Bayesian Dynamic Modeling for Macro & Finance

Homework 5 – The Bootstrap Filter

Sequential Monte Carlo for the AR(1) + Noise DLM

Instructor: Prof. Hedibert F. Lopes · TA: Guilherme Piantino

Released: May 19, 2026 · Due: May 31st, 2026 - 11:59pm BRT)

Submit a single zipped folder containing your knitted PDF and your R script.

---

1. Context and learning goals

Sequential Monte Carlo (SMC), and in particular the bootstrap filter of Gordon, Salmond and Smith (1993), is the workhorse algorithm of nonlinear and non-Gaussian state-space inference. Although the model in this homework is linear and Gaussian – so the Kalman filter delivers the exact filtering distribution – studying the bootstrap filter on this benchmark is the cleanest way to understand its mechanics, its degeneracy modes, and the trade-offs that govern particle-filter design.

After completing this homework you should be able to:

- Derive the predictive and filtering distributions of an AR(1)+noise model and write the Kalman recursions in closed form.
- Implement the bootstrap filter for a generic state-space model, in modular R code that you could re-use for a nonlinear example.
- Diagnose particle-filter behaviour using the effective sample size (ESS) and the empirical filter-mean trajectory.
- Quantify how the signal-to-noise ratio W/V and the persistence ϕ interact to produce harder or easier filtering problems.
- Form an opinion about how many particles are "enough" – and why "enough" depends on the model rather than on N alone.

2. The model

We work with a single univariate state-space model – the AR(1) plus noise:

$$y_t = x_t + e_t, \qquad e_t \sim \mathcal{N}(0, V) \qquad \text{(observation eq.)} \tag{1}$$

$$x_t = \phi\, x_{t-1} + w_t, \qquad w_t \sim \mathcal{N}(0, W) \qquad \text{(state eq.)} \tag{2}$$

$$x_0 \sim \mathcal{N}\left(0, \frac{W}{1-\phi^2}\right) \qquad \text{(stationary init).} \tag{3}$$

Parameters (ϕ, V, W) are known. Time horizon $T = 200$. The only unknowns are the latent states $x_{1:T}$, and we want only the *filtering* distribution $p(x_t \mid y_{1:t})$ for $t = 1, \ldots, T$ (no smoothing, no parameter learning).

Your task: *You will run experiments at every combination of*

- **persistence** $\phi \in \{0.80, 0.90, 0.98\}$
- **observation variance** $V = 1$ (fixed)
- **state variance** $W \in \{0.25, 1.00, 4.00\}$

That gives 9 (ϕ, W) combinations. The signal-to-noise ratio $q = W/V \in \{0.25, 1, 4\}$ crosses the regimes "data-dominated" (q small), "balanced" and "state-dominated" (q large). High ϕ + small W produces a highly persistent, slowly-varying state – the regime where the bootstrap filter struggles most.

3. Background – what you should already know

The bootstrap filter approximates $p(x_t \mid y_{1:t})$ by a discrete weighted sample $\{(x_t^{(i)}, w_t^{(i)})\}_{i=1}^N$. At each time t it does:

(a) **propagate:** $x_t^{(i)*} \sim p(x_t \mid x_{t-1}^{(i)}) = \mathcal{N}(\phi\, x_{t-1}^{(i)}, W)$.

(b) **reweight:** $w_t^{(i)} \propto p(y_t \mid x_t^{(i)*}) = \mathcal{N}(y_t;\, x_t^{(i)*}, V)$.

(c) **resample:** $x_t^{(i)}$ by multinomial draw with probabilities $w_t^{(i)}$ (reset weights to $1/N$).

Steps (a)–(c) are precisely the bootstrap filter. There are no proposal choices to make – the prior transition is the proposal – which is why the algorithm is so simple, and also why it can perform poorly in high-information regimes (small V, sharply peaked likelihood).

Recommended pre-readings (already shared in class):

- Gordon, Salmond & Smith (1993), *IEE Proceedings F*, 140(2), 107–113.
- Lopes & Tsay (2011), *Journal of Forecasting*, 30, 168–209 – sections 2 and 3.
- Doucet, Godsill & Andrieu (2000), *Statistics and Computing*, 10, 197–208 – for ESS and degeneracy.

4. Tasks (graded)

Task 1 – Kalman filter as a benchmark (15 points)

Even though this homework is about the bootstrap filter, you will need the Kalman filter for ground truth. Derive – in writing, with the algebra – the one-step prediction and update recursions for this model:

$$
\begin{aligned}
m_{t|t-1} &= \phi\, m_{t-1|t-1}, \\
C_{t|t-1} &= \phi^2\, C_{t-1|t-1} + W, \\
K_t &= \frac{C_{t|t-1}}{C_{t|t-1} + V}, \\
m_{t|t} &= m_{t|t-1} + K_t\,(y_t - m_{t|t-1}), \\
C_{t|t} &= (1 - K_t)\, C_{t|t-1}.
\end{aligned}
$$

Initialise at the stationary distribution $m_{0|0} = 0$, $C_{0|0} = W/(1-\phi^2)$. Implement these recursions in R and report, for each (ϕ, W) combination, the time-averaged filter variance $\bar{C} = \frac{1}{T}\sum_{t=1}^T C_{t|t}$. Comment on how $\bar{C}$ scales with $q = W/V$ and with ϕ.

Task 2 – Simulate data (5 points)

Simulate one trajectory of length $T = 200$ for each (ϕ, W) combination, using `set.seed(20260514)`. Plot, in a 3×3 grid of panels, both x_t (solid) and y_t (dotted) on the same axes for each setting. In one short paragraph, describe how the visual character of the time series changes across the panels.

Task 3 – Implement the bootstrap filter (30 points)

Write a self-contained R function with the signature:

```
bf <- function(y, phi, V, W, N = 1000, resample = "multinomial",
               ess_threshold = NULL, seed = 42)
```

that takes the observation vector `y` of length T and returns:

- a $T \times N$ matrix of resampled particles $x_t^{(i)}$ (post-resample);

- the filter mean $m_t = \frac{1}{N}\sum_i x_t^{(i)}$;
- the filter standard deviation s_t;
- the effective sample size $\mathrm{ESS}_t = \big(\sum_i w_t^{(i)}\big)^2 / \sum_i \big(w_t^{(i)}\big)^2$ before resampling;
- the log marginal likelihood estimate $\log \hat{p}(y_{1:T})$.

Notes and constraints:

- Use the stationary distribution to initialise $x_0^{(i)} \sim \mathcal{N}\big(0,\, W/(1-\phi^2)\big)$.
- Resample at every step (the "vanilla" bootstrap filter), but make adaptive resampling available via the optional `ess_threshold` argument (resample only when ESS $<$ threshold $\cdot\, N$). You will use the adaptive variant in Task 5.
- Vectorise the propagation and weighting (no for-loop over particles inside the time loop).
- Use `set.seed` inside the function so that runs are reproducible.

Task 4 – First experiments (15 points)

For each of the 9 (ϕ, W) settings:

1. Run the bootstrap filter at $N = 1000$ with multinomial resampling at every step.
2. Plot the true x_t (black), the Kalman filter mean $m_{t|t}$ (blue) and the bootstrap filter mean (red) on a single panel. Use a 3×3 grid as before.
3. Report the RMSE of the bootstrap filter against the Kalman mean,

$$\mathrm{RMSE}_{\phi,W} = \sqrt{\tfrac{1}{T}\textstyle\sum_{t=1}^{T}\big(m_t^{\mathrm{BF}} - m_t^{\mathrm{KF}}\big)^2}.$$

   Tabulate the nine RMSE values.
4. Plot the ESS trajectory in a second 3×3 grid. Use the same y-axis on every panel.

Task 5 – Sensitivity to N and to adaptive resampling (20 points)

For the most challenging setting ($\phi = 0.98$, $W = 0.25$):

5. Repeat Task 4 with $N \in \{100, 500, 1000, 5000\}$. Plot the BF filter mean for each N as a separate line, all on the same axes alongside the Kalman mean. Compute RMSE for each N.
6. Repeat with adaptive resampling at threshold 0.5 (only resample when ESS $< 0.5\,N$). Compare the time-averaged ESS and the RMSE versus the vanilla bootstrap filter.
7. Discuss in 4–6 sentences: under what conditions does increasing N most improve the filter? Why does the "high persistence + small W" regime require many more particles than "low persistence + large W"?

Task 6 – Marginal likelihood and a model comparison (10 points)

Pretend you observe only the y series from the $(\phi = 0.9, W = 1)$ experiment but do not know which (ϕ, W) generated it. Use the bootstrap-filter estimate of $\log p(y_{1:T})$ at each candidate $(\phi, W) \in \{0.80, 0.90, 0.98\} \times \{0.25, 1, 4\}$ and report the highest-scoring combination. Also report the analytic Kalman log-likelihood (the prediction-error decomposition) and discuss any discrepancy.

Task 7 – Reflection (5 points)

In one short paragraph ($\leq$ 200 words): given what you observed, would you recommend a colleague use the bootstrap filter as their default sequential algorithm for a problem of comparable scale? Under what circumstances would you suggest they reach for the auxiliary particle filter or for particle learning instead? Cite at least one figure or number from your own results.

5. Deliverables

- A knitted Rmarkdown PDF ($\leq$ 12 pages, including figures) containing your derivations, code blocks, plots, and discussion.
- Your standalone R script (`.R`) – must be runnable from a fresh R session.
- A `README.txt` with the package versions you used (`sessionInfo()` output).

Submit as a single .zip file named

`HW_BF_<your-surname>.zip`

6. Grading rubric

Task	Points	What earns full marks
1 – Kalman benchmark	15	Correct algebra for the recursions; correct R implementation; meaningful comment on the (ϕ, W)-dependence of $\bar{C}$.
2 – Data simulation	5	3×3 plot is clearly labeled; written description distinguishes "data-dominated" from "state-dominated" panels.
3 – BF implementation	30	Function signature followed; vectorised; ESS computed correctly; log-likelihood computed via the predictive density at every step; reproducible.
4 – First experiments	15	Both 3×3 plots present; nine RMSE values reported; visual comparison BF vs KF is convincing in the easy regimes.
5 – N sensitivity	20	Four BF runs overlaid; clear ESS comparison vanilla vs adaptive; thoughtful discussion that ties number-of-particles to the geometry of $p(x_t \mid y_{1:t})$.
6 – Marginal likelihood	10	BF and Kalman log-marginal-likelihoods reported on a 3×3 table; correct identification of the data-generating setting; honest discussion of Monte-Carlo noise.
7 – Reflection	5	Specific to your numbers; references at least one figure.
Total	**100**	

7. Starter code

A companion R script (`HW_Bootstrap_Filter_AR1_plus_Noise.R`) is provided alongside this PDF. It contains:

- Helpers (DLM specification, simulation, Kalman recursions).
- A skeleton `bf()` function with the signature shown in Task 3, and the bodies of the propagate / reweight / resample steps left as `TODO` comments for you to fill in.
- A driver block that loops over the nine (ϕ, W) settings and produces the 3×3 plots.

You may reorganise the file in any way you like, but the `bf()` function must expose the public interface in Task 3 unchanged.

Skeleton excerpt (the three TODOs you need to fill in):

```
bf <- function(y, phi, V, W, N = 1000, resample = "multinomial",
               ess_threshold = NULL, seed = 42) {
  set.seed(seed)
```

```
  T_  <- length(y)
  X   <- matrix(NA_real_, nrow = T_, ncol = N)
  m   <- numeric(T_); s <- numeric(T_); ESS <- numeric(T_)
  loglik <- 0
  # initialise at stationary distribution
  x_curr <- rnorm(N, 0, sqrt(W / (1 - phi^2)))

  for (t in seq_len(T_)) {
    ## (a) propagate:  x_t^* ~ N(phi * x_{t-1}, W)
    ## YOUR CODE HERE

    ## (b) reweight:   w_t propto N(y_t; x_t^*, V)
    ## YOUR CODE HERE
    ## (and accumulate log p_hat(y_t | y_{1:t-1}) into `loglik`)

    ## (c) resample (or skip if adaptive and ESS_t > threshold * N)
    ## YOUR CODE HERE

    X[t, ] <- x_curr
    m[t]   <- mean(x_curr)
    s[t]   <- sd(x_curr)
  }
  list(X = X, m = m, s = s, ESS = ESS, loglik = loglik)
}
```

8. Academic honesty and acceptable AI use

Pair-discussion of the algorithm is encouraged. The code you submit must be your own – no two submissions should be byte-identical. AI assistants (ChatGPT, Claude, Copilot) may be used to debug R or to clarify concepts but *not* to produce the derivations in Task 1 or the discussions in Tasks 4, 5, 6, 7. If you use an AI assistant for debugging, append a brief "AI use" note to your README listing which prompts you ran and which sections of your code were affected.

Questions? Office hours: Wednesdays 3:00–4:30pm in the BDM Lab. For TA help, contact Guilherme at `guilhermejlp@al.insper.edu.br`.