

UNIVERSITY OF TARTU
FACULTY OF SCIENCE AND TECHNOLOGY
INSTITUTE OF MATHEMATICS AND STATISTICS

Zhexin Rao

**Application of Hidden Markov Models
in Financial Time Series Analysis**

Actuarial and Financial Engineering

Master's Thesis (30 ECTS)

Supervisor: PhD Kristi Kuljus

TARTU 2026

APPLICATION OF HIDDEN MARKOV MODELS IN FINANCIAL TIME SERIES ANALYSIS

Master thesis

Zhexin Rao

Abstract

This thesis studies the use of hidden Markov models (HMMs) for market-regime identification in financial time series, with particular attention to the role of decoding in interpreting fitted regimes. The analysis considers three decoders: the Viterbi path, the pointwise maximum a posteriori (PMAP) path, and a hybrid decoder that balances global path coherence against pointwise classification accuracy. The methodological contribution is to evaluate how these decoders behave in segmentation problems and to show that a small positive hybrid weight can serve as a conservative PMAP-leaning choice when fitted transition matrices contain zero or near-zero entries. A simulation study investigates the effects of state dependence, emission overlap, and the hybrid weight parameter on decoding quality. The results show that decoding becomes substantially more difficult when dependence is weak and emission distributions overlap strongly. PMAP generally achieves slightly lower classification error than Viterbi, while small positive hybrid weights remain close to PMAP in accuracy and modestly reduce incoherent decoded transitions. The empirical application fits a three-state HMM with Gaussian emissions to weekly and fixed-interval monthly CSI 300 returns. At the weekly frequency, heavy overlap between the estimated regimes and implausible transition dynamics make the classification economically weak. The monthly model is more interpretable, especially in the earlier sample, but after 2016 it assigns most observations to the sidewalk state, making the regime map too coarse. An expanding-window monthly trading validation also underperforms buy-and-hold. Overall, the thesis shows that a simple three-state HMM with Gaussian emissions is only partly adequate for the CSI 300, and that data frequency, regime separability, and model fit matter more than the choice among closely related decoders.

CERCS research specialisation: P160 Statistics, operations research, programming, financial and actuarial mathematics.

Key Words: hidden Markov model, segmentation, decoding, market regimes, CSI 300.

VARJATUD MARKOVI MUDELITE RAKENDAMINE FINANTSAEGRIDADE

ANALÜÜSIS

Magistritöö

Zhexin Rao

Lühikokkuvõte

Käesolevas magistritöös uuritakse varjatud Markovi mudelite (HMM) kasutamist tururežiimide tuvastamisel finantsaegridades, pöörates seejuures erilist tähelepanu dekodeerimise rollile hinnatud režiimide tõlgendamisel. Vaadeldakse kolme dekodeerit: Viterbi joondust, punktiviisilisi tinglikke tõenäosusi maksimeerivat PMAP-joondust ning hübriidjoondust, mis tasakaalustab globaalse seisundijada kooskõla ja punktiviisilist klassifitseerimistäpsust. Töö metodoloogiline panus seisneb selles, et hinnatakse nende dekodeerite käitumist segmenteerimisülesannetes ning näidatakse, et väike positiivne hübriidkaal võib toimida ettevaatliku PMAP-lähedase valikuna juhul, kui hinnatud üleminekumaatriks sisaldab null- või nullilähedasi elemente. Simulatsiooniuuring käsitleb sõltuvuse, emissioonijaotuste kattuvuse ja hübriidkaalu mõju dekodeerimise kvaliteedile. Tulemused näitavad, et dekodeerimine muutub oluliselt raskemaks siis, kui sõltuvus on nõrk ja emissioonijaotused kattuvad tugevalt. PMAP saavutab üldjuhul veidi väiksema klassifitseerimisvea kui Viterbi, samal ajal kui hübriidjoondus, kus Viterbi on väikese kaaluga, jääb täpsuselt PMAPile lähedaseks. Empiirilises osas rakendatakse CSI 300 indeksi nädala- ja fikseeritud intervalliga kuutootlustele kolme olekuga HMM-i, mille emissioonijaotused on Gaussi jaotused. Nädalasagedusel muudavad hinnatud režiimide emissioonijaotuste tugev kattuvus ja ebarealistlik üleminekutõenäosuste dünaamika klassifitseerimise majandusliku tõlgenduse nõrgaks. Kuutootlustel põhinev mudel on paremini tõlgendatav, eriti valimi varasemas osas, kuid pärast 2016. aastat liigitatakse enamik vaatlusi trendita režiimi, mistõttu muutub režiimikaart liiga jämedaks. Ka paisuva aknaga kuine kauplemisreegli valideerimine jääb ostu-ja-hoia strateegiale alla. Kokkuvõttes näitab töö, et lihtne kolme olekuga HMM Gaussi emissioonijaotustega sobib CSI 300 jaoks ainult osaliselt ning et andmesagedus, režiimide eristatavus ja mudeli sobivus mängivad olulisemat rolli kui dekodeerimismeetodi valik.

CERCS teaduseriala: P160 Statistika, operatsioonianalüüs, programmeerimine, finants- ja kindlustusmatemaatika.

Märksõnad: varjatud Markovi mudel, segmenteerimine, dekodeerimine, tururežiimid, CSI 300.

Contents

Introduction	5
1 Hidden Markov Model	7
1.1 Model Definition	7
1.2 Forward and Backward Probabilities	9
1.3 Decoding Methods	12
1.3.1 Viterbi Algorithm	12
1.3.2 PMAP Path	14
1.3.3 Hybrid Decoding Method	15
2 Empirical Characteristics of Financial Returns	18
2.1 CSI 300 Index Data	18
2.2 Non-Normality and Heavy Tails	19
2.3 Volatility Clustering	22
2.4 Impact of Time Intervals on Return Distributions	23
3 Market Regimes and Identification Methods	25
3.1 Review of Identification Methods	25
3.2 Implications	27
4 Simulation Study	28
4.1 Impact of State Dependence	28
4.2 Impact of Emission Overlap	30
4.3 Choice of the Weight Parameter ω	31
4.4 Implications for the Empirical Analysis	33
5 Application to CSI 300	35
5.1 Empirical Setup	35
5.2 Application 1: Regime Identification	36

5.2.1	Weekly Return Analysis	36
5.2.2	Monthly Return Analysis	38
5.3	Application 2: Monthly Trading Validation	40
5.4	Discussion	43
References		45
A Appendix A: Simulation Code		47
A.1	Shared HMM Utilities	47
A.2	Simulation Driver	55
B Appendix B: CSI 300 Application Code		63
B.1	CSI 300 Application Driver	63

Introduction

Since the 1960s, Hidden Markov Models (HMMs) have found widespread application in fields such as speech recognition and pattern identification due to their ability to model observable sequences driven by unobservable (hidden) states (Rabiner, 1989). In an HMM, a sequence of observations $X = (X_1, \dots, X_n)$ is assumed to depend on a latent sequence of states $Z = (Z_1, \dots, Z_n)$, where Z follows a Markov chain with K possible states. We denote by $x = (x_1, \dots, x_n)$ and $z = (z_1, \dots, z_n)$ the realizations of X and Z respectively, and let $t \in \{1, \dots, n\}$ index the time steps.

The primary goal in segmentation problems is to estimate the underlying state sequence Z given the observation sequence X , a process known as decoding in speech recognition. This is also relevant in financial time series, where hidden states may represent market regimes, such as bear market, sidewalk market, and bull market in stock returns or low-price regime, high-price regime, and spike-regime in electricity prices (Apergis et al., 2019; Liu and Wang, 2017). Common segmentation methods include the Viterbi path, which maximizes the posterior probability $P(Z = z|X = x)$, i.e., the probability of the state path for given observation sequence, and the *pointwise maximum a posteriori* (PMAP path), which maximizes the pointwise posterior probabilities $P(Z_t = z_t|X = x)$ for each time step. Additionally, hybrid estimator interpolates between Viterbi and PMAP by optimizing a weighted objective function and balances global sequence likelihood with local state accuracy (Lember and Koloydenko, 2014). For a weight parameter $\omega \in [0, 1]$, this estimator maximizes:

$$\omega \ln P(Z = z|X = x) + (1 - \omega) \sum_{t=1}^n \ln P(Z_t = z_t|X = x),$$

where $\omega \in [0, 1]$ controls the trade-off, with $\omega = 0$ corresponding to PMAP and $\omega = 1$ to Viterbi.

The Viterbi and PMAP segmentation methods each have distinct characteristics. The resulting sequence from Viterbi path may have poor accuracy in terms of correctly classified states, tending to be more conservative with fewer state transitions compared to PMAP, a behaviour that also depends on the structure of the transition matrix (Lember and Koloydenko, 2014). Moreover, the Viterbi path is more sensitive to model specification than the PMAP path, meaning small changes in model assumptions or parameters can significantly alter the decoded sequence (Zucchini, MacDonald, and Langrock, 2016). The PMAP estimator selects the most likely state at each time step, and because of that it may produce state sequences with low or zero posterior probability, including transitions that are not permitted by the model's transition matrix. We denote by $p_{ij} = P(Z_{t+1} = j|Z_t = i)$ the transition probability from state i to state j , for $i, j \in \{1, \dots, K\}$. We define an *admissible* transition as one where the transition probability $p_{ij} > 0$, meaning a transition from state i to

state j is allowed under the model’s transition matrix $\mathbb{P}$. Conversely, an *inadmissible* transition occurs when $p_{ij} = 0$ theoretically, indicating a prohibited move. Hybrid estimator combines these approaches, adjusting the balance between global and local objectives via the parameter ω . As ω increases, the estimator prioritizes the global posterior probability $P(Z = z|X = x)$, favoring state sequences that maximize the joint posterior probability, similar to Viterbi, which avoids inadmissible transitions but may sacrifice the expected number of correctly estimated states. Conversely, lower ω values emphasize pointwise posterior probabilities $P(Z_t = z_t|X = x)$, resembling PMAP’s behavior. These characteristics, alongside the specific demands of segmentation tasks, guide the selection of decoding methods. For instance, in applications like gene identification, the pointwise error rate is not always the primary metric (Lember and Koloydenko, 2014). In contrast, in financial market analysis, accurately identifying market regimes, especially bear and bull markets, is critical for optimizing portfolio decisions, making the accuracy of detecting these specific states a key consideration.

Given the distinct strengths and limitations of the Viterbi and PMAP paths, their performance must be evaluated under controlled conditions to understand their behavior in segmentation tasks. The aim of this thesis is to investigate the efficacy of HMM segmentation methods for segmentation problems with a focus on financial time series analysis. Specifically, it seeks to:

1. Illustrate and compare the behavior of Viterbi, PMAP and hybrid decoding methods through simulations, examining factors such as sequence length, emission distribution overlap, and the strength of dependence structure in the underlying state sequence to assess their impact on decoding accuracy and the adherence to admissibility constraints on state transitions.
2. Apply HMM to financial return series, to detect market regimes such as bear, sidewalk, and bull markets, evaluating the practical performance of decoding methods and the ability of HMMs to capture empirical properties of financial data, such as heavy tails and volatility clustering in real-world financial contexts (McNeil, Frey, and Embrechts, 2015).

Acknowledgements

I would like to thank my supervisor, Kristi Kuljus, for her guidance and comments during the preparation of this thesis. I appreciate the time she devoted to reading and discussing the manuscript. I also acknowledge that the thesis could likely have been improved further through earlier and more regular consultation during the later stages of the work.

1 Hidden Markov Model

This thesis builds on the Hidden Markov Model (HMM), a probabilistic framework widely used for modeling sequences of observations driven by an underlying, unobservable state sequence. In financial applications, HMM can be valuable for modeling economic cycles or market regimes, such as bear, bull, or sideways markets, modelled as latent states that govern the distribution of observable returns or prices (Ang and Timmermann, 2011). This section introduces the HMM framework, its notation, and the mathematical properties that underpin its application to segmentation tasks. The formulation presented here, including the forward-backward algorithm, is based on the foundational work of Rabiner (1989).

1.1 Model Definition

An HMM describes a sequence of observations $X^n = (X_1, X_2, \dots, X_n)$, where each $X_t \in \mathcal{X}$, typically a subset of $\mathbb{R}$ (e.g., log returns of financial assets). These observations are driven by a latent sequence of states $Z^n = (Z_1, Z_2, \dots, Z_n)$, where each Z_t takes values in a finite state space $\mathcal{S} = \{1, 2, \dots, K\}$. The model is parameterized by $\lambda = (\pi, \mathbb{P}, \{f_i\})$, consisting of three components:

1. **Initial state distribution:** The initial state distribution is defined by the probabilities $\pi_i = P(Z_1 = i)$ for each state $i \in \mathcal{S}$, satisfying $\sum_{i=1}^K \pi_i = 1$.
2. **Transition matrix:** The state dynamics are governed by a $K \times K$ transition matrix $\mathbb{P} = (p_{ij})$, where $p_{ij} = P(Z_{t+1} = j \mid Z_t = i)$ represents the probability of transitioning from state i to state j , with $\sum_{j=1}^K p_{ij} = 1$ for each i . In this thesis, we assume homogeneity, meaning that these transition probabilities are constant over time, ensuring time-invariant state dynamics. In financial applications, these transition probabilities capture the dynamics of market regimes, such as the persistence of a bull market or the likelihood of switching to a bear market, often structured to reflect realistic economic cycles (Hamilton, 1989).
3. **Emission distribution:** The emission distribution for each state $i \in \mathcal{S}$ is given by a density function $f_i(x)$, representing the likelihood of observing x given state i , where $f_i(x) = f(x \mid Z_t = i)$. In this thesis, the emission distribution is modeled as a Gaussian distribution, thus $f_i(x)$ is the density of $\mathcal{N}(\mu_i, \sigma_i^2)$, due to its computational simplicity, with state-specific means and variances capturing variations in return behavior across regimes. To better account for stylized facts of asset returns such as heavy tails or volatility clustering, more flexible dis-

tributions such as t -distributions or normal inverse Gaussian distributions may be employed (Cont, 2001).

These components enable HMMs to model market regimes, such as bear, bull, or sideways markets, where each state corresponds to distinct statistical properties of returns, with state-specific means and variances reflecting the different return and risk characteristics of each regime.

The HMM is typically defined under two simplifying assumptions, which, while not strictly necessary, are commonly adopted for simplicity.

1. **First-order Markov property:** The state at time $t + 1$ depends only on the state at time t , such that

$$P(Z_{t+1} = j \mid Z_1, \dots, Z_t) = P(Z_{t+1} = j \mid Z_t = i) = p_{ij}.$$

This ensures that the state sequence forms a Markov chain, where future states are conditionally independent of past states given the current state. If the Markov chain satisfies the following conditions:

- **Irreducibility:** every state can be reached from any other state, and
- **Aperiodicity:** the chain does not cycle in a fixed pattern,

then it possesses a stationary distribution π^* , which represents the long-run probability of being in each state. Formally, the stationary distribution satisfies $\pi^* \mathbb{P} = \pi^*$, meaning that the state probabilities remain unchanged after applying the transition matrix $\mathbb{P}$. Equivalently, for an initial state distribution π_0 , the stationary distribution can be characterized as

$$\pi^* = \lim_{n \rightarrow \infty} \pi_0 \mathbb{P}^n,$$

assuming the limit exists.

2. **Conditional independence of observations:** The observation at time t depends only on the current state Z_t , such that

$$p(X_t \mid Z_1, \dots, Z_n) = f_{Z_t}(X_t).$$

This implies that observations are conditionally independent given the state sequence:

$$p(X^n = x^n \mid Z^n = z^n) = \prod_{t=1}^n f_{z_t}(x_t).$$

1.2 Forward and Backward Probabilities

To compute probabilities associated with state sequences given observations, two key quantities are introduced: the forward and backward probabilities. These probabilities are essential for inferring the latent state sequence. In decoding, they support PMAP which assigns states based on maximum pointwise posterior probabilities. For parameter estimation, the Baum-Welch algorithm, a form of the expectation-maximization (EM) algorithm, uses these probabilities to estimate model parameters.

The forward probability at time t for state i , denoted $\alpha_t(i)$, is the likelihood of observing the first t observations and being in state i at time t , given the model parameters $\lambda = (\pi, \mathbb{P}, \{f_i\})$:

$$\alpha_t(i) = p(x_1, \dots, x_t, Z_t = i).$$

The forward probabilities can be computed recursively:

(1) for $t = 1$:

$$\alpha_1(i) = \pi_i f_i(x_1), \quad i = 1, \dots, K,$$

(2) for $t = 2, \dots, n$:

$$\alpha_t(j) = \left[\sum_{i=1}^K \alpha_{t-1}(i) p_{ij} \right] f_j(x_t), \quad j = 1, \dots, K.$$

The forward probabilities $\alpha_t(i)$ provide a recursive framework to compute the joint likelihood of the observation sequence up to time t and the state at time t . Summing these probabilities over all possible states at time t yields the likelihood of the observation sequence $x_1, \dots, x_t$ given the model parameters λ :

$$p(x_1, \dots, x_t) = \sum_{i=1}^K \alpha_t(i).$$

The backward probability at time t for state i , denoted $\beta_t(i)$, is the likelihood of observing the remaining observations from $t+1$ to n given that the state at time t is i , given the model parameters λ :

$$\beta_t(i) = p(x_{t+1}, \dots, x_n \mid Z_t = i).$$

The backward probabilities can also be computed recursively:

(1) for $t = n$:

$$\beta_n(i) = 1, \quad i = 1, \dots, K,$$

(2) for $t = n - 1, \dots, 1$:

$$\beta_t(i) = \sum_{j=1}^K p_{ij} f_j(x_{t+1}) \beta_{t+1}(j).$$

In the recursive computation of the forward probability $\alpha_t(j)$, the previous step's probabilities $\alpha_{t-1}(i)$ are multiplied by p_{ij} and $f_j(x_t)$. When processing long sequences, $\alpha_t(j)$ can decay exponentially, approaching zero and risking computational errors due to the finite precision of numerical systems. A similar issue arises with the backward probabilities $\beta_t(i)$. To address this, scaled forward and backward probabilities $\hat{\alpha}_t(j)$ and $\hat{\beta}_t(j)$ are used in numerical computations, normalizing the probabilities at each time step, as described in Bishop (2006).

Scaled forward probability $\hat{\alpha}_t(i)$ is defined as the conditional probability of the state at time t given the observations up to time t :

$$\hat{\alpha}_t(i) = P(Z_t = i \mid x_1, \dots, x_t) = \frac{p(x_1, \dots, x_t, Z_t = i)}{p(x_1, \dots, x_t)},$$

thus

$$\sum_{i=1}^K \hat{\alpha}_t(i) = 1.$$

To compute the scaled forward probabilities recursively, we introduce a scaling factor c_t , defined as the marginal likelihood of the observation x_t given the previous observations:

$$c_t = p(x_t \mid x_1, \dots, x_{t-1}).$$

By the chain rule of probability we have:

$$p(x_1, \dots, x_n) = p(x_1) p(x_2 \mid x_1) \cdots p(x_n \mid x_1, \dots, x_{n-1}) = \prod_{t=1}^n c_t.$$

For the initial step ($t = 1$):

$$c_1 = p(x_1) = \sum_{i=1}^K \pi_i f_i(x_1),$$

thus

$$\hat{\alpha}_1(i) = \frac{\pi_i f_i(x_1)}{c_1}.$$

For $t = 2, \dots, n$, observe that by the definition of $\alpha_t(j)$ and the forward recursion:

$$\alpha_t(j) = \left[\sum_{i=1}^K \alpha_{t-1}(i) p_{ij} \right] f_j(x_t).$$

Since $\hat{\alpha}_{t-1}(i) = \alpha_{t-1}(i)/p(x_1, \dots, x_{t-1})$, substituting $\alpha_{t-1}(i) = \hat{\alpha}_{t-1}(i) \cdot p(x_1, \dots, x_{t-1})$ gives:

$$c_t \hat{\alpha}_t(j) = \left[\sum_{i=1}^K \hat{\alpha}_{t-1}(i) p_{ij} \right] f_j(x_t), \quad j = 1, \dots, K.$$

Summing over all states j and using $\sum_{j=1}^K \hat{\alpha}_t(j) = 1$, c_t can be computed as

$$c_t = \sum_{j=1}^K \left[f_j(x_t) \sum_{i=1}^K \hat{\alpha}_{t-1}(i) p_{ij} \right].$$

Since c_t is now determined, we obtain the recursion for the scaled forward probability:

$$\hat{\alpha}_t(j) = \frac{1}{c_t} \left[f_j(x_t) \sum_{i=1}^K \hat{\alpha}_{t-1}(i) p_{ij} \right], \quad j = 1, \dots, K.$$

Similarly, the scaled backward probability is defined as

$$\hat{\beta}_t(i) = \frac{p(x_{t+1}, \dots, x_n \mid Z_t = i)}{p(x_1, \dots, x_n \mid x_1, \dots, x_t)} = \frac{\beta_t(i)}{\prod_{m=t+1}^n c_m}.$$

Analogously to the scaled forward recursion, observe that:

$$c_{t+1} \hat{\beta}_t(i) = \sum_{j=1}^K p_{ij} f_j(x_{t+1}) \hat{\beta}_{t+1}(j), \quad i = 1, \dots, K,$$

where the scaling factors $c_1, \dots, c_n$ are already available from the forward recursion. Since c_{t+1} is known, we obtain the recursion for the scaled backward probability:

$$\hat{\beta}_t(i) = \frac{1}{c_{t+1}} \sum_{j=1}^K p_{ij} f_j(x_{t+1}) \hat{\beta}_{t+1}(j), \quad i = 1, \dots, K,$$

with the terminal condition:

$$\hat{\beta}_n(i) = 1, \quad i = 1, \dots, K.$$

The joint probability of being in state i at time t and observing all observations can be decomposed by splitting the observation sequence at time t :

$$p(Z_t = i, x^n) = p(x_1, \dots, x_t, Z_t = i) \cdot p(x_{t+1}, \dots, x_n \mid Z_t = i, x_1, \dots, x_t).$$

By the conditional independence assumption, the future observations $x_{t+1}, \dots, x_n$ depend on the

past only through the current state $Z_t = i$, so:

$$p(x_{t+1}, \dots, x_n \mid Z_t = i, x_1, \dots, x_t) = p(x_{t+1}, \dots, x_n \mid Z_t = i) = \beta_t(i).$$

Since $p(x_1, \dots, x_t, Z_t = i) = \alpha_t(i)$ by definition, we obtain:

$$p(Z_t = i, x^n) = \alpha_t(i)\beta_t(i).$$

1.3 Decoding Methods

Decoding in HMMs involves estimating the latent state sequence $Z^n = (Z_1, \dots, Z_n)$ given the observation sequence $X^n = x^n$. Three decoding methods—Viterbi, PMAP, and a hybrid method based on Lember and Koloydenko (2014)—are presented in this subsection. A key quantity in decoding is the smoothing probability, denoted $\gamma_t(i) := P(Z_t = i \mid x^n)$, which represents the posterior probability of being in state i at time t given all observations. It can be expressed as

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{p(x_1, \dots, x_n)},$$

where $p(x_1, \dots, x_n)$ can be expressed in terms of the scaling factors as

$$p(x_1, \dots, x_n) = \prod_{t=1}^n c_t,$$

Thus, using the scaled forward and backward probabilities, $\gamma_t(i)$ can be easily computed as

$$\gamma_t(i) = \hat{\alpha}_t(i)\hat{\beta}_t(i).$$

In our segmentation tasks, $\gamma_t(i)$ quantifies the probability of a specific market regime at time t .

1.3.1 Viterbi Algorithm

The Viterbi algorithm is a dynamic programming algorithm which finds the most likely state sequence $Z^n = (Z_1, \dots, Z_n)$ given the observation sequence $X^n = x^n$. It maximizes the posterior probability $P(Z^n = z^n \mid X^n = x^n)$, which is equivalent to maximizing the joint likelihood $p(Z^n = z^n, X^n = x^n)$, since $p(X^n = x^n)$ does not depend on z^n and thus acts as a constant. Formally, the

Viterbi algorithm computes:

$$\widehat{z^n} = \arg \max_{z^n} p(Z^n = z^n, X^n = x^n),$$

where the joint likelihood can be expressed as

$$p(Z^n = z^n, X^n = x^n) = P(Z^n = z^n) \cdot p(x^n | Z^n = z^n).$$

The Viterbi algorithm introduces two key variables. The first is $\delta_t(i)$, defined as the maximum joint likelihood over all state sequences ending in state i at time t for the observations $x_1, \dots, x_t$:

$$\delta_t(i) = \max_{z_1, \dots, z_{t-1}} p(Z_t = i, Z_1 = z_1, \dots, Z_{t-1} = z_{t-1}, x_1, \dots, x_t), \quad i = 1, \dots, K.$$

The second is $\psi_t(i)$, which tracks the previous state that leads to this maximum likelihood, facilitating backtracking to construct the optimal state sequence:

$$\psi_t(i) = \arg \max_{j \in \mathcal{S}} \delta_{t-1}(j) p_{ji}, \quad i = 1, \dots, K.$$

The algorithm based on Rabiner (1989) is outlined below.

Viterbi Algorithm:

1. **Initialization:** for each state $i \in \mathcal{S}$, define:

$$\delta_1(i) = \pi_i f_i(x_1),$$

and set $\psi_1(i) = 0$ (no previous state at $t = 1$).

2. **Recursion:** for $t = 2, \dots, n$, and each state $j \in \mathcal{S}$, compute:

$$\delta_t(j) = \max_{i \in \mathcal{S}} [\delta_{t-1}(i) p_{ij}] f_j(x_t),$$

and store the previous state that gives this maximum:

$$\psi_t(j) = \arg \max_{i \in \mathcal{S}} [\delta_{t-1}(i) p_{ij}].$$

This step evaluates all possible transitions from state i at time $t - 1$ to state j at time t , selecting the path with the highest likelihood.

3. **Termination:** identify the highest likelihood at the final time step n :

$$\delta^* = \max_{i \in \mathcal{S}} \delta_n(i),$$

and set the final state of the optimal sequence as

$$\hat{z}_n = \arg \max_{i \in \mathcal{S}} \delta_n(i).$$

4. **Backtracking:** for $t = n - 1, \dots, 1$, reconstruct the state sequence by following the stored previous states:

$$\hat{z}_t = \psi_{t+1}(\hat{z}_{t+1}).$$

The Viterbi algorithm is valuable for its ability to produce a globally consistent state sequence that respects the Markov chain's transition probabilities. However, maximizing the global posterior probability $P(Z^n = z^n \mid X^n = x^n)$ does not guarantee optimality at the level of individual time points. The state $\hat{z}_t$ selected by Viterbi at time t may not be the state with the highest marginal posterior probability $P(Z_t = z_t \mid X^n = x^n)$. For instance, if multiple state sequences have similar probabilities, they may differ significantly in their state sequences, potentially overlooking alternatives that achieve higher accuracy in terms of correctly classified states. This limitation may reduce its effectiveness in applications where minimizing the pointwise misclassification rate is the primary objective..

1.3.2 PMAP Path

The *pointwise maximum a posteriori* (PMAP) path is defined as the state path obtained by independently maximizing the posterior probability at each time step.. Unlike the Viterbi algorithm, which seeks the globally optimal state sequence by maximizing $P(Z^n = z^n \mid X^n = x^n)$, PMAP focuses on local optimality by selecting the most likely state at each time t based on the full observation sequence $X^n = x^n$.

For each time $t = 1, \dots, n$, PMAP selects the state $\hat{z}_t$ as:

$$\hat{z}_t = \arg \max_{i \in \mathcal{S}} \gamma_t(i),$$

where the smoothing probability $\gamma_t(i)$ is computed using the scaled forward and backward probabilities $\gamma_t(i) = \hat{\alpha}_t(i)\hat{\beta}_t(i)$, $i = 1, \dots, K$.

The PMAP path is optimal in the sense that it maximizes the expected number of correctly classified states, which makes it preferable to the Viterbi algorithm when the primary objective is pointwise accuracy. However, the PMAP path assigns states independently at each time step, which means the transition probabilities p_{ij} are not taken into account in the assignment. As a result, the state sequence $\hat{z}_1, \hat{z}_2, \dots, \hat{z}_n$ may include transitions that are unlikely or even impossible under the model's transition matrix $\mathbb{P}$. For example, if the transition matrix specifies that $p_{ij} = 0$ for some states i and j , indicating that a transition from state i to state j is prohibited, PMAP might still assign $\hat{z}_t = i$ and $\hat{z}_{t+1} = j$ if these states individually maximize the posterior probabilities at times t and $t+1$. This lack of global consistency contrasts with the Viterbi algorithm, which respects the transition matrix by optimizing the joint probability of the entire state sequence, ensuring that all transitions are feasible under $\mathbb{P}$.

1.3.3 Hybrid Decoding Method

Lember and Koloydenko (2014) refine Rabiner's idea of a decoder maximizing the expected number of correctly inferred k -blocks, which targets subsequences of length k . They propose a family of decoders based on k -blocks, defined as subsequences of k consecutive states, $Z_t^{t+k-1} = (Z_t, Z_{t+1}, \dots, Z_{t+k-1})$, for $1 \leq t \leq n - k + 1$. Instead of summing block-wise posterior probabilities as in Rabiner's original approach, Lember and Koloydenko (2014) replace the sum with a product, yielding a subfamily of decoders that produce paths with positive posterior probability for $k \geq 2$, with the posterior path probability increasing monotonically with k , thereby eliminating inadmissible transitions that violate the HMM's transition constraints. For $k = 1$, the decoder maximizes the smoothing probability $\gamma_t(z_t) = P(Z_t = z_t \mid X^n = x^n)$, yielding the PMAP path. For $k = n$, the single block covers the entire state sequence $Z_1^n = (Z_1, \dots, Z_n)$, so maximizing the block posterior probability is equivalent to maximizing $P(Z^n = z^n \mid X^n = x^n)$, aligning with Viterbi algorithm. Intermediate k values balance these objectives. They further formulate a hybrid decoder which is parameterized by $\omega \in [0, 1]$, where $\omega = \frac{k-1}{k}$ links the block size k to the decoder's behavior. This is achieved by optimizing a weighted objective that blends two goals: maximizing the overall probability of the state sequence (like Viterbi) and maximizing the probability of each state at individual time points (like PMAP). The weighting parameter $\omega \in [0, 1]$ controls this trade-off. It solves the optimization problem:

$$\widehat{z^n} = \arg \max_{z^n} \left[\omega \ln P(Z^n = z^n \mid X^n = x^n) + (1 - \omega) \sum_{t=1}^n \ln P(Z_t = z_t \mid X^n = x^n) \right],$$

where $P(Z_t = z_t \mid X^n = x^n) = \gamma_t(z_t) = \hat{\alpha}_t(z_t)\hat{\beta}_t(z_t)$.

Since

$$P(Z^n = z^n \mid X^n = x^n) = \frac{p(Z^n = z^n, X^n = x^n)}{p(X^n = x^n)},$$

and $p(X^n = x^n)$ is constant for a given observation sequence, the objective can be rewritten as:

$$\widehat{z^n} = \arg \max_{z^n} \left[\omega \ln p(Z^n = z^n, X^n = x^n) + (1 - \omega) \sum_{t=1}^n \ln \gamma_t(z_t) \right],$$

where $p(Z^n = z^n, X^n = x^n) = \pi_{z_1} f_{z_1}(x_1) \prod_{t=2}^n p_{z_{t-1}, z_t} f_{z_t}(x_t)$.

The algorithm computes a score $\delta_t(j)$ for each state j at time t as outlined below.

Hybrid Path Algorithm for Given ω :

1. **Initialization:** for each state $i \in \mathcal{S}$, compute the initial score, combining the pointwise posterior and joint probability terms:

$$\delta_1(i) = (1 - \omega) \ln \gamma_1(i) + \omega \ln(\pi_i f_i(x_1)).$$

Set $\psi_1(i) = 0$.

2. **Recursion:** for $t = 2, \dots, n$, and each state $j \in \mathcal{S}$, compute the score by evaluating transitions from all possible previous states i :

$$\delta_t(j) = \max_{i \in \mathcal{S}} [\delta_{t-1}(i) + (1 - \omega) \ln \gamma_t(j) + \omega \ln(p_{ij} f_j(x_t))],$$

and store the previous state that maximizes the score, considering only the terms that depend on i :

$$\psi_t(j) = \arg \max_{i \in \mathcal{S}} [\delta_{t-1}(i) + \omega \ln p_{ij}].$$

3. **Termination:** at the final time step n , identify the highest score to determine the optimal final state:

$$\delta^* = \max_{i \in \mathcal{S}} \delta_n(i),$$

and select the corresponding state:

$$\hat{z}_n = \arg \max_{i \in \mathcal{S}} \delta_n(i).$$

4. **Backtracking:** reconstruct the state sequence by tracing back through the stored previous

states, from $t = n - 1$ down to 1:

$$\hat{z}_t = \psi_{t+1}(\hat{z}_{t+1}).$$

The recursion evaluates transitions, incorporating $\ln \gamma_t(j) = \ln(\hat{\alpha}_t(j)\hat{\beta}_t(j))$ and the log-likelihood of transitioning to state j and observing x_t . The termination and backtracking steps construct the state sequence, similarly to Viterbi.

2 Empirical Characteristics of Financial Returns

Financial returns often exhibit empirical properties. These properties, commonly referred to as stylized facts, represent consistent patterns observed across various markets and assets (Cont, 2001). Three such stylized facts — non-normality, fat tails, and volatility clustering — are examined in this chapter (McNeil, Frey, and Embrechts, 2015). The analysis focuses on their manifestation in the CSI 300 index, a capitalization-weighted index tracking the top 300 stocks traded on the Shanghai and Shenzhen Stock Exchanges, which serves as a key benchmark for the Chinese stock market. Using approximately 20 years of daily returns from April 8, 2005, to May 13, 2025, we examine how these stylized facts manifest in the data, and how their statistical properties change as returns are aggregated from daily to weekly and monthly frequencies.

Understanding these characteristics is crucial prior to applying HMM for two reasons. First, they are relevant to the specification of emission distributions in the HMM. It is important to note that non-normality observed in the marginal return distribution does not necessarily imply that assuming normality within each state is inadequate. Nevertheless, if heavy-tailed behavior persists within individual states, more flexible emission distributions, such as the Student's t or normal inverse Gaussian distributions, may better capture return behavior (Cont, 2001). Second, these stylized facts facilitate the interpretation of the HMM's outputs and help us validate its effectiveness. For instance, volatility clustering, where periods of high volatility persist, relates directly to the transition matrix of the HMM, as persistent states (high diagonal probabilities in the transition matrix) can model prolonged high-volatility regimes.

2.1 CSI 300 Index Data

The data consists of daily closing prices of the CSI 300 index, sourced from Investing.com. Daily returns are calculated using the logarithmic formula:

$$r_t = 100 \times (\ln S_t - \ln S_{t-1}),$$

where S_t is the closing price on day t . For longer intervals, we define the h -period log-return as:

$$r_t^{(h)} = 100 \times \left[\ln \frac{S_t}{S_{t-1}} + \ln \frac{S_{t-1}}{S_{t-2}} + \dots + \ln \frac{S_{t-(h-1)}}{S_{t-h}} \right] = 100 \times (\ln S_t - \ln S_{t-h}),$$

with $h = 5$ for weekly returns (assuming 5 trading days per week) and $h = 22$ for monthly returns, approximating the average number of trading days in a month. We note that $h = 22$ is a fixed

approximation and does not account for variation in the actual number of trading days across months due to public holidays. As a result, the monthly returns computed here are fixed-interval log-returns rather than calendar-month returns, and the aggregation periods may not always align precisely with calendar months. These longer-interval returns allow us to examine how return properties change with data aggregation over time. At each frequency, the first return is undefined because no earlier price is available for comparison, so that initial observation is omitted from the return sample rather than being set equal to zero.

Figure 1 presents the time series of daily returns over the sample period. The plot reveals distinct periods of high volatility, characterized by clusters of extreme positive and negative returns, often corresponding to economic downturns or market crises, interspersed with periods of lower volatility during stable market conditions. This clustering of extreme values suggests that market turbulence persists over time, a phenomenon known as volatility clustering, which is further explored in Section 2.3.

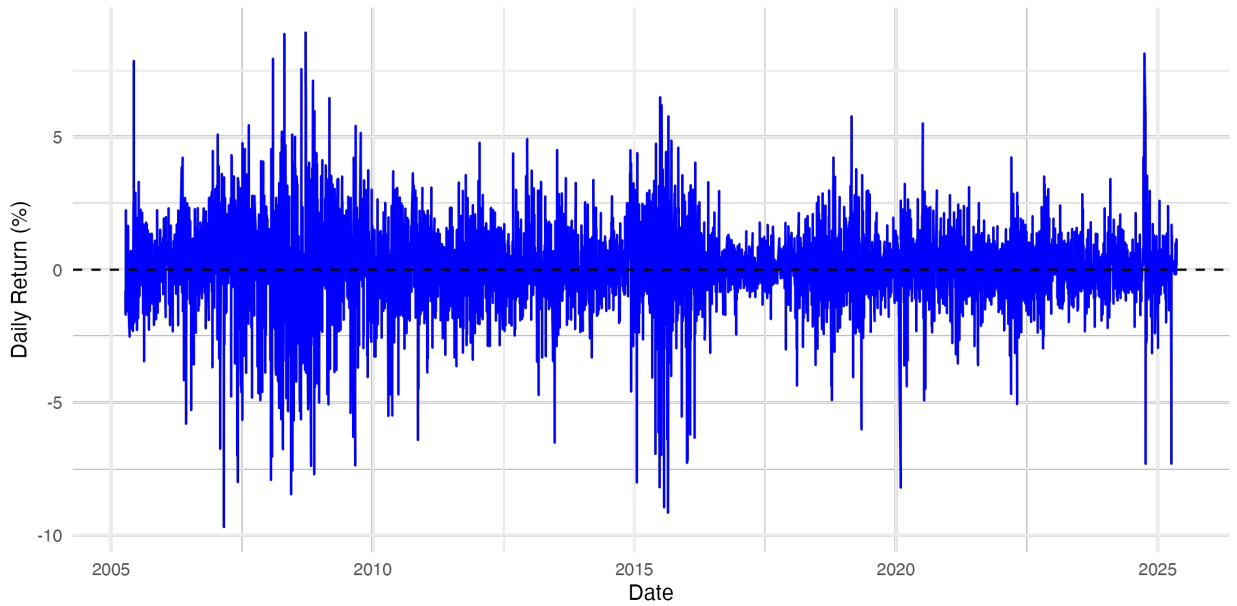


Figure 1: Time series of daily returns for CSI 300.

2.2 Non-Normality and Heavy Tails

Financial returns are often assumed to follow a normal distribution in theoretical models, but empirical evidence consistently shows deviations, including non-normality and heavy tails, often referred to as leptokurtic distributions (McQuire and Kume, 2023). Leptokurtic distributions are characterized by a higher probability of extreme values (fat tails) compared to a normal distribution, making them important for accurately characterizing the tail behavior of return distributions.

To assess the distribution of CSI 300 daily returns, we examine a histogram of daily returns, overlaid with a normal distribution parameterized by the sample mean and standard deviation (Figure 2). The histogram suggests departures from normality, although the heavy-tailed nature of the distribution is not especially clear because most observations are concentrated near the center. For this reason, an additional log-scale plot is used to examine the tails more closely.

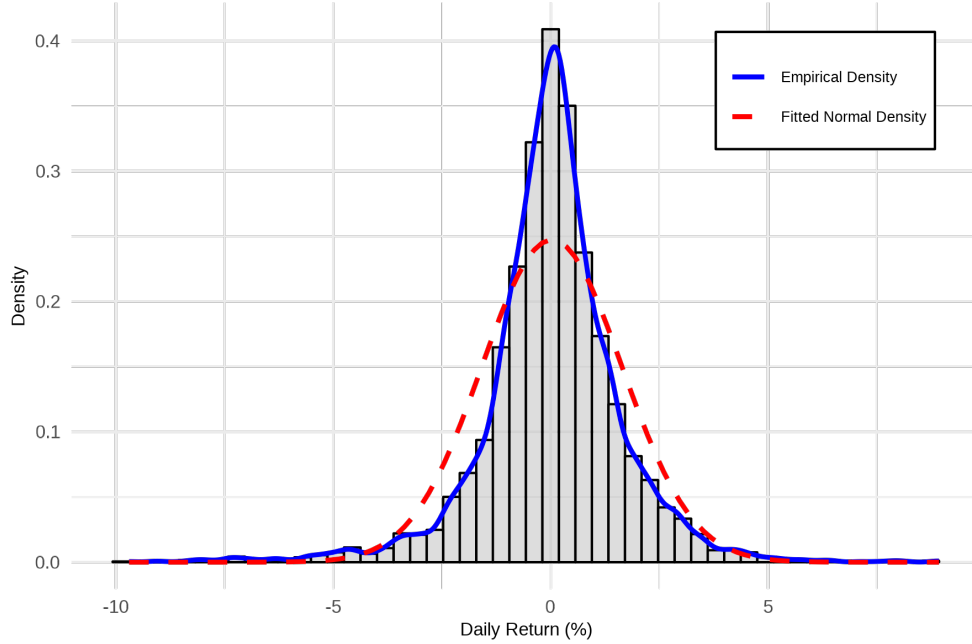


Figure 2: Histogram of daily returns with overlaid normal distribution.

To better visualize tail behavior, Figure 3 presents the distribution of daily returns on a logarithmic scale on the y-axis. This transformation compresses the central part of the distribution and makes the tail regions more visible, allowing deviations from the fitted normal distribution to be seen more clearly (Green, 2014). In particular, the empirical distribution lies above the normal benchmark in the far tails, indicating that extreme positive and negative returns occur more frequently than would be expected under normality. This provides clearer evidence of heavy-tailed behavior in CSI 300 daily returns.

To further demonstrate departures from normality, Figure 4 presents a quantile–quantile (QQ) plot comparing the empirical distribution of daily returns with the quantiles of a normal distribution. The plot shows clear deviations from the diagonal line, particularly in the lower and upper tails, indicating that the marginal distribution of returns is not normal. Such a result is not surprising in financial return series, where the data may reflect changing market conditions over time. The pronounced tail deviations therefore provide further evidence of non-normality and are consistent with heavy-tailed behavior in the return series.

To quantify these deviations from normality, we compute the sample skewness and sample kurtosis

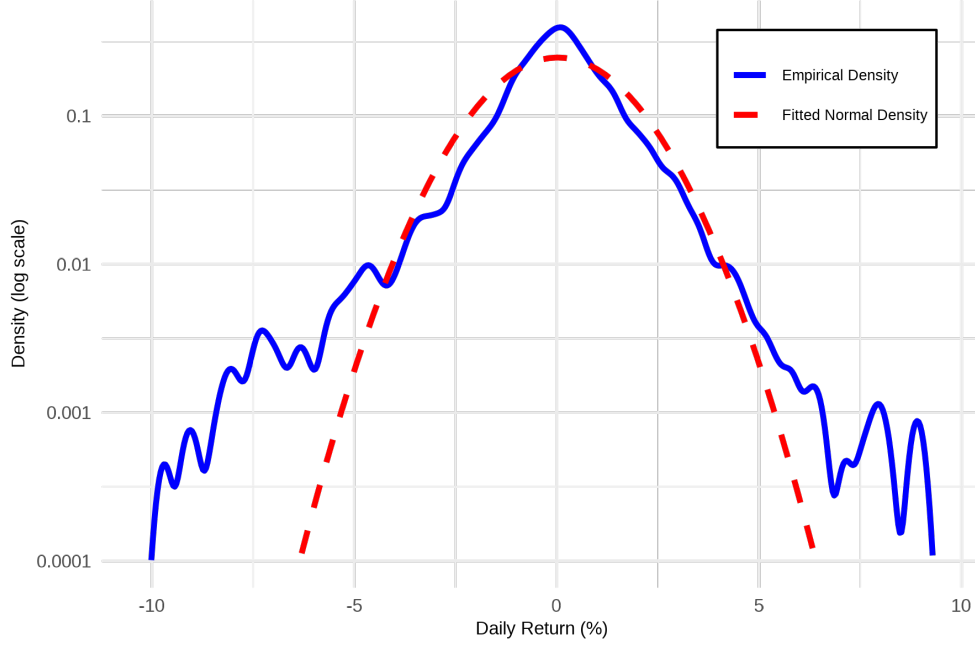


Figure 3: Log-scale density of daily returns.

of returns. Skewness measures the asymmetry of a distribution, with positive values indicating a longer right tail and negative values indicating a longer left tail. Kurtosis measures the heaviness of the tails relative to the center of the distribution. The sample skewness S and sample kurtosis K are defined as

$$S = \frac{\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^3}{s^3}, \quad K = \frac{\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^4}{s^4},$$

where $\bar{x}$ is the sample mean, s is the sample standard deviation, and n is the number of observations. For a normal distribution, kurtosis equals 3. Values greater than 3 indicate a leptokurtic distribution, meaning that extreme observations occur more frequently than under normality.

Table 1 presents summary statistics for daily, weekly, and monthly returns. For daily returns, the skewness is -0.494, indicating a slight left skew, and the kurtosis is 7.492, which substantially exceeds the benchmark value of 3 for a normal distribution. This confirms that the daily return distribution is leptokurtic, with more frequent extreme observations than would be expected under normality. Weekly and monthly returns display similar but less pronounced deviations, as discussed in Section 2.4.

Table 1: Summary statistics for CSI 300 returns.

Period	Mean	Std. Dev.	Skewness	Kurtosis
Daily	0.029	1.616	-0.494	7.492
Weekly	0.143	3.465	-0.194	5.620
Monthly	0.612	7.973	-0.364	4.972

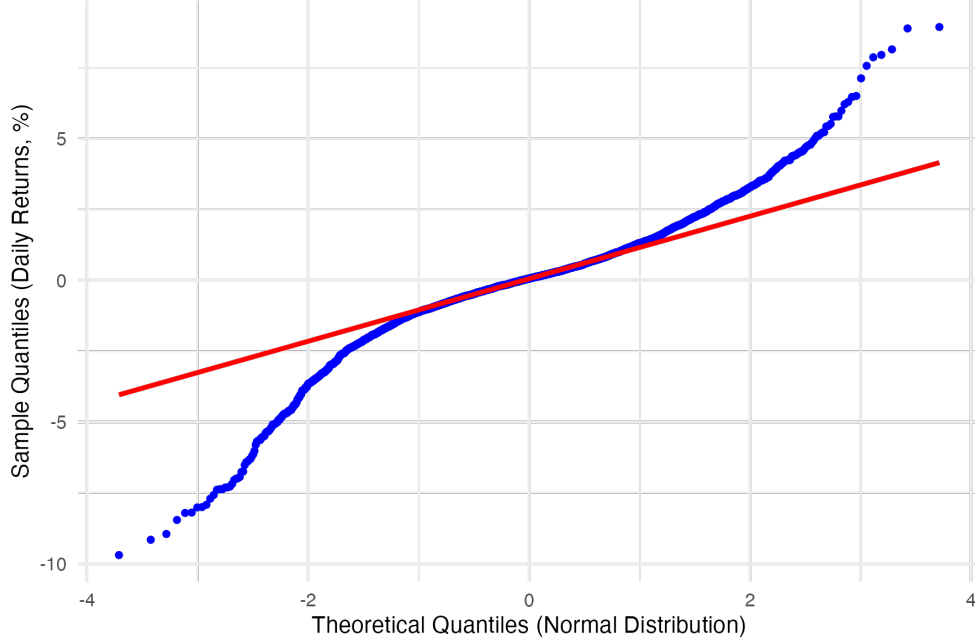


Figure 4: QQ plot of daily returns against normal distribution.

2.3 Volatility Clustering

Volatility clustering is another empirical feature of financial returns: periods of high volatility tend to be followed by further periods of high volatility, while periods of low volatility also tend to persist. This persistence suggests that return is not independent and identically distributed (non-iid), as turbulent market conditions tend to persist, often during economic crises, while stable periods align with calmer markets (McNeil, Frey, and Embrechts, 2015).

To examine volatility clustering, we compute the autocorrelation function (ACF) for daily returns and absolute daily returns, presented in Figure 5. The left panel shows the ACF of daily returns, which is close to zero for most lags. This is broadly consistent with the weak-form efficient market hypothesis (EMH), according to which past price information should not systematically help to predict future returns (Fama, 1970). In a common formulation, this means that demeaned returns — that is, returns after subtracting their mean — behave approximately as a martingale difference sequence with respect to the information set generated by past prices (LeRoy, 1989). That is,

$$E(r_t - \mu \mid \mathcal{F}_{t-1}) \approx 0,$$

where r_t denotes the return at time t , μ is the mean return, and $\mathcal{F}_{t-1}$ is the information available up to time $t - 1$. Thus, the weak autocorrelation in raw returns suggests that past returns contain limited information for predicting future returns.

In contrast, the right panel shows the ACF of absolute daily returns, revealing significant pos-

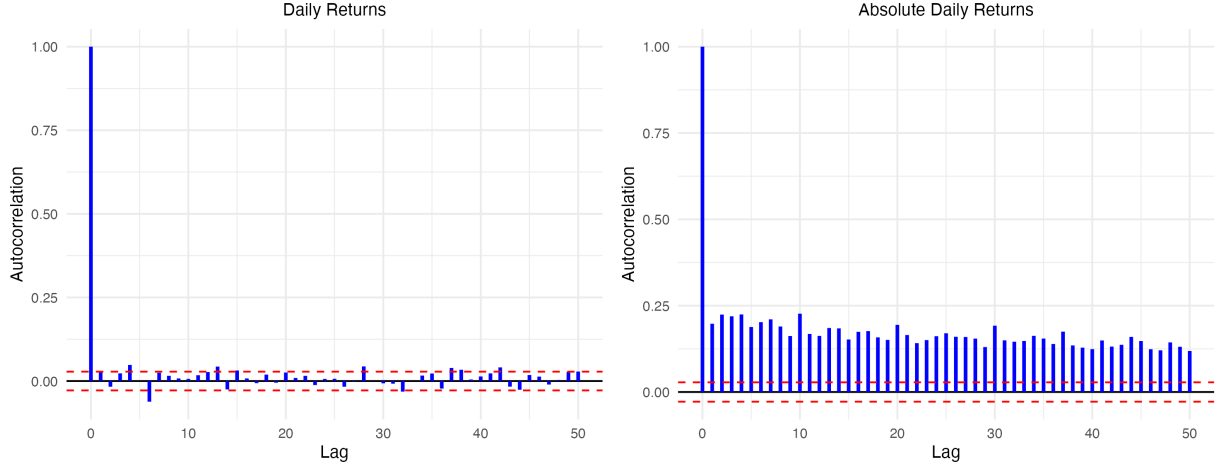


Figure 5: ACF of daily returns (left) and absolute daily returns (right).

itive autocorrelation over a range of lags. This indicates dependence in volatility, as periods of high volatility (large absolute returns) are more likely to be followed by similar periods. Volatility clustering is evident in the time series plot (Figure 1), where clusters of extreme returns appear during turbulent periods. These findings are relevant for the HMM framework. Although an HMM is a probabilistic model and does not necessarily reproduce all features of the return series exactly, persistent hidden states can provide a useful representation of periods of relatively high and low volatility.

2.4 Impact of Time Intervals on Return Distributions

The properties of financial returns often change with the time interval over which they are calculated. In this thesis, the h -period return is defined as

$$r_t^{(h)} = r_t + r_{t-1} + \dots + r_{t-(h-1)},$$

so that it is the sum of h consecutive one-period log-returns. As the aggregation interval increases from daily to weekly to monthly, the return distribution is often observed to become closer to normal. Intuitively, aggregating returns tends to smooth some of the non-normal features that are more pronounced at high frequencies. Although financial returns are not independent, their serial dependence in raw returns is often weak, so arguments related to the central limit theorem can still help explain why the distribution of aggregated returns tends to approach normality under suitable conditions (Cont, 2001).

To explore this, we aggregate daily returns into weekly and monthly returns using the h -period log-return formula and repeat the analyses conducted for daily returns. Table 1 summarizes the

results. For weekly returns, the kurtosis is 5.620, which indicates less pronounced heavy-tailed behavior than for daily returns, although the distribution remains clearly leptokurtic. For monthly returns, the kurtosis decreases further to 4.972, which is closer to the benchmark value of 3 for a normal distribution. Overall, these results suggest that the return distribution becomes closer to normal as the aggregation interval increases, although deviations from normality remain visible, especially at shorter horizons.

To further illustrate the impact of time intervals, we also examine the S&P 500 index, a benchmark for the U.S. stock market, using approximately 20 years of daily returns constructed from closing prices from July 1, 2005, to May 13, 2025, sourced from Investing.com. Returns are calculated using the same logarithmic formula as for the CSI 300. Table 2 presents the summary statistics for daily, weekly, and monthly returns of the S&P 500. For daily returns, the kurtosis is 15.882, substantially higher than the corresponding value of 7.492 for the CSI 300, indicating more pronounced heavy-tailed behavior. Although kurtosis declines as the return horizon increases, it remains well above 3 even at the monthly frequency, suggesting that non-normality is still substantial in the S&P 500 return series.

Table 2: Summary statistics for S&P 500 returns.

Period	Mean	Std. Dev.	Skewness	Kurtosis
Daily	0.031	1.215	-0.530	15.882
Weekly	0.148	2.481	-0.958	11.293
Monthly	0.648	4.397	-0.779	4.548

Although the return distributions of both indices remain non-normal at all aggregation levels considered, time aggregation remains relevant for modeling because it changes the marginal distribution of the observed return series. Longer-interval returns are closer to normality and may therefore be easier to model with Gaussian state-dependent emissions, whereas daily returns retain more pronounced non-normal features and provide a more demanding setting for evaluating model performance. At the same time, these results describe the marginal distribution of returns; they do not by themselves determine whether Gaussian emission distributions are adequate within individual latent states.

3 Market Regimes and Identification Methods

In general, the idea of classifying the state of the stock market into bear and bull markets is similar to identifying recession and expansion periods of real economic activity (Nyberg, 2013). Hamilton (1989) uses Markov-switching models to capture shifts between positive and negative growth regimes in economic time series, inferring these regimes probabilistically from observed growth dynamics; this framework is analogous to modeling bull and bear phases in financial markets. These market regimes represent distinct phases in financial markets, each characterized by unique patterns in asset price movements, volatility, and investor sentiment. The most commonly recognized regimes are the bull market, marked by rising asset prices, investor optimism, and expectations of sustained growth, and the bear market, defined by declining prices, pessimism, and often linked to economic downturns or corrections (Hanna, 2018). These definitions, while broadly accepted, vary in their precise criteria across studies, leading to diverse methods for identifying these regimes.

3.1 Review of Identification Methods

Market regime identification methods can be broadly categorized into deterministic rule-based approaches and statistical models. Rule-based methods rely on deterministic criteria to identify market regimes based on price movements. The seminal work of Bry and Boschan (1971) introduced a systematic algorithm for identifying turning points in business cycles, which has been widely adapted for financial markets to detect bull and bear regimes. Their approach systematically detects peaks and troughs in data series, by applying rules to ensure that turning points are significant and not due to minor fluctuations. This methodology laid the foundation for subsequent rule-based approaches.

Building on this, Pagan and Sossounov (2003) adapted the Bry and Boschan methodology to stock market cycles, introducing the PS approach for identifying bull and bear market phases. Their method employs a multi-pass algorithm with specific rules to detect turning points in price series; phases have to be significant in terms of magnitude and duration. For instance, they require a certain percentage drop or rise in prices to define a bear or bull market, respectively, and impose minimum duration constraints—specifically, phases must last at least 4 months and complete cycles at least 16 months—to avoid identifying short-term fluctuations as regime changes.

Another notable rule-based method is the PZ algorithm, proposed by Zegadło (2022), which repeatedly applies a simplified Bry–Boschan-type turning-point procedure to successive sets of local maxima and minima. The method uses a single parameter—the number of filtering operations—which

helps reduce subjectivity in regime identification. This method enforces the alternation of peaks and troughs to ensure logical coherence in regime transitions and avoids imposing minimum duration or price amplitude constraints. Similarly, the top-down (TD) approach, introduced by Hanna (2018), recursively partitions long-term market trends into sub-trends, creating a hierarchical structure of bull and bear phases. The TD approach adheres to principles such as end-boundedness, parameter stability, and no historical revisions. It prohibits *intermediate extrema*—local maximums or minimums occurring within a market phase, between its starting and ending points—to ensure that each phase is defined by significant turning points, avoiding minor fluctuations that could obscure major trends. Additionally, the TD approach incorporates a symmetry constraint, defined as $\lambda_{\text{bear}}/(1 - \lambda_{\text{bear}}) = \lambda_{\text{bull}}$, where λ_{bull} and λ_{bear} are percentage thresholds for price increases or decreases used to identify bull and bear phases, respectively. For example, if $\lambda_{\text{bull}} = 0.20$ (a 20% increase defines a bull phase), then $\lambda_{\text{bear}} = 0.20/(1+0.20) \approx 0.167$ (approximately a 16.7% decrease defines a bear phase). This constraint ensures that the thresholds are proportionally balanced in log-price space, preventing anomalies such as new highs during bear phases or new lows during bull phases. State transitions in the TD approach are determined by significant price reversals detected during the recursive partitioning process.

Statistical models, particularly Markov-switching models, provide a probabilistic framework for regime identification by modeling market states as a Markov process. HMM and Markov-switching models differ in their treatment of observations. Markov-switching model extends HMM by allowing observations to depend on both the current hidden state and their historical values, capturing autocorrelation in the data (Lu, Zeng, and Chen, 2010). For example, Maheu and McCurdy (2000) employ a duration-dependent Markov-switching model to identify bull and bear markets in stock returns. Markov-switching model allows for the flexibility of having multiple states, such as two-state models distinguishing bull and bear markets, or three-state models that include a neutral or sidewalk market to capture periods of stability or range-bound behavior. Liu and Wang (2017) apply a three-state hidden semi-Markov model (HSMM) to the CSI 300 index, identifying three regimes based on the sign and significance of mean returns: bear market (significantly negative returns, mean -0.51), sidewalk market (returns not significantly different from zero, mean -0.020), and bull market (significantly positive returns, mean 0.622). The sidewalk market captures periods of stability where prices fluctuate within a range, reflecting mean-reversion behavior common in volatile markets like the Chinese stock market. Similarly, Yuan and Gautam (2019) employ a three-state HMM for the FTSE 100 and Euro Stoxx 50, labeling the regimes as bull, bear, and neutral, with the neutral state representing periods of low directional movement. The number of states can be determined based on model selection criteria like AIC and BIC. Lihn (2017) finds that a

five-state HMM is preferred for the S&P 500 index, implying that market dynamics are better represented as a spectrum of return-volatility states rather than as a simple binary bull/bear split. In his interpretation, the five states comprise two low-volatility bullish states, one transition state, and two high-volatility bearish states, although the economic meaning of higher-dimensional state decompositions is less straightforward than in a simple bull/bear framework.

3.2 Implications

Rule-based methods typically identify two states (bull and bear), though hierarchical approaches like the TD method can reveal sub-phases within major regimes. Statistical models offer greater flexibility, with the number of states determined by data-driven criteria or specified based on considerations. Rule-based methods impose specific constraints to ensure logical coherence in regime identification. For instance, the Bry-Boschan algorithm and its adaptation by Pagan and Sossounov (2003) enforce alternation of peaks and troughs, while the TD approach prohibits intermediate extrema and incorporates a symmetry constraint. In statistical models, transitions are governed by a transition matrix, where probabilities are estimated from data and do not inherently forbid specific transitions unless explicitly modeled (e.g., setting certain probabilities to zero).

In conclusion, there is no universal consensus on the exact definitions or number of market regimes, nor on the specific constraints governing transitions between them. While the cited studies do not provide empirical evidence suggesting that specific transitions (e.g., bull to neutral or bull to bear) are forbidden, it is prudent to exercise caution when applying PMAP, as it may generate state sequences that do not align with empirical market behavior or the probabilistic structure of the model.

4 Simulation Study

This section examines decoder behavior under controlled HMM designs before turning to the CSI 300 application. The simulation study concentrates on three questions: how state dependence affects recovery, how emission overlap affects recovery, and how the hybrid weight parameter ω should be handled in practice.

In subsections 4.1 and 4.2 the data-generating parameters are treated as known, so those experiments isolate *decoding* difficulty rather than *estimation* difficulty. Both the latent state path and the observations are simulated from a known HMM, so the true hidden path is available by construction and can be compared directly with each decoded path. In subsection 4.3, by contrast, the model parameters are estimated separately for each generated sample using the `depmixS4` package in order to mimic the empirical application more closely.

Unless otherwise stated, the first two experiments use 500 Monte Carlo repetitions. To keep the hybrid decoder visible before the weight parameter is discussed formally, Tables 3 and 5 report a fixed reference case with $\omega = 0.50$; the interpretation and choice of ω are taken up explicitly only in subsection 4.3. Error rates are reported in percent, and the numbers in parentheses are Monte Carlo standard deviations across repetitions. In Tables 3 and 5, PMAP-Viterbi Diff. denotes the average percentage of time points at which the PMAP and Viterbi paths disagree, and PMAP Admissible Paths counts how many replications produce a PMAP path with no such transition. In the first two subsections, impossible-transition statistics are computed against the true transition matrix because the simulation design is known exactly.

4.1 Impact of State Dependence

The first experiment studies the dependence structure (regime persistence) of the hidden state sequence by varying the transition matrix while holding the emission distributions fixed. Here regime persistence means the tendency of the hidden chain to remain in the same state, so it is controlled mainly by the diagonal entries of the transition matrix. The sequence length is $n = 200$, the initial distribution is $\pi = (1/3, 1/3, 1/3)$, and the emission distributions are

$$N(-1, 3^2), N(0, 1^2), N(1, 2^2).$$

The four transition matrices are

$$P_{0.75} = \begin{bmatrix} 0.75 & 0.25 & 0 \\ 0 & 0.75 & 0.25 \\ 0.25 & 0 & 0.75 \end{bmatrix}, \quad P_{0.90} = \begin{bmatrix} 0.90 & 0.10 & 0 \\ 0 & 0.90 & 0.10 \\ 0.10 & 0 & 0.90 \end{bmatrix},$$

$$P_{0.95} = \begin{bmatrix} 0.95 & 0.05 & 0 \\ 0 & 0.95 & 0.05 \\ 0.05 & 0 & 0.95 \end{bmatrix}, \quad P_{\text{mixed}} = \begin{bmatrix} 0.75 & 0.25 & 0 \\ 0 & 0.90 & 0.10 \\ 0.05 & 0 & 0.95 \end{bmatrix}.$$

Some transition probabilities are set to zero in order to make the admissibility issue visible in the simulation. These zero entries are part of the experimental design and are not given an economic interpretation.

Table 3: Effect of state dependence on decoding performance. Error rates and impossible-transition rates are reported in %; Monte Carlo standard deviations are shown in parentheses. Pointwise differences are reported in percentage points.

Transition Setting	Viterbi Error	PMAP Error	Hybrid Error	PMAP-Viterbi Diff.	PMAP Admissible Paths
Diagonal 0.75	34.06 (5.65)	31.06 (5.02)	31.65 (5.43)	15.37	6/500
Diagonal 0.90	20.74 (6.75)	18.01 (5.61)	18.46 (6.12)	9.99	238/500
Diagonal 0.95	11.99 (6.49)	10.32 (5.18)	10.49 (5.44)	6.06	406/500
Mixed matrix	17.53 (6.71)	15.24 (5.46)	15.42 (5.75)	8.59	196/500

Higher dependence substantially improves recovery. Moving from diagonal value 0.75 to 0.95 reduces PMAP’s average error rate from 31.06% to 10.32%, and the same monotone improvement is visible for Viterbi and the hybrid decoder. The reason is straightforward: when the hidden chain is persistent, the transition structure itself carries useful information about the current regime.

Table 4: State-specific hybrid error rates in the dependence experiment. Error rates are reported in %; Monte Carlo standard deviations are shown in parentheses.

Transition Setting	State 1 Hybrid Error	State 3 Hybrid Error
Diagonal 0.75	38.91 (9.37)	35.56 (9.66)
Diagonal 0.90	22.82 (11.75)	21.70 (12.55)
Diagonal 0.95	14.68 (12.45)	14.00 (12.53)
Mixed matrix	47.90 (19.14)	9.11 (6.77)

Table 4 shows where the mixed-matrix result comes from. In that design, State 1 has persistence 0.75 while State 3 has persistence 0.95, and the hybrid state-specific error rates are 47.90% and 9.11%, respectively. By contrast, in the symmetric diagonal cases the State 1 and State 3 error rates move much closer together. This confirms that the harder recovery of State 1 in the mixed

case is directly tied to its lower own-state persistence.

The impossible-transition results highlight a second effect. At diagonal value 0.75, only 6 out of 500 PMAP paths are admissible for the whole sequence, whereas at 0.95 this rises to 406 out of 500. Thus stronger dependence improves both classification accuracy and the coherence of PMAP paths.

4.2 Impact of Emission Overlap

The second experiment varies the separation between emission distributions while holding the transition matrix fixed. The sequence length is $n = 200$, the initial distribution is $\pi = (1/3, 1/3, 1/3)$, and the transition matrix is

$$P = \begin{bmatrix} 0.75 & 0.25 & 0 \\ 0.20 & 0 & 0.80 \\ 0 & 0.70 & 0.30 \end{bmatrix}.$$

Again, some transition probabilities are set to zero in the design in order to expose potential inadmissibility in PMAP paths. The three emission configurations are

Low overlap: $N(-3, 3^2)$, $N(0, 1^2)$, $N(3, 2^2)$,

Medium overlap: $N(-2, 3^2)$, $N(0, 1^2)$, $N(2, 2^2)$,

High overlap: $N(-1, 3^2)$, $N(0, 1^2)$, $N(1, 2^2)$.

The overlap between two emission densities is defined in the code as the overlap coefficient

$$\text{OVL}(f_i, f_j) = \int \min\{f_i(x), f_j(x)\} dx,$$

and Figure 6 reports these pairwise overlap values for the three scenarios. Larger values indicate that the two states are harder to separate from the observations.

Table 5: Effect of emission overlap on decoding performance. Error rates and impossible-transition rates are reported in %; Monte Carlo standard deviations are shown in parentheses. Pointwise differences are reported in percentage points.

Overlap Level	Viterbi Error	PMAP Error	Hybrid Error	PMAP-Viterbi Diff.	PMAP Admissible Paths
Low	11.02 (3.41)	10.89 (3.20)	10.91 (3.30)	1.65	322/500
Medium	20.03 (4.35)	19.40 (4.00)	19.54 (4.18)	4.59	54/500
High	31.48 (5.61)	30.26 (5.10)	30.57 (5.41)	9.36	12/500

Within this design, increasing overlap is the strongest source of decoding difficulty. As the emission distributions move closer together, all three decoders become less accurate, the disagreement between PMAP and Viterbi rises from 1.65 to 9.36 percentage points, and the number of admissible PMAP paths falls from 322 to 12 out of 500. This confirms that overlap affects not only pointwise accuracy but also the coherence of locally decoded paths.

4.3 Choice of the Weight Parameter ω

The final simulation addresses a practical question rather than a search for a universally optimal ω . The purpose is not simply to illustrate how the hybrid decoder moves from PMAP to Viterbi as ω increases, because that behavior can already be studied when the true model parameters are known. Instead, this subsection asks what happens when ω is varied in a setting closer to the empirical application, where the HMM parameters are first estimated from the data and decoding is then performed with those estimated parameters.

This distinction matters because two transition matrices are involved. The data are generated from a *true* transition matrix that contains zero restrictions, but the HMM is then fitted *without* imposing those restrictions. As a result, the fitted transition matrix might be fully positive, so under the fitted model all transitions are admissible. Consequently, the transition diagnostic reported below should not be interpreted as counting inadmissible moves under the fitted HMM. Instead, it counts decoded transitions that violate the zero restrictions in the true data-generating matrix. The diagnostic is therefore a simulation-based way to assess how far the decoded path, obtained from estimated parameters, drifts away from the true transition structure that generated the data.

This subsection therefore differs from the previous two because the model parameters are estimated separately for each generated sample using the `depmixS4` package. The experiment uses $n = 1000$, 200 repetitions, initial distribution $\pi = (1/3, 1/3, 1/3)$, the diagonal-0.90 transition design from subsection 4.1,

$$P = \begin{bmatrix} 0.90 & 0.10 & 0 \\ 0 & 0.90 & 0.10 \\ 0.10 & 0 & 0.90 \end{bmatrix},$$

and the same emission distributions as subsection 4.1,

$$N(-1, 3^2), \quad N(0, 1^2), \quad N(1, 2^2).$$

The convention used throughout the thesis is fixed as follows: $\omega = 0$ corresponds to PMAP and $\omega = 1$ corresponds to Viterbi. Thus increasing ω shifts the hybrid decoder away from PMAP and toward Viterbi.

The estimation step is intentionally unrestricted: the fitting procedure does not force the estimated transition matrix to retain the zero entries of the true matrix. If the transition diagnostic were evaluated against the fitted matrix, the reported impossible transition percentages would be mechanically zero for Viterbi decoders. For this reason, Table 6 evaluates decoded transitions against the known zero restrictions in the true data-generating matrix instead.

Table 6 reports representative values from the full grid $\omega \in \{0, 0.05, \dots, 1\}$ generated by the code in Appendix A.

Table 6: Representative results from the hybrid-weight experiment. Error rates are reported in %; Monte Carlo standard deviations are shown in parentheses. Impossible transitions are evaluated against the true transition restrictions.

Metric	$\omega = 0.00$	$\omega = 0.10$	$\omega = 0.50$	$\omega = 1.00$
Overall error	21.12 (11.53)	21.14 (11.54)	21.37 (11.44)	23.33 (11.00)
State 1 error	24.22 (14.12)	24.48 (14.12)	25.31 (14.01)	26.85 (14.41)
State 3 error	25.01 (15.03)	25.06 (15.09)	25.37 (15.03)	26.75 (15.71)
Impossible transitions	1.21 (2.15)	0.87 (2.20)	0.72 (2.20)	0.58 (1.90)

The result is that classification accuracy changes very little as ω varies. Moving from PMAP to $\omega = 0.10$ changes the mean overall error only from 21.12% to 21.14%, and even at $\omega = 1$ the deterioration remains moderate. The state-specific error rates for States 1 and 3 also rise only gradually as ω moves toward Viterbi.

The impossible-transition rate falls from 1.21% at $\omega = 0$ to 0.87% at $\omega = 0.10$ and to 0.58% at $\omega = 1.00$. This means that small positive weights leave pointwise accuracy almost unchanged while modestly reducing the frequency of decoded moves that conflict with the true zero restrictions. The interpretation requires care: the nonzero value at $\omega = 1.00$ does not mean that the Viterbi path is inadmissible under the fitted HMM. Viterbi respects the fitted transition matrix by construction. The positive percentage arises because the fitted matrix, estimated without zero constraints, can assign positive probability to moves that are zero in the true data-generating matrix. Hence the table measures disagreement with the true transition structure, not inadmissibility under the fitted model.

This relates to the more recent discussion in Bæk et al. (2025), who introduce Artemis analysis and the Artemis plot as a simulation-based way to study how the hybrid tuning parameter trades

off the pointwise accuracy against log-joint probability. The present thesis does not attempt that optimization exercise and therefore makes no claim that $\omega = 0.10$ is optimal in any universal sense. For the empirical application below, $\omega = 0.10$ is therefore used as a PMAP-leaning hybrid choice. The justification is modest rather than universal: in this re-estimation setting, $\omega = 0.10$ stays very close to PMAP in pointwise accuracy while introducing a small global-path component and slightly reducing disagreement with the true transition restrictions.

4.4 Implications for the Empirical Analysis

The simulation study yields three implications for Chapter 5.

First, decoder disagreement is largest when regimes overlap strongly and state persistence is weak. Second, the state-dependence experiment shows weak persistence makes the same overlap much more problematic. The empirical application should therefore interpret the estimated emission distributions and the transition matrix jointly rather than separately.

Third, the weight experiment does not reveal a sharp data-driven optimum for ω . Instead, it shows that small positive weights can reduce the frequency of decoded moves that violate the true transition restrictions while leaving pointwise accuracy almost unchanged. The empirical use of a small positive hybrid weight should therefore be interpreted as a conservative choice rather than as a universally necessary correction.

These simulations are not a full finite-sample estimation study. Once model parameters must be estimated from real data, sample size and numerical fitting uncertainty can matter materially.

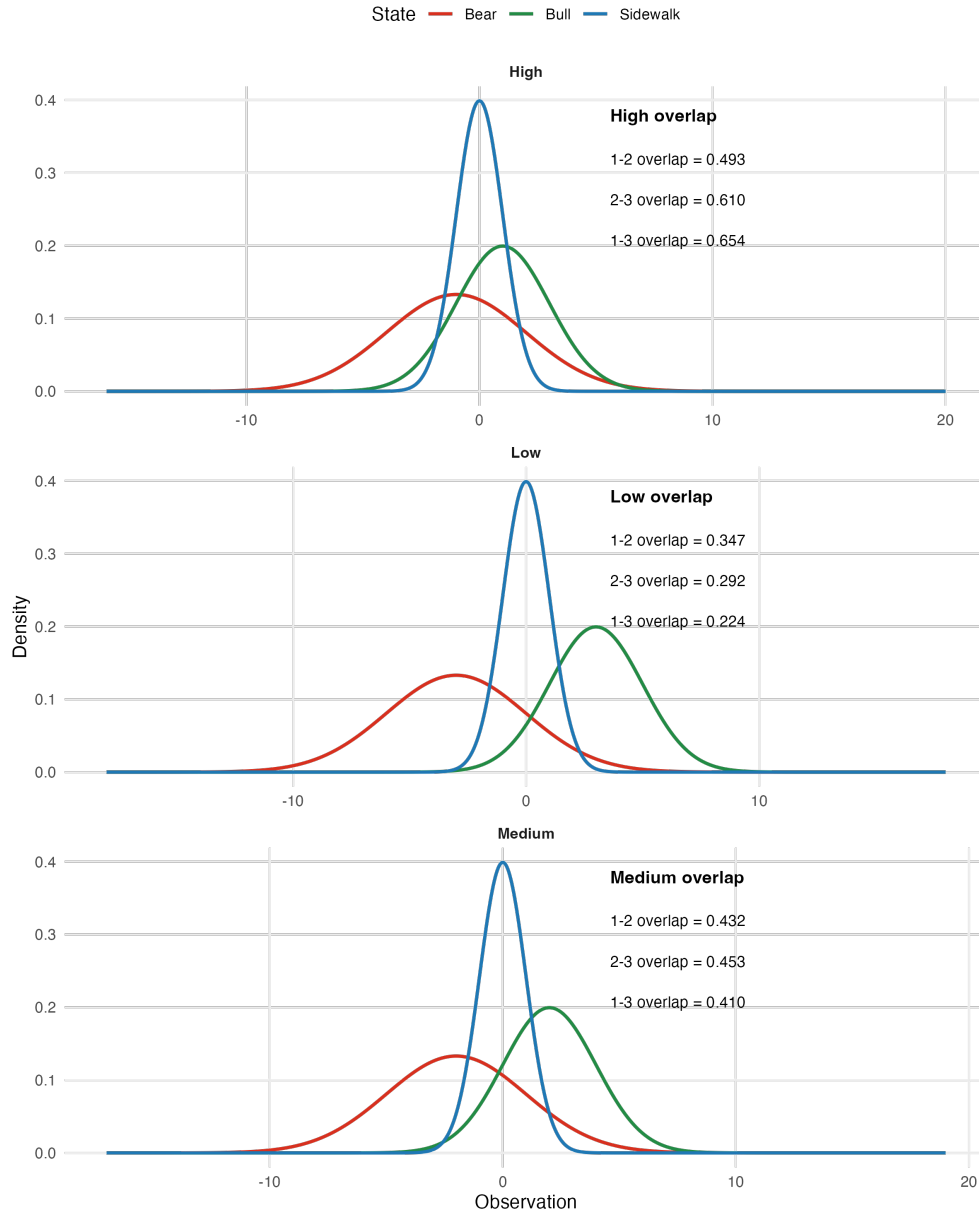


Figure 6: Emission distributions used in the overlap experiment. Larger overlap makes the hidden states harder to distinguish.

5 Application to CSI 300

This chapter applies the three-state HMM with Gaussian emissions to the CSI 300 index at weekly and monthly frequencies. The chapter has two aims. Application 1 studies regime identification at both frequencies and compares the resulting decoder classifications. Application 2 then uses a monthly expanding-window validation exercise to assess whether the monthly decoded regimes translate into economically meaningful trading signals.

Liu and Wang (2017) analyze daily CSI 300 returns with a three-state hidden semi-Markov model, interpret the states as bear, sidewalk, and bull markets, and use a recursive trading strategy as an economic plausibility check. In their setting, profitability is driven mainly by short positions in bear periods, that is, positions that profit when prices fall because the asset is sold first and repurchased later at a lower price. The present chapter differs in two ways: it uses a simpler HMM with Gaussian emissions rather than a HSMM, and it works with aggregated weekly and fixed-interval monthly returns rather than daily data.

5.1 Empirical Setup

The sample covers CSI 300 closing prices from April 8, 2005 to May 13, 2025. In Chapter 2, the weekly and monthly series were constructed as h -period log returns with $h = 5$ trading days for the weekly data and $h = 22$ trading days for the monthly data. Thus the weekly window length is one fixed 5-trading-day interval, whereas the monthly window length is one fixed 22-trading-day interval rather than an exact calendar month. This return construction yields 1017 weekly observations and 238 monthly observations after omitting the undefined first return at each frequency.

For each frequency, a three-state HMM is estimated using the `depmixS4` package in R. The fitted model uses Gaussian state-dependent distributions and a homogeneous transition matrix. After estimation, the states are relabeled by ascending estimated mean return so that State 1 is interpreted as *bear*, State 2 as *sidewalk*, and State 3 as *bull*. Viterbi, PMAP, and hybrid paths are then computed from the fitted model. The hybrid decoder uses $\omega = 0.10$, where $\omega = 0$ corresponds to PMAP and $\omega = 1$ corresponds to Viterbi.

The role of hybrid decoding differs across the two frequencies. In the weekly model, the fitted transition matrix has no exact zero restrictions, so hybrid decoding is mainly a robustness comparison against PMAP and Viterbi. In the monthly model, by contrast, the fitted transition matrix contains zero entries, so hybrid decoding is more relevant because it preserves admissibility while remaining close to PMAP.

The trading-validation design therefore follows the recursive expanding-window logic of Liu and Wang (2017). For the monthly validation, the initial estimation window contains 102 monthly returns, running from 2005-07-01 to 2013-12-01, and the remaining 136 monthly returns from 2014-01-01 to 2025-04-01 form the testing sample. For each forecast origin in that testing sample, the model is re-estimated on all data available up to that date, the last decoded state in the window is recorded, and that state determines the position taken over the next return interval. The weekly model is not taken forward as a trading exercise, because its role in this chapter is diagnostic rather than predictive. The underlying R scripts and the validation-summary code are included in the appendices.

5.2 Application 1: Regime Identification

5.2.1 Weekly Return Analysis

Table 7 reports the estimated parameters of the weekly model.

Table 7: Estimated HMM parameters for CSI 300 weekly returns. Means and standard deviations are reported in %.

Parameter	Bear (State 1)	Sidewalk (State 2)	Bull (State 3)
Mean (μ_i)	-0.31	0.31	1.63
Std. Dev. (σ_i)	2.15	5.05	1.15

The estimated transition matrix is

$$P = \begin{bmatrix} 0.8683 & 0.0002 & 0.1315 \\ 0.0151 & 0.9358 & 0.0490 \\ 0.5337 & 0.1794 & 0.2869 \end{bmatrix}. \quad (1)$$

The weekly model produces one low-return state and one high-return state, but the empirical separation is weak. The central difficulty is not only that the estimated means are relatively close; it is also that the sidewalk state has very high volatility and therefore overlaps strongly with both the bear and bull states. In addition, the fitted transition matrix is economically awkward: the estimated bull state has only a 28.69% probability of remaining in the same state and a 53.37% probability of moving directly to the bear state in the next week. This makes the weekly bull regime unstable even before any decoder is chosen.

Figure 7 makes the problem visible. The estimated weekly emission distributions overlap heavily, especially because the sidewalk state is much more volatile than the other two.

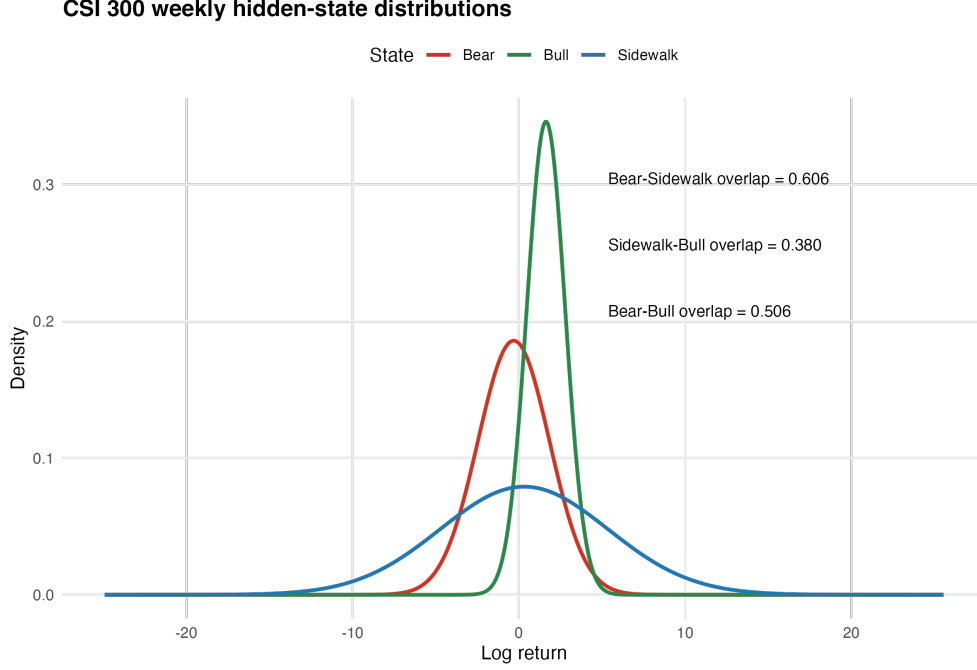


Figure 7: Estimated emission distributions for CSI 300 weekly returns.

Table 8 summarizes the hybrid-decoded weekly regimes. In the regime-summary tables, positive and negative return frequency refer to the share of returns above and below zero within a decoded regime, while sample mean, sample standard deviation, and sample size are computed from the returns assigned to that regime.

Table 8: Summary statistics by hybrid-decoded regime for CSI 300 weekly returns. Frequencies are reported in %; returns are reported in %.

Metric	Bear	Sidewalk	Bull
Positive return frequency	50.80	54.75	100.00
Negative return frequency	49.20	45.25	0.00
Sample size	626	358	33
Sample mean	-0.07	0.36	1.93
Sample Std. Dev.	2.11	5.09	0.76

The economic interpretation is weak. The weekly bull state is attractive in sign and volatility, but it is rare, with only 33 observations. The weekly bear state is much less convincing: positive and negative weeks are almost evenly split, and its sample mean is only slightly negative. Decoder disagreement is also material. The pointwise differences are 7.08% between Viterbi and PMAP, 7.28% between Viterbi and hybrid, and 1.08% between PMAP and hybrid. Taken together, these results suggest that the three-state HMM with Gaussian emissions does not provide a very stable economic regime classification at the weekly frequency.

5.2.2 Monthly Return Analysis

Table 9 reports the estimated parameters of the monthly model.

Table 9: Estimated HMM parameters for CSI 300 monthly returns. Means and standard deviations are reported in %.

Parameter	Bear (State 1)	Sidewalk (State 2)	Bull (State 3)
Mean (μ_i)	-6.86	-0.35	8.55
Std. Dev. (σ_i)	11.73	4.77	7.08

The estimated transition matrix is

$$P = \begin{bmatrix} 0.8018 & 0 & 0.1982 \\ 0 & 0.9590 & 0.0410 \\ 0.1253 & 0.1355 & 0.7392 \end{bmatrix}. \quad (2)$$

Relative to the weekly model, the monthly specification is easier to interpret at first sight. The bear and bull states are much farther apart in mean return, the sidewalk state has lower volatility than the bear state, and PMAP and hybrid now coincide exactly in the full sample. The pointwise differences are 7.14% between Viterbi and PMAP, 7.14% between Viterbi and hybrid, and 0.00% between PMAP and hybrid.

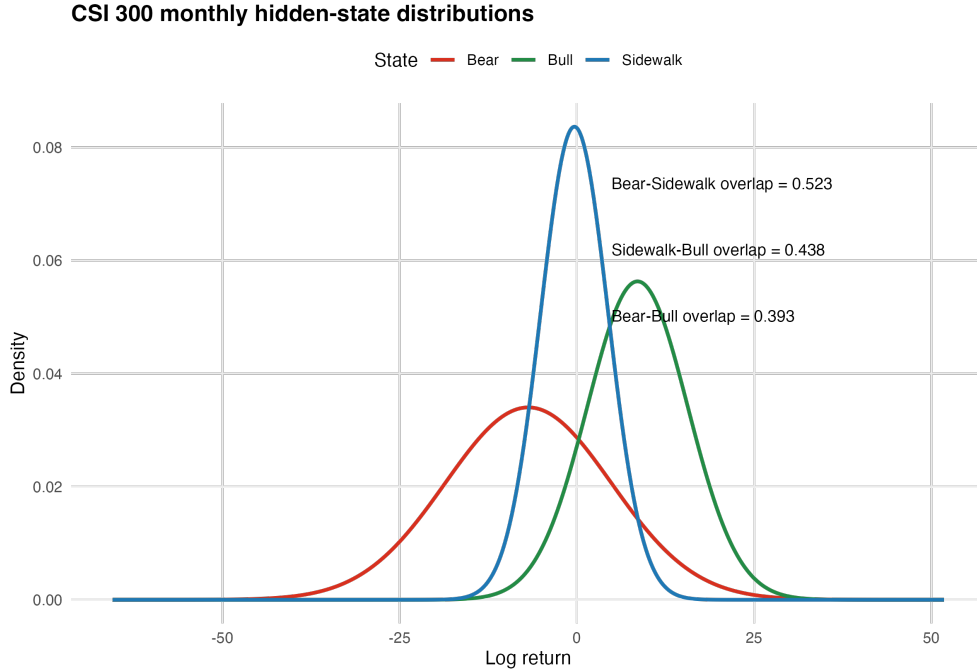


Figure 8: Estimated emission distributions for CSI 300 monthly returns.

Figure 8 shows that overlap is still present, but it is materially weaker than in the weekly model.

Two estimated transition probabilities are exactly zero, so the fitted monthly model does not allow direct bear-to-sidewalk or sidewalk-to-bear moves. These zeros should not be interpreted as literal economic impossibilities. They are better read as artifacts of the fitted three-state HMM with Gaussian emissions in a finite sample. Their importance is therefore mainly diagnostic: they reveal that the fitted model is imposing a fairly rigid regime structure. The diagonal is also informative: $p_{11} = 0.8018$, $p_{22} = 0.9590$, and $p_{33} = 0.7392$ are all large, especially the sidewalk persistence. This helps explain the long uninterrupted sidewalk blocks in Figure 9 and foreshadows why the later-sample classification becomes so heavily concentrated in the sidewalk regime.

Table 10 summarizes the hybrid-decoded monthly regimes.

Table 10: Summary statistics by hybrid-decoded regime for CSI 300 monthly returns. Frequencies are reported in %; returns are reported in %.

Metric	Bear	Sidewalk	Bull
Positive return frequency	34.48	46.88	93.88
Negative return frequency	65.52	53.12	6.12
Sample size	29	160	49
Sample mean	−8.07	−0.45	9.22
Sample Std. Dev.	11.20	4.63	6.68

The monthly regimes are economically more plausible than the weekly ones. The bear state is dominated by negative months, the bull state is dominated by positive months, and the sidewalk state captures long stretches of weak market direction. At the same time, the later part of the sample reveals an important limitation: from 2016 onward, 107 of 112 monthly observations are classified as sidewalk, while only 2 are classified as bear and 3 as bull. Before 2016, the corresponding counts are 27 bear, 53 sidewalk, and 46 bull. Thus the later classification becomes too coarse to track all visually apparent trend changes in the index level.

Figure 9 shows much longer and less fragmented state blocks while keeping the price trajectory visible. In the earlier years, the monthly model produces extended bear, sidewalk, and bull episodes that align reasonably well with broad movements in the CSI 300 index. In the later years, however, the plot also makes clear that the model relies heavily on the sidewalk state, even across periods where the price level trends visibly upward or downward. This is not necessarily a contradiction, because the HMM is fitted to one-period monthly returns rather than to the price trend itself, but it does show that the later-sample monthly classification is too coarse for a fully satisfactory market-regime interpretation.

Figure 10 helps explain why the later monthly classification becomes so sidewalk-dominated. Before 2016, the monthly returns are much more dispersed and the fitted normal benchmark is visibly

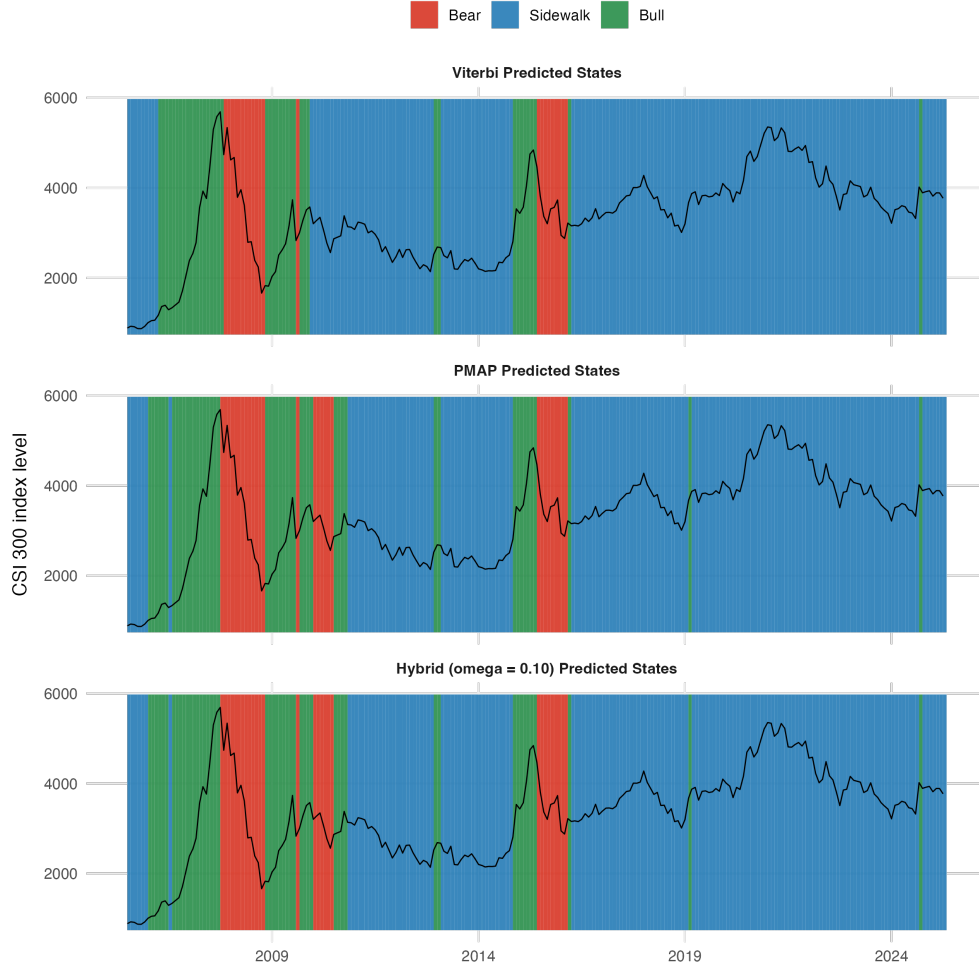


Figure 9: Hybrid-decoded market regimes for the full CSI 300 monthly sample, shown as regime bands behind the CSI 300 index level.

wider, so large positive and negative observations occur often enough for the bear and bull regimes to appear clearly. From 2016 onward, the monthly returns are far more concentrated near zero, with a sharper center and fewer extreme observations.

5.3 Application 2: Monthly Trading Validation

The trading exercise follows the recursive logic of Liu and Wang (2017), but with one adaptation. Their paper uses global decoding in a three-state HSMM. Here the same expanding-window trading design is combined with the hybrid HMM path for the monthly model only.

The sample is split into two parts using December 2013 as the last training period. This gives an initial estimation window of 102 monthly returns and a testing sample of 136 monthly returns. The strategy is implemented as follows.

1. Fix the start of the estimation window at the first observation in the sample.

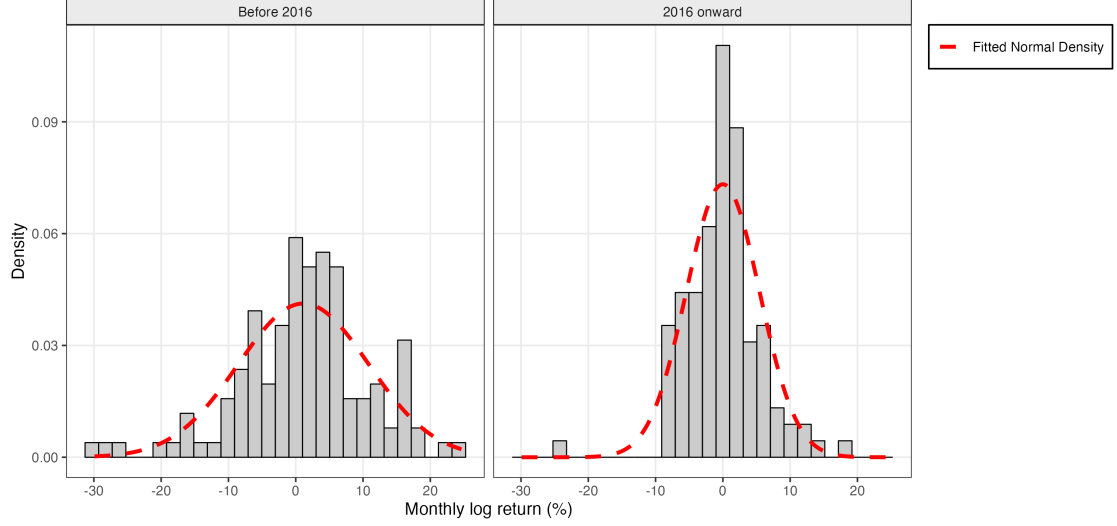


Figure 10: Monthly CSI 300 return histograms before and after 2016 with fitted normal densities.

2. Set the initial end of the window at the last observation in the training sample.
3. Estimate the three-state HMM on all observations in the current window.
4. Decode the entire window with the hybrid path and record only the last decoded state.
5. Convert that last decoded state into the trading signal applied over the next return interval:

$$s_t = \begin{cases} -1, & \text{if the last decoded state is bear,} \\ 0, & \text{if the last decoded state is sidewalk,} \\ 1, & \text{if the last decoded state is bull.} \end{cases}$$

6. Move the end of the window forward by one observation and repeat the procedure until the end of the testing sample.

This is a *filtered* rule because the signal for period $t + 1$ uses only the information available at the end of period t . By contrast, a decoded path produced from the full sample uses future observations and is therefore not interpreted here as a tradable rule.

Let

$$r_{t+1} = \log \left(\frac{S_{t+1}}{S_t} \right)$$

denote the observed next-period log return in decimal form, so that $r_{t+1} = 0.05$ corresponds to a 5% log return rather than to the percentage-scaled value 5. This distinction matters because the descriptive tables in this chapter report returns in percent, but the trading validation uses the underlying decimal log returns in the cumulative-return calculations.

The corresponding one-period simple return is

$$R_{t+1} = \exp(r_{t+1}) - 1.$$

This conversion is needed because the trading signal $s_t \in \{-1, 0, 1\}$ represents a portfolio position: long in a bull state, zero exposure in a sidewalk state, and short in a bear state. The one-period strategy return is therefore defined as

$$R_{t+1}^{\text{strat}} = s_t R_{t+1}.$$

This makes economic sense because it maps the decoded state directly into the next-period payoff. When $s_t = 1$, the strategy earns the market return; when $s_t = 0$, it earns zero; and when $s_t = -1$, a negative market return produces a positive strategy return.

The sole performance measure is cumulative return. After m testing periods, it is computed as

$$\text{CR}_m = \prod_{j=1}^m (1 + R_j^{\text{strat}}) - 1.$$

This definition is natural because simple returns compound multiplicatively over time. For comparison, the buy-and-hold benchmark keeps the market fully invested in every testing month, so

$$\text{CR}_m^{\text{BH}} = \prod_{j=1}^m (1 + R_j) - 1 = \exp\left(\sum_{j=1}^m r_j\right) - 1.$$

This validation abstracts from transaction costs, financing costs, and practical short-selling frictions. The reported returns should therefore be interpreted only as model-based diagnostics.

Two short examples clarify the construction. In the monthly validation file, the window ending on 2015-06-01 is classified as bull, so $s_t = 1$ for 2015-07-01. The next monthly log return is -0.1587 , which corresponds to a simple return of $\exp(-0.1587) - 1 \approx -14.7\%$. Because the signal is long, the strategy return is also about -14.7% . Earlier, from 2014-01-01 to 2014-12-01, every monthly signal is sidewalk, so $s_t = 0$ throughout that year: the strategy remains flat while monthly buy-and-hold rises strongly. The interpretation is straightforward: the filtered rule avoids exposure when the model is cautious, but it also gives up long upward trends.

Table 11 reports the monthly validation results.

The monthly validation is conservative. Its cumulative return is about -3.2% , compared with about 61.8% for monthly buy-and-hold. The main reason is visible directly in Table 11: out of 136 testing

Table 11: Monthly expanding-window validation results for the hybrid-decoded trading strategy. Cumulative returns are reported for the testing sample only.

Test Months	Strategy CR	Buy-and-hold CR	Bear	Sidewalk	Bull
136	-3.2%	61.8%	7	118	11

months, 118 are classified as sidewalk, so the strategy is out of the market most of the time, and the few short signals are not strong enough to offset the missed long exposure.

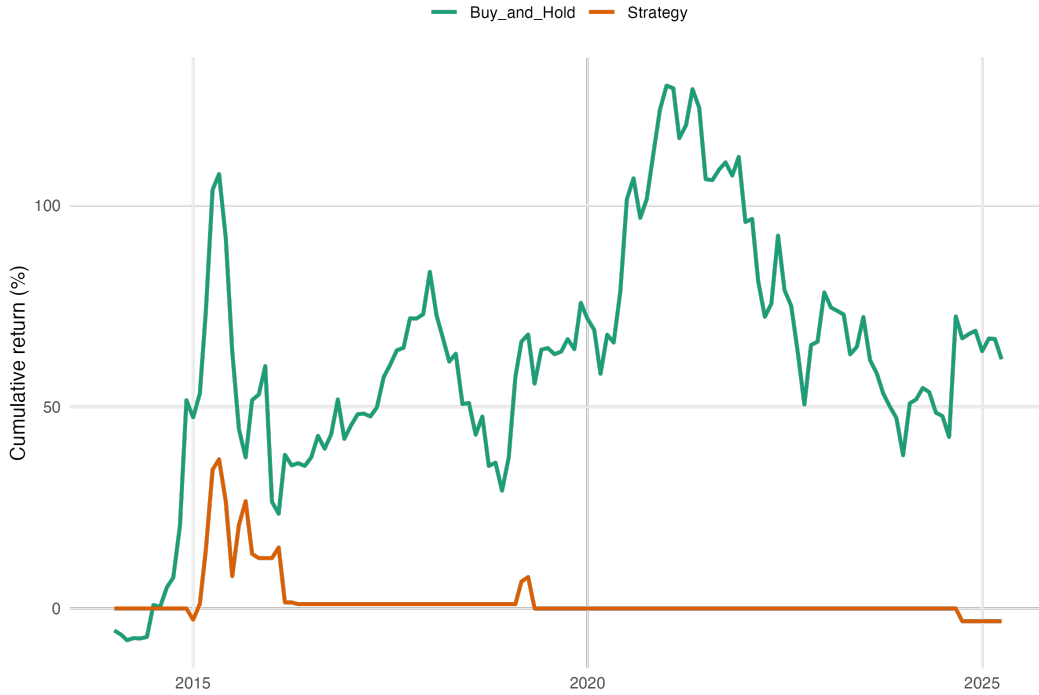


Figure 11: Testing-sample cumulative returns of the monthly expanding-window hybrid strategy and monthly buy-and-hold.

Figure 11 visualizes the monthly result. The expanding-window strategy is transparent and fully out-of-sample, but it remains too cautious to match buy-and-hold. This is different from Liu and Wang (2017), where profitability comes mainly from sustained short exposure during bear phases in a daily HSMM setting. In the present HMM application, the monthly rule remains far too cautious and ends slightly negative, so it underperforms badly. The trading section should therefore be read as a validation exercise rather than as evidence of a strong exploitable trading rule at the monthly frequency.

5.4 Discussion

The CSI 300 application yields three main conclusions.

First, the weekly and monthly HMMs are not equally informative, and neither fit is fully satisfactory. The weekly model suffers from heavy overlap and economically awkward transition structure. The monthly model produces a clearer bear-sidewalk-bull interpretation overall, but that advantage is concentrated mainly in the earlier part of the sample. In the post-2016 period, the monthly decoder relies too heavily on the sidewalk state, so its regime map becomes much less informative than the early-sample figure might suggest.

Second, hybrid decoding is not the main source of success or failure in this chapter. In the weekly model it serves mainly as a comparison, because the fitted transition matrix has no exact zero entries. In the monthly model, a small positive hybrid weight remains methodologically justified because the fitted matrix contains zeros, but hybrid decoding cannot solve the more basic problem that the fitted monthly model becomes too coarse in the later sample.

Third, the monthly trading evidence should be interpreted honestly. The expanding-window monthly strategy ends slightly negative and falls far short of passive monthly buy-and-hold. The main empirical contribution of the chapter is therefore diagnostic rather than predictive: for the CSI 300, a simple three-state HMM with Gaussian emissions is only partly adequate, and the adequacy of the fitted model is more consequential than the decoder comparison in this application.

References

- Ang, A. and A. Timmermann (2011). *Regime Changes and Financial Markets*. Working Paper 17182. National Bureau of Economic Research.
- Apergis, N. et al. (2019). “Decoding the Australian electricity market: New evidence from three-regime hidden semi-Markov model”. In: *Energy Economics* 78, pp. 129–142.
- Bishop, C. M. (2006). *Pattern recognition and machine learning*. New York: Springer.
- Bry, G. and C. Boschan (1971). *Cyclical analysis of time series: Selected procedures and computer programs*. National Bureau of Economic Research.
- Bæk, Z. E. D. et al. (2025). “Advanced posterior analyses of hidden Markov models: finite Markov chain imbedding and hybrid decoding”. In: *arXiv preprint arXiv:2504.15156*.
- Cont, R. (2001). “Empirical properties of asset returns: Stylized facts and statistical issues”. In: *Quantitative Finance* 1, pp. 223–236.
- Fama, E. F. (1970). “Efficient capital markets: A review of theory and empirical work”. In: *The Journal of Finance* 25, pp. 383–417.
- Green, E. (Oct. 2014). “Analysis and Modelling of Financial Logarithmic Return Data using Multifractal and Agent-Based Techniques”. PhD thesis. Maynooth, Ireland: National University of Ireland Maynooth.
- Hamilton, J. D. (1989). “A new approach to the economic analysis of nonstationary time series and the business cycle”. In: *Econometrica* 57, pp. 357–384.
- Hanna, A. J. (2018). “A top-down approach to identifying bull and bear market states”. In: *International Review of Financial Analysis* 55, pp. 93–110.
- Lember, J. and A. Koloydenko (2014). “Bridging Viterbi and posterior decoding: a generalized risk approach to hidden path inference based on hidden Markov models”. In: *Journal of Machine Learning Research* 15, pp. 1–58.
- LeRoy, S. F. (1989). “Efficient capital markets and martingales”. In: *Journal of Economic Literature* 27, pp. 1583–1621.
- Lihn, S. H. (2017). “Hidden Markov Model for Financial Time Series and Its Application to S&P 500 Index”. In: *Quantitative Finance*, pp. 1–20.
- Liu, Z. and S. Wang (2017). “Decoding Chinese stock market returns: Three-state hidden semi-Markov model”. In: *Pacific-Basin Finance Journal* 44, pp. 127–149.

- Lu, H.-M., D. Zeng, and H. Chen (2010). “Prospective Infectious Disease Outbreak Detection Using Markov Switching Models”. In: *IEEE Transactions on Knowledge and Data Engineering* 22, pp. 565–577.
- Maheu, J. M. and T. H. McCurdy (2000). “Identifying bull and bear markets in stock returns”. In: *Journal of Business & Economic Statistics* 18, pp. 100–112.
- McNeil, A. J., R. Frey, and P. Embrechts (2015). *Quantitative risk management: Concepts, techniques and tools*. revised. Princeton: Princeton University Press.
- McQuire, P. and A. Kume (2023). *R Programming for Actuarial Science*. 1st ed. West Sussex: John Wiley & Sons Ltd.
- Nyberg, H. (2013). “Predicting bear and bull stock markets with dynamic binary time series models”. In: *Journal of Banking & Finance* 37, pp. 3351–3363.
- Pagan, A. and K. Sossounov (2003). “A simple framework for analysing bull and bear markets”. In: *Journal of Applied Econometrics* 18, pp. 23–46.
- Rabiner, L. R. (1989). “A tutorial on hidden Markov models and selected applications in speech recognition”. In: *Proceedings of the IEEE* 77, pp. 257–286.
- Yuan, Y. and M. Gautam (2019). “Market regime identification using hidden Markov models”. In.
- Zegadło, P. (2022). “Identifying bull and bear market regimes with a robust rule-based method”. In: *Research in International Business and Finance* 60, pp. 1–22.
- Zucchini, W., I. L. MacDonald, and R. Langrock (2016). *Hidden Markov models for time series: An introduction using R*. 2nd ed. Boca Raton: Chapman and Hall/CRC.

A Appendix A: Simulation Code

This appendix contains the compact R scripts used to reproduce the Chapter 4 simulation outputs. To avoid repeating the same decoder logic in multiple listings, the appendix shows one shared utility file and one simulation driver script. The driver produces both the overall decoder summaries and the state-specific error comparisons reported in the chapter, and in the weight experiment it evaluates inadmissible moves against the known data-generating transition restrictions.

A.1 Shared HMM Utilities

```
get_script_path <- function() {
  args <- commandArgs(trailingOnly = FALSE)
  file_arg <- grep("^--file=", args, value = TRUE)
  decode_script_arg <- function(path) {
    normalizePath(gsub("~\\+", " ", path), mustWork = FALSE)
  }
  if (length(file_arg) > 0) {
    return(decode_script_arg(sub("^--file=", "", file_arg[1])))
  }

  for (frame in rev(sys.frames())) {
    if (!is.null(frame$ofile)) {
      return(normalizePath(frame$ofile, mustWork = FALSE))
    }
  }

  normalizePath(getwd(), mustWork = FALSE)
}

project_paths <- function() {
  script_path <- get_script_path()
  script_dir <- if (grepl("\\.R$", basename(script_path))) dirname(script_path) else script_path
  thesis_dir <- normalizePath(file.path(script_dir, ".."), mustWork = TRUE)
  workspace_dir <- normalizePath(file.path(thesis_dir, ".."), mustWork = TRUE)

  list(
    script_dir = script_dir,
    thesis_dir = thesis_dir,
    workspace_dir = workspace_dir,
    code_dir = file.path(workspace_dir, "thesis code")
  )
}

ensure_dir <- function(path) {
  if (!dir.exists(path)) {
```

```

    dir.create(path, recursive = TRUE, showWarnings = FALSE)
  }
}

write_csv_file <- function(data, path, row.names = FALSE) {
  ensure_dir(dirname(path))
  write.csv(data, path, row.names = row.names)
}

save_plot_multiple <- function(plot, paths, width, height, dpi = 300) {
  for (path in unique(paths)) {
    ensure_dir(dirname(path))
    ggplot2::ggsave(filename = path, plot = plot, width = width, height = height, dpi = dpi)
  }
}

state_labels <- c("Bear", "Sidewalk", "Bull")
state_colors <- c("Bear" = "#d7301f", "Sidewalk" = "#1f78b4", "Bull" = "#238b45")

simulate_hmm <- function(n, means, sds, transition_matrix, initial_probs) {
  nstates <- length(means)
  states <- integer(n)
  states[1] <- sample.int(nstates, size = 1, prob = initial_probs)

  if (n > 1) {
    for (t in 2:n) {
      states[t] <- sample.int(nstates, size = 1, prob = transition_matrix[states[t - 1], ])
    }
  }

  observations <- rnorm(n, mean = means[states], sd = sds[states])
  data.frame(state = states, observation = observations)
}

overlap_area <- function(mu1, sigma1, mu2, sigma2) {
  x <- seq(
    min(mu1 - 4 * sigma1, mu2 - 4 * sigma2),
    max(mu1 + 4 * sigma1, mu2 + 4 * sigma2),
    length.out = 1000
  )
  density_1 <- dnorm(x, mean = mu1, sd = sigma1)
  density_2 <- dnorm(x, mean = mu2, sd = sigma2)
  sum(pmin(density_1, density_2)) * (x[2] - x[1])
}

emission_matrix <- function(observations, means, sds) {
  sapply(seq_along(means), function(idx) {
    dnorm(observations, mean = means[idx], sd = sds[idx])
  })
}

```

```

    })
  }

forward_scaled <- function(observations, transition_matrix, initial_probs, emissions) {
  nstates <- nrow(transition_matrix)
  n <- length(observations)

  if (n == 1) {
    emissions <- matrix(emissions, nrow = 1)
  }

  phi <- matrix(initial_probs, nrow = 1)
  alpha <- matrix(NA_real_, nrow = n, ncol = nstates)
  scale <- numeric(n)

  for (t in seq_len(n)) {
    if (t > 1) {
      phi <- phi %*% transition_matrix
    }
    phi <- phi %*% diag(emissions[t, ])
    sumphi <- sum(phi)
    phi <- phi / sumphi
    scale[t] <- sumphi
    alpha[t, ] <- phi
  }

  list(alpha = alpha, scale = scale)
}

backward_scaled <- function(observations, transition_matrix, emissions, last_row, scale) {
  nstates <- nrow(transition_matrix)
  n <- length(observations)
  beta <- matrix(NA_real_, nrow = n, ncol = nstates)
  beta[n, ] <- last_row
  phi <- matrix(last_row, ncol = 1)

  if (n > 1) {
    for (t in seq(n - 1, 1, -1)) {
      phi <- transition_matrix %*% diag(emissions[t + 1, ]) %*% phi
      phi <- phi / scale[t + 1]
      beta[t, ] <- phi
    }
  }

  beta
}

posterior_probabilities <- function(observations, transition_matrix, initial_probs, emissions) {

```

```

forward_res <- forward_scaled(observations, transition_matrix, initial_probs, emissions)
beta <- backward_scaled(
  observations,
  transition_matrix,
  emissions,
  last_row = rep(1, nrow(transition_matrix)),
  scale = forward_res$scale
)
posterior <- forward_res$alpha * beta
posterior / rowSums(posterior)
}

viterbi_path <- function(observations, transition_matrix, initial_probs, emissions) {
  n <- length(observations)
  nstates <- nrow(transition_matrix)
  score <- matrix(-Inf, nrow = n, ncol = nstates)
  path <- integer(n)

  score[1, ] <- log(initial_probs) + log(emissions[1, ])

  if (n > 1) {
    log_transition <- log(transition_matrix)
    for (t in 2:n) {
      previous <- matrix(score[t - 1, ], nrow = nstates, ncol = nstates)
      score[t, ] <- apply(previous + log_transition, 2, max) + log(emissions[t, ])
    }
    path[n] <- which.max(score[n, ])
    for (t in seq(n - 1, 1, -1)) {
      path[t] <- which.max(log_transition[, path[t + 1]] + score[t, ])
    }
  } else {
    path[1] <- which.max(score[1, ])
  }

  path
}

pmap_path <- function(observations, transition_matrix, initial_probs, emissions) {
  posterior <- posterior_probabilities(observations, transition_matrix, initial_probs, emissions)
  apply(posterior, 1, which.max)
}

safe_weighted_log <- function(probabilities, weight) {
  if (weight == 0) {
    return(rep(0, length(probabilities)))
  }
  weight * log(probabilities)
}

```

```

hybrid_path <- function(observations, transition_matrix, initial_probs, emissions, omega = 0.5) {
  n <- length(observations)
  nstates <- nrow(transition_matrix)
  posterior <- posterior_probabilities(observations, transition_matrix, initial_probs, emissions)
  score <- matrix(-Inf, nrow = n, ncol = nstates)
  backpointer <- matrix(0L, nrow = n, ncol = nstates)
  path <- integer(n)

  posterior_weight <- 1 - omega
  transition_weight <- omega

  score[1, ] <- safe_weighted_log(posterior[1, ], posterior_weight) +
    safe_weighted_log(initial_probs * emissions[1, ], transition_weight)

  if (n == 1) {
    path[1] <- which.max(score[1, ])
    return(path)
  }

  for (t in 2:n) {
    post_term <- safe_weighted_log(posterior[t, ], posterior_weight)
    emission_term <- safe_weighted_log(emissions[t, ], transition_weight)
    for (j in seq_len(nstates)) {
      previous_scores <- score[t - 1, ] + safe_weighted_log(transition_matrix[, j], transition_weight)
      score[t, j] <- post_term[j] + emission_term[j] + max(previous_scores)
      backpointer[t, j] <- which.max(previous_scores)
    }
  }

  path[n] <- which.max(score[n, ])
  for (t in seq(n - 1, 1, -1)) {
    path[t] <- backpointer[t + 1, path[t + 1]]
  }

  path
}

decode_paths <- function(observations, transition_matrix, initial_probs, means, sds, omega = 0.1) {
  emissions <- emission_matrix(observations, means, sds)
  posterior <- posterior_probabilities(observations, transition_matrix, initial_probs, emissions)
  viterbi_states <- viterbi_path(observations, transition_matrix, initial_probs, emissions)
  pmap_states <- pmap_path(observations, transition_matrix, initial_probs, emissions)

  if (omega == 0) {
    hybrid_states <- pmap_states
  } else if (omega == 1) {
    hybrid_states <- viterbi_states
  }
}

```

```

} else {
  hybrid_states <- hybrid_path(observations, transition_matrix, initial_probs, emissions, omega = omega)
}

list(
  emissions = emissions,
  posterior = posterior,
  viterbi = viterbi_states,
  pmap = pmap_states,
  hybrid = hybrid_states
)
}

fit_ordered_hmm <- function(observations, seed = 123) {
  hmm_data <- data.frame(observation = observations)
  model <- depmixS4::depmix(
    response = observation ~ 1,
    data = hmm_data,
    nstates = 3,
    family = gaussian(),
    transition = ~1,
    prior = ~1
  )

  if (!is.null(seed)) {
    set.seed(seed)
  }

  utils::capture.output({
    fitted_model <- depmixS4::fit(model, verbose = FALSE)
  })
  parameters <- depmixS4::getpars(fitted_model)

  initial_probs <- parameters[1:3]
  initial_probs <- initial_probs / sum(initial_probs)
  transition_matrix <- matrix(parameters[4:12], nrow = 3, byrow = TRUE)
  means <- parameters[c(13, 15, 17)]
  sds <- parameters[c(14, 16, 18)]

  order_index <- order(means)
  list(
    fit = fitted_model,
    initial_probs = initial_probs[order_index],
    transition_matrix = transition_matrix[order_index, order_index],
    means = means[order_index],
    sds = sds[order_index]
  )
}

```

```

fit_and_decode <- function(observations, fit_seed = 123, omega = 0.1) {
  fitted <- fit_ordered_hmm(observations, seed = fit_seed)
  decoded <- decode_paths(
    observations = observations,
    transition_matrix = fitted$transition_matrix,
    initial_probs = fitted$initial_probs,
    means = fitted$means,
    sds = fitted$sds,
    omega = omega
  )

  c(fitted, decoded)
}

format_metric <- function(values, digits = 4) {
  paste0(
    round(mean(values), digits),
    " (sd=",
    round(stats::sd(values), digits),
    ")"
  )
}

format_mean_sd <- function(mean_value, sd_value, digits = 2) {
  paste0(round(mean_value, digits), " (sd=", round(sd_value, digits), ")")
}

transition_matrix_string <- function(transition_matrix, digits = 4) {
  paste(
    apply(round(transition_matrix, digits), 1, function(row) {
      paste0("[", paste(row, collapse = ", "), "]")
    }),
    collapse = "\n"
  )
}

incorrect_rate <- function(reference_states, predicted_states) {
  mean(reference_states != predicted_states)
}

state_error_rate <- function(reference_states, predicted_states, state_id) {
  mask <- reference_states == state_id
  if (!any(mask)) {
    return(0)
  }
  mean(reference_states[mask] != predicted_states[mask])
}

```

```

pointwise_difference <- function(path_1, path_2) {
  mean(path_1 != path_2)
}

impossible_transition_percent <- function(states, transition_matrix) {
  if (length(states) < 2) {
    return(0)
  }
  impossible <- sum(transition_matrix[cbind(states[-length(states)], states[-1])] == 0)
  impossible / (length(states) - 1) * 100
}

compute_return_frequencies <- function(log_returns, states, labels = state_labels) {
  positive <- sapply(seq_along(labels), function(idx) {
    state_returns <- log_returns[states == idx]
    if (length(state_returns) == 0) {
      return(0)
    }
    mean(state_returns > 0)
  })

  negative <- sapply(seq_along(labels), function(idx) {
    state_returns <- log_returns[states == idx]
    if (length(state_returns) == 0) {
      return(0)
    }
    mean(state_returns <= 0)
  })

  result <- data.frame(rbind(positive, negative))
  names(result) <- labels
  rownames(result) <- c("Positive return freq.", "Negative return freq.")
  result
}

compute_state_stats <- function(log_returns, states, model_means, model_sds, labels = state_labels) {
  columns <- lapply(seq_along(labels), function(idx) {
    state_returns <- log_returns[states == idx]
    sample_mean <- if (length(state_returns) > 0) mean(state_returns) else 0
    sample_sd <- if (length(state_returns) > 1) stats::sd(state_returns) else 0
    sample_size <- length(state_returns)
    z_stat <- if (sample_size > 0) sample_mean / (model_sds[idx] / sqrt(sample_size)) else 0

    c(
      model_means[idx],
      model_sds[idx],
      sample_mean,

```

```

        sample_sd,
        sample_size,
        z_stat
    )
})

result <- as.data.frame(do.call(cbind, columns))
names(result) <- labels
rownames(result) <- c(
    "Model Mean",
    "Model Std.dev",
    "Sample Mean",
    "Sample Std.dev",
    "Sample Size",
    "Z-statistics"
)

result[1:4, ] <- round(result[1:4, ], 4)
result[6, ] <- round(result[6, ], 4)
result
}

```

A.2 Simulation Driver

```

library(ggplot2)

script_file_arg <- grep("--file=", commandArgs(trailingOnly = FALSE), value = TRUE)
decode_script_arg <- function(path) {
    normalizePath(gsub("~\\+", " ", path), mustWork = FALSE)
}
script_dir <- if (length(script_file_arg) > 0) {
    dirname(decode_script_arg(sub("--file=", "", script_file_arg[1])))
} else {
    normalizePath(getwd(), mustWork = FALSE)
}

source(file.path(script_dir, "hmm_utils.R"))

paths <- project_paths()

dependence_dir <- file.path(paths$code_dir, "Impact of dependence")
overlap_dir <- file.path(paths$code_dir, "Impact of overlap")
weight_dir <- file.path(paths$code_dir, "Impact of weight parameter")
figure_dir <- file.path(paths$thesis_dir, "cha3")

for (dir_path in c(dependence_dir, overlap_dir, weight_dir, figure_dir)) {
    ensure_dir(dir_path)
}

```

```

}

build_overlap_levels_plot <- function(overlap_settings) {
  panel_data <- list()
  header_notes <- list()
  overlap_notes <- list()

  for (label in names(overlap_settings)) {
    means <- overlap_settings[[label]]$means
    sds <- overlap_settings[[label]]$sds
    x_grid <- seq(min(means - 5 * sds), max(means + 7 * sds), length.out = 1000)
    max_density <- max(sapply(seq_along(means), function(idx) dnorm(x_grid, mean = means[idx], sd = sds[idx])
    ))

    panel_data[[label]] <- do.call(rbind, lapply(seq_along(means), function(idx) {
      data.frame(
        Scenario = label,
        x = x_grid,
        Density = dnorm(x_grid, mean = means[idx], sd = sds[idx]),
        State = state_labels[idx]
      )
    })))

    note_x <- min(x_grid) + diff(range(x_grid)) * 0.60
    header_notes[[label]] <- data.frame(
      Scenario = label,
      x = note_x,
      y = max_density * 0.95,
      label = paste(label, "overlap")
    )

    overlap_notes[[label]] <- data.frame(
      Scenario = label,
      x = note_x,
      y = max_density * c(0.80, 0.66, 0.52),
      label = c(
        sprintf("1-2 overlap = %.3f", overlap_area(means[1], sds[1], means[2], sds[2])),
        sprintf("2-3 overlap = %.3f", overlap_area(means[2], sds[2], means[3], sds[3])),
        sprintf("1-3 overlap = %.3f", overlap_area(means[1], sds[1], means[3], sds[3]))
      )
    )
  }

  panel_data <- do.call(rbind, panel_data)
  header_notes <- do.call(rbind, header_notes)
  overlap_notes <- do.call(rbind, overlap_notes)

  ggplot(panel_data, aes(x = x, y = Density, color = State)) +

```

```

geom_line(linewidth = 0.9) +
geom_text(
  data = header_notes,
  aes(x = x, y = y, label = label),
  inherit.aes = FALSE,
  hjust = 0,
  size = 3.6,
  fontface = "bold"
) +
geom_text(
  data = overlap_notes,
  aes(x = x, y = y, label = label),
  inherit.aes = FALSE,
  hjust = 0,
  size = 3.2
) +
facet_wrap(~Scenario, ncol = 1, scales = "free_x") +
scale_color_manual(values = state_colors) +
labs(x = "Observation", y = "Density") +
theme_minimal(base_size = 11) +
theme(
  legend.position = "top",
  panel.grid.minor = element_blank(),
  strip.text = element_text(face = "bold")
)
}

summarize_known_decoder_case <- function(sim_data, transition_matrix, initial_probs, means, sds, omega) {
  decoded <- decode_paths(sim_data$observation, transition_matrix, initial_probs, means, sds, omega = omega)

  c(
    viterbi_error = incorrect_rate(sim_data$state, decoded$viterbi),
    pmap_error = incorrect_rate(sim_data$state, decoded$pmap),
    hybrid_error = incorrect_rate(sim_data$state, decoded$hybrid),
    state1_viterbi = state_error_rate(sim_data$state, decoded$viterbi, 1),
    state1_pmap = state_error_rate(sim_data$state, decoded$pmap, 1),
    state1_hybrid = state_error_rate(sim_data$state, decoded$hybrid, 1),
    state3_viterbi = state_error_rate(sim_data$state, decoded$viterbi, 3),
    state3_pmap = state_error_rate(sim_data$state, decoded$pmap, 3),
    state3_hybrid = state_error_rate(sim_data$state, decoded$hybrid, 3),
    pmap_vs_viterbi = pointwise_difference(decoded$viterbi, decoded$pmap),
    hybrid_vs_viterbi = pointwise_difference(decoded$viterbi, decoded$hybrid),
    impossible_viterbi = impossible_transition_percent(decoded$viterbi, transition_matrix),
    impossible_pmap = impossible_transition_percent(decoded$pmap, transition_matrix),
    impossible_hybrid = impossible_transition_percent(decoded$hybrid, transition_matrix)
  )
}

```

```

format_known_decoder_table <- function(results_matrix, labels, column_name) {
  summary_df <- data.frame(
    scenario = labels,
    Viterbi_Error_Rate = apply(results_matrix[, , "viterbi_error"], 2, format_metric),
    PMAP_Error_Rate = apply(results_matrix[, , "pmap_error"], 2, format_metric),
    Hybrid_Error_Rate = apply(results_matrix[, , "hybrid_error"], 2, format_metric),
    State_1_Error_Viterbi = apply(results_matrix[, , "state1_viterbi"], 2, format_metric),
    State_1_Error_PMAP = apply(results_matrix[, , "state1_pmap"], 2, format_metric),
    State_1_Error_Hybrid = apply(results_matrix[, , "state1_hybrid"], 2, format_metric),
    State_3_Error_Viterbi = apply(results_matrix[, , "state3_viterbi"], 2, format_metric),
    State_3_Error_PMAP = apply(results_matrix[, , "state3_pmap"], 2, format_metric),
    State_3_Error_Hybrid = apply(results_matrix[, , "state3_hybrid"], 2, format_metric),
    Pointwise_Diff_PMAP_Viterbi = round(colMeans(results_matrix[, , "pmap_vs_viterbi"]), 4),
    Pointwise_Diff_Hybrid_Viterbi = round(colMeans(results_matrix[, , "hybrid_vs_viterbi"]), 4),
    Impossible_Transitions_Percent_Viterbi = apply(results_matrix[, , "impossible_viterbi"], 2, function(x)
      format_metric(x, digits = 2)),
    Impossible_Transitions_Percent_PMAP = apply(results_matrix[, , "impossible_pmap"], 2, function(x) format_
      metric(x, digits = 2)),
    Impossible_Transitions_Percent_Hybrid = apply(results_matrix[, , "impossible_hybrid"], 2, function(x)
      format_metric(x, digits = 2))
  )

  transposed <- as.data.frame(t(summary_df))
  colnames(transposed) <- paste0(column_name, "_", summary_df$scenario)
  transposed
}

run_dependence_study <- function() {
  repetitions <- 500
  n <- 200
  omega <- 0.5
  initial_probs <- c(1 / 3, 1 / 3, 1 / 3)
  means <- c(-1, 0, 1)
  sds <- c(3, 1, 2)

  transition_matrices <- list(
    '0.75' = matrix(c(0.75, 0.25, 0, 0, 0.75, 0.25, 0.25, 0, 0.75), nrow = 3, byrow = TRUE),
    '0.9' = matrix(c(0.9, 0.1, 0, 0, 0.9, 0.1, 0.1, 0, 0.9), nrow = 3, byrow = TRUE),
    '0.95' = matrix(c(0.95, 0.05, 0, 0, 0.95, 0.05, 0.05, 0, 0.95), nrow = 3, byrow = TRUE),
    Mixed = matrix(c(0.75, 0.25, 0, 0, 0.9, 0.1, 0.05, 0, 0.95), nrow = 3, byrow = TRUE)
  )

  metrics <- c(
    "viterbi_error", "pmap_error", "hybrid_error",
    "state1_viterbi", "state1_pmap", "state1_hybrid",
    "state3_viterbi", "state3_pmap", "state3_hybrid",
    "pmap_vs_viterbi", "hybrid_vs_viterbi",
    "impossible_viterbi", "impossible_pmap", "impossible_hybrid"
  )

```

```

)

set.seed(123)
results <- array(
  NA_real_,
  dim = c(repetitions, length(transition_matrices), length(metrics)),
  dimnames = list(NULL, names(transition_matrices), metrics)
)

for (scenario_idx in seq_along(transition_matrices)) {
  transition_matrix <- transition_matrices[[scenario_idx]]
  for (rep_idx in seq_len(repetitions)) {
    sim_data <- simulate_hmm(n, means, sds, transition_matrix, initial_probs)
    results[rep_idx, scenario_idx, ] <- summarize_known_decoder_case(
      sim_data, transition_matrix, initial_probs, means, sds, omega
    )
  }
}

summary_table <- format_known_decoder_table(results, names(transition_matrices), "Transition")
rownames(summary_table)[1] <- "Diagonal_Value"
write_csv_file(
  summary_table,
  file.path(dependence_dir, "HMM_Simulation_Results_Transition_Matrix_Transposed.csv"),
  row.names = TRUE
)
}

run_overlap_study <- function() {
  repetitions <- 500
  n <- 200
  omega <- 0.5
  initial_probs <- c(1 / 3, 1 / 3, 1 / 3)
  transition_matrix <- matrix(c(0.75, 0.25, 0, 0.2, 0, 0.8, 0, 0.7, 0.3), nrow = 3, byrow = TRUE)

  overlap_settings <- list(
    Low = list(means = c(-3, 0, 3), sds = c(3, 1, 2)),
    Medium = list(means = c(-2, 0, 2), sds = c(3, 1, 2)),
    High = list(means = c(-1, 0, 1), sds = c(3, 1, 2))
  )

  overlap_plot <- build_overlap_levels_plot(overlap_settings)

  save_plot_multiple(
    overlap_plot,
    c(
      file.path(overlap_dir, "density_combined.png"),
      file.path(figure_dir, "emission_overlap_levels.png")
    )
  )
}

```

```

    ),
    width = 8,
    height = 10
  )

metrics <- c(
  "viterbi_error", "pmap_error", "hybrid_error",
  "state1_viterbi", "state1_pmap", "state1_hybrid",
  "state3_viterbi", "state3_pmap", "state3_hybrid",
  "pmap_vs_viterbi", "hybrid_vs_viterbi",
  "impossible_viterbi", "impossible_pmap", "impossible_hybrid"
)

set.seed(123)
results <- array(
  NA_real_,
  dim = c(repetitions, length(overlap_settings), length(metrics)),
  dimnames = list(NULL, names(overlap_settings), metrics)
)

for (scenario_idx in seq_along(overlap_settings)) {
  means <- overlap_settings[[scenario_idx]]$means
  sds <- overlap_settings[[scenario_idx]]$sds
  for (rep_idx in seq_len(repetitions)) {
    sim_data <- simulate_hmm(n, means, sds, transition_matrix, initial_probs)
    results[rep_idx, scenario_idx, ] <- summarize_known_decoder_case(
      sim_data, transition_matrix, initial_probs, means, sds, omega
    )
  }
}

summary_table <- format_known_decoder_table(results, names(overlap_settings), "Overlap")
rownames(summary_table)[1] <- "Overlap_Level"
write_csv_file(
  summary_table,
  file.path(overlap_dir, "HMM_Simulation_Results_Emission_Overlap_Transposed.csv"),
  row.names = TRUE
)
}

run_weight_study <- function() {
  repetitions <- 200
  n <- 1000
  initial_probs <- c(1 / 3, 1 / 3, 1 / 3)
  means <- c(-1, 0, 1)
  sds <- c(3, 1, 2)
  transition_matrix <- matrix(c(0.9, 0.1, 0, 0, 0.9, 0.1, 0.1, 0, 0.9), nrow = 3, byrow = TRUE)
  omega_values <- c(0, seq(0.05, 0.95, by = 0.05), 1)

```

```

set.seed(123)

overall_error <- matrix(NA_real_, nrow = repetitions, ncol = length(omega_values))
state1_error <- matrix(NA_real_, nrow = repetitions, ncol = length(omega_values))
state3_error <- matrix(NA_real_, nrow = repetitions, ncol = length(omega_values))
impossible_transitions <- matrix(NA_real_, nrow = repetitions, ncol = length(omega_values))

for (rep_idx in seq_len(repetitions)) {
  sim_data <- simulate_hmm(n, means, sds, transition_matrix, initial_probs)
  fitted <- fit_ordered_hmm(sim_data$observation, seed = 123 + rep_idx)

  for (omega_idx in seq_along(omega_values)) {
    decoded <- decode_paths(
      observations = sim_data$observation,
      transition_matrix = fitted$transition_matrix,
      initial_probs = fitted$initial_probs,
      means = fitted$means,
      sds = fitted$sds,
      omega = omega_values[omega_idx]
    )

    overall_error[rep_idx, omega_idx] <- incorrect_rate(sim_data$state, decoded$hybrid)
    state1_error[rep_idx, omega_idx] <- state_error_rate(sim_data$state, decoded$hybrid, 1)
    state3_error[rep_idx, omega_idx] <- state_error_rate(sim_data$state, decoded$hybrid, 3)
    # The fitted matrix is usually fully interior here, so use the known DGP
    # restrictions to measure whether decoded paths violate truly impossible moves.
    impossible_transitions[rep_idx, omega_idx] <- impossible_transition_percent(decoded$hybrid, transition_
matrix)
  }
}

results <- data.frame(
  Omega = omega_values,
  Overall_Error = apply(overall_error, 2, format_metric),
  State1_Error = apply(state1_error, 2, format_metric),
  State3_Error = apply(state3_error, 2, format_metric),
  Impossible_Transitions_Percent = apply(impossible_transitions, 2, function(x) format_metric(x, digits =
2))
)

transposed <- as.data.frame(t(results[, -1]))
colnames(transposed) <- sprintf("Omega_%.2f", results$Omega)
rownames(transposed) <- c(
  "Overall_Error_Rate",
  "State_1_Error_Rate",
  "State_3_Error_Rate",
  "Impossible_Transitions_Percent_True_Matrix"
)

```

```
write_csv_file(  
  transposed,  
  file.path(weight_dir, "HMM_Simulation_Omega_Metrics_Transposed.csv"),  
  row.names = TRUE  
)  
}  
  
run_dependence_study()  
run_overlap_study()  
run_weight_study()
```

B Appendix B: CSI 300 Application Code

This appendix contains the compact R script used in Chapter 5. It reuses the shared utilities from Appendix A, fits the weekly and monthly hidden Markov models, writes the decoder summaries, and regenerates the monthly expanding-window validation outputs and trading figure.

B.1 CSI 300 Application Driver

```
library(ggplot2)
library(dplyr)

script_file_arg <- grep("--file=", commandArgs(trailingOnly = FALSE), value = TRUE)
decode_script_arg <- function(path) {
  normalizePath(gsub("~\\+~", " ", path), mustWork = FALSE)
}
script_dir <- if (length(script_file_arg) > 0) {
  dirname(decode_script_arg(sub("--file=", "", script_file_arg[1])))
} else {
  normalizePath(getwd(), mustWork = FALSE)
}

source(file.path(script_dir, "hmm_utils.R"))

paths <- project_paths()
omega <- 0.1
training_cutoff <- as.Date("2013-12-31")

read_csi300_data <- function(data_path) {
  read.csv(data_path, header = TRUE, stringsAsFactors = FALSE) %>%
    mutate(
      Date = as.Date(Date, format = "%m/%d/%Y"),
      Price = as.numeric(gsub(",", "", Price))
    ) %>%
    arrange(Date) %>%
    mutate(
      Log_Return_Decimal = log(Price) - log(lag(Price)),
      Log_Return = 100 * Log_Return_Decimal
    ) %>%
    filter(!is.na(Log_Return_Decimal))
}

emission_plot <- function(means, sds, title_text) {
  x_grid <- seq(min(means - 5 * sds), max(means + 5 * sds), length.out = 1000)
  density_df <- do.call(rbind, lapply(seq_along(means), function(idx) {
    data.frame(
      x = x_grid,
```

```

    Density = dnorm(x_grid, mean = means[idx], sd = sds[idx]),
    State = state_labels[idx]
  )
}))

overlap_notes <- c(
  sprintf("Bear-Sidewalk overlap = %.3f", overlap_area(means[1], sds[1], means[2], sds[2])),
  sprintf("Sidewalk-Bull overlap = %.3f", overlap_area(means[2], sds[2], means[3], sds[3])),
  sprintf("Bear-Bull overlap = %.3f", overlap_area(means[1], sds[1], means[3], sds[3]))
)

ggplot(density_df, aes(x = x, y = Density, color = State)) +
  geom_line(linewidth = 1) +
  scale_color_manual(values = state_colors) +
  labs(title = title_text, x = "Log return", y = "Density") +
  theme_minimal(base_size = 11) +
  theme(
    legend.position = "top",
    panel.grid.minor = element_blank(),
    plot.title = element_text(face = "bold")
  ) +
  annotate("text", x = min(x_grid) + diff(range(x_grid)) * 0.60, y = max(density_df$Density) * 0.88,
    label = overlap_notes[1], hjust = 0, size = 3.3) +
  annotate("text", x = min(x_grid) + diff(range(x_grid)) * 0.60, y = max(density_df$Density) * 0.74,
    label = overlap_notes[2], hjust = 0, size = 3.3) +
  annotate("text", x = min(x_grid) + diff(range(x_grid)) * 0.60, y = max(density_df$Density) * 0.60,
    label = overlap_notes[3], hjust = 0, size = 3.3)
}

monthly_histogram_pre_post_plot <- function(data) {
  period_levels <- c("Before 2016", "2016 onward")
  plot_data <- data %>%
    mutate(
      Period = ifelse(Date < as.Date("2016-01-01"), "Before 2016", "2016 onward"),
      Period = factor(Period, levels = period_levels)
    )

  stats_df <- plot_data %>%
    group_by(Period) %>%
    summarise(mu = mean(Log_Return), sigma = sd(Log_Return), .groups = "drop")

  x_grid <- seq(min(plot_data$Log_Return), max(plot_data$Log_Return), length.out = 800)
  normal_df <- bind_rows(lapply(seq_len(nrow(stats_df)), function(idx) {
    data.frame(
      Period = stats_df$Period[idx],
      x = x_grid,
      Density = dnorm(x_grid, mean = stats_df$mu[idx], sd = stats_df$sigma[idx])
    )
  })

```

```

})) %>%
  mutate(Period = factor(Period, levels = period_levels))

ggplot(plot_data, aes(x = Log_Return)) +
  geom_histogram(aes(y = after_stat(density)), bins = 28, fill = "grey80", color = "black", linewidth =
    0.35) +
  geom_line(
    data = normal_df,
    aes(x = x, y = Density, color = "Fitted Normal Density", linetype = "Fitted Normal Density"),
    linewidth = 1.2
  ) +
  facet_wrap(~Period, ncol = 2) +
  scale_color_manual(values = c("Fitted Normal Density" = "red")) +
  scale_linetype_manual(values = c("Fitted Normal Density" = "dashed")) +
  labs(x = "Monthly log return (%)", y = "Density", color = NULL, linetype = NULL) +
  theme_bw(base_size = 11) +
  theme(
    legend.position.inside = c(0.5, 0.985),
    legend.justification = c(0.5, 1),
    legend.direction = "horizontal",
    legend.background = element_rect(fill = "white", color = "black"),
    legend.key = element_blank(),
    strip.background = element_rect(fill = "grey92", color = "black"),
    panel.grid.minor = element_blank()
  )
}

transition_plot <- function(dates, prices, viterbi_states, pmap_states, hybrid_states, title_text) {
  interval_days <- max(1, round(stats::median(as.numeric(diff(dates))), na.rm = TRUE))
  decoder_levels <- c(
    "Viterbi Predicted States",
    "PMAP Predicted States",
    "Hybrid (omega = 0.10) Predicted States"
  )

  regime_df <- bind_rows(
    data.frame(Date = dates, Decoder = "Viterbi Predicted States", State = factor(viterbi_states, levels =
      1:3, labels = state_labels)),
    data.frame(Date = dates, Decoder = "PMAP Predicted States", State = factor(pmap_states, levels = 1:3,
      labels = state_labels)),
    data.frame(Date = dates, Decoder = "Hybrid (omega = 0.10) Predicted States", State = factor(hybrid_states
      , levels = 1:3, labels = state_labels))
  ) %>%
  mutate(Decoder = factor(Decoder, levels = decoder_levels)) %>%
  arrange(Decoder, Date) %>%
  group_by(Decoder) %>%
  mutate(
    xmin = Date,

```

```

    xmax = lead(Date, default = max(Date) + interval_days)
  ) %>%
  ungroup()

price_df <- bind_rows(
  data.frame(Date = dates, Price = prices, Decoder = "Viterbi Predicted States"),
  data.frame(Date = dates, Price = prices, Decoder = "PMAP Predicted States"),
  data.frame(Date = dates, Price = prices, Decoder = "Hybrid (omega = 0.10) Predicted States")
) %>%
  mutate(Decoder = factor(Decoder, levels = decoder_levels))

y_min <- min(prices, na.rm = TRUE) * 0.85
y_max <- max(prices, na.rm = TRUE) * 1.05

ggplot() +
  geom_rect(
    data = regime_df,
    aes(xmin = xmin, xmax = xmax, ymin = y_min, ymax = y_max, fill = State),
    alpha = 0.88
  ) +
  geom_line(
    data = price_df,
    aes(x = Date, y = Price),
    color = "black",
    linewidth = 0.35
  ) +
  facet_wrap(~Decoder, ncol = 1) +
  scale_fill_manual(values = state_colors) +
  scale_x_date(date_breaks = "5 years", date_labels = "%Y") +
  labs(title = title_text, x = NULL, y = "CSI 300 index level", fill = NULL) +
  theme_minimal(base_size = 11) +
  theme(
    legend.position = "top",
    panel.grid.minor = element_blank(),
    strip.text = element_text(face = "bold"),
    plot.title = element_blank(),
    axis.title.y = element_text(margin = margin(r = 8))
  )
}

save_hmm_outputs <- function(data, decoded, output_dir, prefix, figure_paths, summary_transposed = FALSE) {
  n <- nrow(data)

  pointwise_summary <- c(
    Pointwise_Diff_Viterbi_PMAP = round(pointwise_difference(decoded$viterbi, decoded$pmap), 4),
    Pointwise_Diff_Viterbi_Hybrid01 = round(pointwise_difference(decoded$viterbi, decoded$hybrid), 4),
    Pointwise_Diff_PMAP_Hybrid01 = round(pointwise_difference(decoded$pmap, decoded$hybrid), 4)
  )
}

```

```

summary_df <- data.frame(
  Bear_mu = format_mean_sd(decoded$means[1], decoded$sds[1]),
  Sidewalk_mu = format_mean_sd(decoded$means[2], decoded$sds[2]),
  Bull_mu = format_mean_sd(decoded$means[3], decoded$sds[3]),
  Est_Transition_Matrix = transition_matrix_string(decoded$transition_matrix),
  Sequence_Length = n,
  Pointwise_Diff_Viterbi_PMAP = pointwise_summary[["Pointwise_Diff_Viterbi_PMAP"]],
  Pointwise_Diff_Viterbi_Hybrid01 = pointwise_summary[["Pointwise_Diff_Viterbi_Hybrid01"]],
  Pointwise_Diff_PMAP_Hybrid01 = pointwise_summary[["Pointwise_Diff_PMAP_Hybrid01"]]
)

summary_path <- file.path(output_dir, paste0(prefix, "_3State_Results_Hybrid.csv"))
if (summary_transposed) {
  summary_transposed_df <- as.data.frame(t(summary_df))
  colnames(summary_transposed_df) <- "Value"
  write_csv_file(summary_transposed_df, summary_path, row.names = TRUE)
} else {
  write_csv_file(summary_df, summary_path, row.names = FALSE)
}

posterior_df <- data.frame(
  Time = seq_len(n),
  Date = data$Date,
  Bear_Prob = decoded$posterior[, 1],
  Sidewalk_Prob = decoded$posterior[, 2],
  Bull_Prob = decoded$posterior[, 3],
  Viterbi = factor(decoded$viterbi, levels = 1:3, labels = state_labels),
  PMAP = factor(decoded$pmap, levels = 1:3, labels = state_labels),
  Hybrid_01 = factor(decoded$hybrid, levels = 1:3, labels = state_labels),
  Price = data$Price,
  Log_Return = data$Log_Return
)

write_csv_file(
  posterior_df,
  file.path(output_dir, paste0(prefix, "_Posterior_Probabilities_Hybrid.csv")),
  row.names = FALSE
)

viterbi_stats <- compute_state_stats(data$Log_Return, decoded$viterbi, decoded$means, decoded$sds)
pmap_stats <- compute_state_stats(data$Log_Return, decoded$pmap, decoded$means, decoded$sds)
hybrid_stats <- compute_state_stats(data$Log_Return, decoded$hybrid, decoded$means, decoded$sds)
viterbi_freq <- round(compute_return_frequencies(data$Log_Return, decoded$viterbi), 4)
pmap_freq <- round(compute_return_frequencies(data$Log_Return, decoded$pmap), 4)
hybrid_freq <- round(compute_return_frequencies(data$Log_Return, decoded$hybrid), 4)

write_csv_file(viterbi_stats, file.path(output_dir, paste0(prefix, "_viterbi_stats.csv")), row.names = TRUE
)

```

```

write_csv_file(pmap_stats, file.path(output_dir, paste0(prefix, "_pmap_stats.csv")), row.names = TRUE)
write_csv_file(hybrid_stats, file.path(output_dir, paste0(prefix, "_hybrid_01_stats.csv")), row.names =
  TRUE)

write_csv_file(viterbi_freq, file.path(output_dir, paste0(prefix, "_viterbi_freq.csv")), row.names = TRUE)
write_csv_file(pmap_freq, file.path(output_dir, paste0(prefix, "_pmap_freq.csv")), row.names = TRUE)
write_csv_file(hybrid_freq, file.path(output_dir, paste0(prefix, "_hybrid_01_freq.csv")), row.names = TRUE)

emission_figure <- emission_plot(decoded$means, decoded$sds, figure_paths$emission_title)
save_plot_multiple(emission_figure, figure_paths$emission, width = 8, height = 5.5)

if (!is.null(figure_paths$transition)) {
  transition_figure <- transition_plot(
    data$Date, data$Price, decoded$viterbi, decoded$pmap, decoded$hybrid, figure_paths$transition_title
  )
  save_plot_multiple(transition_figure, figure_paths$transition, width = 8, height = 8)
}

invisible(list(
  summary = summary_df,
  posterior = posterior_df,
  viterbi_stats = viterbi_stats,
  pmap_stats = pmap_stats,
  hybrid_stats = hybrid_stats,
  viterbi_freq = viterbi_freq,
  pmap_freq = pmap_freq,
  hybrid_freq = hybrid_freq
))
}

run_csi300_model <- function(data_file, output_dir, prefix, figure_paths, summary_transposed = FALSE) {
  ensure_dir(output_dir)
  data <- read_csi300_data(data_file)
  decoded <- fit_and_decode(data$Log_Return, fit_seed = 123, omega = omega)
  save_hmm_outputs(data, decoded, output_dir, prefix, figure_paths, summary_transposed = summary_transposed)
}

run_expanding_window_validation <- function(data_file, output_path, fit_seed = 123) {
  data <- read_csi300_data(data_file)
  test_indices <- which(data$Date > training_cutoff)

  validation_rows <- lapply(test_indices, function(idx) {
    window_data <- data[seq_len(idx - 1), , drop = FALSE]
    decoded <- fit_and_decode(window_data$Log_Return, fit_seed = fit_seed, omega = omega)
    decoded_state_id <- tail(decoded$hybrid, 1)
    signal <- c(-1L, 0L, 1L)[decoded_state_id]
    next_log_return <- data$Log_Return_Decimal[idx]
    next_simple_return <- exp(next_log_return) - 1
  })
}

```

```

strategy_simple_return <- signal * next_simple_return

data.frame(
  Date = data$Date[idx],
  Window_End_Date = window_data$Date[nrow(window_data)],
  Decoded_State = state_labels[decoded_state_id],
  Signal = signal,
  Next_Period_Log_Return = next_log_return,
  Next_Period_Simple_Return = next_simple_return,
  Strategy_Simple_Return = strategy_simple_return
)
})

validation_df <- bind_rows(validation_rows) %>%
  mutate(
    Strategy_Cumulative_Return = cumprod(1 + Strategy_Simple_Return) - 1,
    BuyHold_Cumulative_Return = cumprod(1 + Next_Period_Simple_Return) - 1
  )

write_csv_file(validation_df, output_path, row.names = FALSE)
validation_df
}

validation_summary_figure <- function(monthly_validation) {
  plot_df <- monthly_validation %>%
    transmute(
      Date = as.Date(Date),
      Strategy = 100 * Strategy_Cumulative_Return,
      Buy_and_Hold = 100 * BuyHold_Cumulative_Return
    ) %>%
    tidyr::pivot_longer(-Date, names_to = "Series", values_to = "Cumulative_Return")

  ggplot(plot_df, aes(x = Date, y = Cumulative_Return, color = Series)) +
    geom_line(linewidth = 1) +
    scale_color_manual(values = c("Strategy" = "#d95f02", "Buy_and_Hold" = "#1b9e77")) +
    labs(x = NULL, y = "Cumulative return (%)", color = NULL) +
    theme_minimal(base_size = 11) +
    theme(
      legend.position = "top",
      panel.grid.minor = element_blank()
    )
}

write_validation_summary <- function() {
  monthly_validation <- run_expanding_window_validation(
    data_file = file.path(paths$code_dir, "CSI 300 monthly data.csv"),
    output_path = file.path(paths$code_dir, "csi 300 monthly", "CSI300_monthly_expanding_window_strategy.csv"
  )
}

```

```

)

summary_df <- data.frame(
  Frequency = "Monthly",
  Strategy_Cumulative_Return = tail(monthly_validation$Strategy_Cumulative_Return, 1),
  BuyHold_Cumulative_Return = tail(monthly_validation$BuyHold_Cumulative_Return, 1),
  Bear_Count = sum(monthly_validation$Decoded_State == "Bear"),
  Sidewalk_Count = sum(monthly_validation$Decoded_State == "Sidewalk"),
  Bull_Count = sum(monthly_validation$Decoded_State == "Bull")
)

write_csv_file(
  summary_df,
  file.path(paths$code_dir, "csi 300 monthly", "CSI300_validation_summary.csv"),
  row.names = FALSE
)

validation_figure <- validation_summary_figure(monthly_validation)
save_plot_multiple(
  validation_figure,
  c(
    file.path(paths$code_dir, "csi 300 monthly", "CSI300_monthly_strategy_comparison.png"),
    file.path(paths$thesis_dir, "cha5", "CSI300_monthly_strategy_comparison.png")
  ),
  width = 8,
  height = 5.5
)
}

run_csi300_model(
  data_file = file.path(paths$code_dir, "CSI 300 weekly data.csv"),
  output_dir = file.path(paths$code_dir, "csi 300 weekly"),
  prefix = "CSI300_Weekly",
  figure_paths = list(
    emission = c(
      file.path(paths$code_dir, "csi 300 weekly", "CSI300_Weekly_Emission_Distributions.png"),
      file.path(paths$thesis_dir, "cha4", "CSI300_Weekly_Emission_Distributions.png")
    ),
    transition = NULL,
    emission_title = "CSI 300 weekly hidden-state distributions",
    transition_title = "CSI 300 weekly state paths"
  ),
  summary_transposed = FALSE
)

run_csi300_model(
  data_file = file.path(paths$code_dir, "CSI 300 monthly data.csv"),
  output_dir = file.path(paths$code_dir, "csi 300 monthly"),

```

```

prefix = "CSI300_monthly",
figure_paths = list(
  emission = c(
    file.path(paths$code_dir, "csi 300 monthly", "CSI300_monthly_Emission_Distributions.png"),
    file.path(paths$thesis_dir, "cha5", "CSI300_monthly_Emission_Distributions.png")
  ),
  transition = c(
    file.path(paths$code_dir, "csi 300 monthly", "CSI300_monthly_Full_Transitions.png"),
    file.path(paths$thesis_dir, "cha5", "CSI300_monthly_Full_Transitions.png")
  ),
  emission_title = "CSI 300 monthly hidden-state distributions",
  transition_title = "CSI 300 monthly state paths"
),
summary_transposed = TRUE
)

monthly_data <- read_csi300_data(file.path(paths$code_dir, "CSI 300 monthly data.csv"))
monthly_histogram_figure <- monthly_histogram_pre_post_plot(monthly_data)
save_plot_multiple(
  monthly_histogram_figure,
  c(
    file.path(paths$code_dir, "csi 300 monthly", "CSI300_monthly_histogram_pre_post_2016.png"),
    file.path(paths$thesis_dir, "cha5", "CSI300_monthly_histogram_pre_post_2016.png")
  ),
  width = 10,
  height = 4.8
)

write_validation_summary()

```

Non-exclusive licence to reproduce thesis and make thesis public

I, **Zhexin Rao**,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright, **Application of Hidden Markov Models in Financial Time Series Analysis**, supervised by **Kristi Kuljus**.
2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.
3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.
4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Zhexin Rao

30/04/2026