrelation and Partial Autocorrelation. *Econometrics Toolbox Documentation.* Natick, MA: The MathWorks, Inc. URL: `https://www.mathworks.com/help/econ/autocorrelation-and-partial-autocorrelation.html` (Accessed 13 September 2025).

[33] Migon, H. S., Gamerman, D., Lopes, H. F., & Ferreira, M. A. R. (2005). Dynamic models. In D. K. Dey & C. R. Rao (Eds.), *Handbook of Statistics*, **25**, 553–588. Amsterdam: Elsevier.

[34] Newey, W. K., & West, K. D. (1987). A simple, positive semi-definite, heteroskedasticity and autocorrelation consistent covariance matrix. *Econometrica*, **55**(3), 703–708.

[35] Nocedal, J., & Wright, S. J. (2006). *Numerical Optimization* (2nd ed.). New York, NY: Springer.

[36] Petris, G., Petrone, S., & Campagnoli, P. (2009). *Dynamic Linear Models with R*. New York, NY: Springer.

[37] Petris, G. (2010). An R package for dynamic linear models. *Journal of Statistical Software*, **36**(12), 1–16.

[38] Phillips, P. C. B., & Perron, P. (1988). Testing for a unit root in time series regression. *Biometrika*, **75**(2), 335–346.

[39] R Core Team (2025). *R: A Language and Environment for Statistical Computing*. Vienna: R Foundation for Statistical Computing. URL: `https://www.R-project.org/` (Accessed 13 September 2025).

[40] Rauch, H. E., Tung, F., & Striebel, C. T. (1965). Maximum likelihood estimates of linear dynamic systems. *AIAA Journal*, **3**(8), 1445–1450.

[41] Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics*, **6**(2), 461–464.

[42] Shapiro, S. S., & Wilk, M. B. (1965). An analysis of variance test for normality (complete samples). *Biometrika*, **52**(3/4), 591–611.

[43] Slutsky, E. (1937). The summation of random causes as the source of cyclic processes. *Econometrica*, **5**(2), 105–146.

[44] Spiegelhalter, D. J., Best, N. G., Carlin, B. P., & Van der Linde, A. (2002). Bayesian measures of model complexity and fit. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **64**(4), 583–639.

[45] Tashman, L. J. (2000). Out-of-sample tests of forecasting accuracy: An analysis and review. *International Journal of Forecasting*, **16**(4), 437–450.

[46] Vehtari, A., Gelman, A., & Gabry, J. (2017). Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC. *Statistics and Computing*, **27**, 1413–1432.

[47] Watanabe, S. (2010). Asymptotic equivalence of Bayes cross validation and widely applicable information criterion (WAIC). *Journal of Machine Learning Research*, **11**, 3571–3594.

[48] West, M., Harrison, P. J., & Migon, H. S. (1985). Dynamic generalized linear models and Bayesian forecasting. *Journal of the American Statistical Association*, **80**(389), 73–97.

[49] West, M., & Harrison, J. (1997). *Bayesian Forecasting and Dynamic Models* (2nd ed.). New York, NY: Springer.

[50] World Bank (2025). *World Development Indicators—NY.GDP.MKTP.KD.ZG (GDP growth, annual%)*. Washington, DC: The World Bank. URL: `https://data.worldbank.org/indicator/NY.GDP.MKTP.KD.ZG` (Accessed 13 September 2025).

[51] Wold, H. (1938). *A Study in the Analysis of Stationary Time Series*. Uppsala: Almqvist & Wiksell.

[52] Yule, G. U. (1927). On a method of investigating periodicities in disturbed series, with special reference to Wolfer's sunspot numbers. *Philosophical Transactions of the Royal Society of London. Series A*, **226**, 267–298.

[53] Petris, G. (2025). *dlm: Bayesian and Likelihood Analysis of Dynamic Linear Models*. R package. CRAN, R Project. URL: `https://CRAN.R-project.org/package=dlm` (Accessed 13 September 2025).

[54] Hyndman, R. J., *et al.* (2025). *forecast: Forecasting Functions for Time Series and Linear Models*. R package. CRAN, R Project. URL: `https://CRAN.R-project.org/package=forecast` (Accessed 13 September 2025).