Week 3 - Working with Real Data

Max Richards

2025-07-29

Summary of Week 2

In Week 2, I started working with different models, firstly I began by adjusting my error distribution, going from a gaussian model to a students t-distribution, and analysing some filters in this context. I then went on to, explore the Bootstrap particle filter and Kalman filter, in a 2 Dimensional Linear Gaussian Model playing around with this setup and exploring numerous 3D plots.

I concluded the Week by exploring different ways I could assess the effectiveness of my filtering methods implemented, beyond the simple RMSE. These included angular displacement, confidence ellipse convergence regions as well as Covariance determinant/trace.

Finding New Data

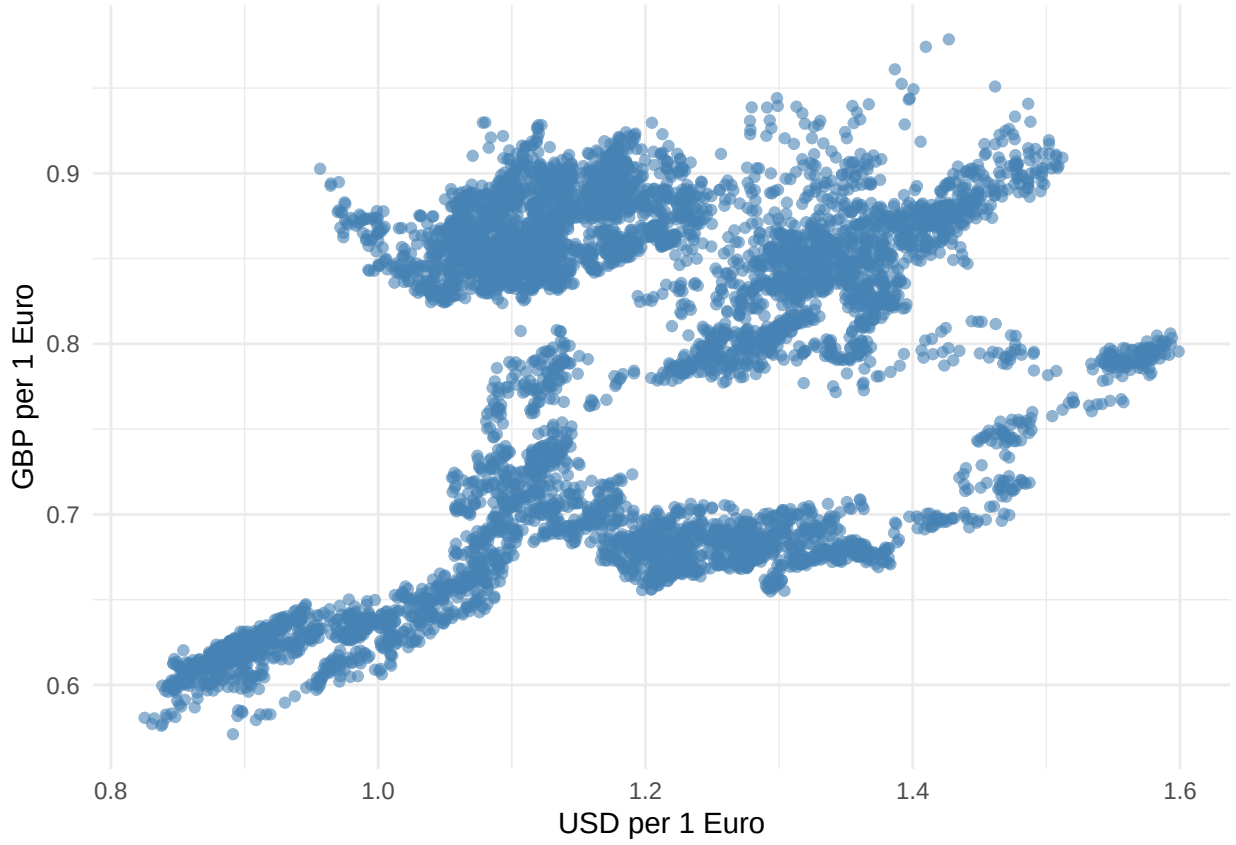
The Aim of this week, was to become comfortable taking on data, and generating filtering methods to assess certain variables/characteristics of our data set.

The Data set I found and wanted to analyse was a set containing the different convergence rates of currencies vs the euro over time.

Visualisation in 2 Dimensions

I wanted to work in 2 Dimensions again, i.e. using 3D plots with time added as an axis, and so decided to select data from 2 columns in my data set, those being the "USD" and "GBP" columns, displaying the equivalent value each of these had to 1 Euro.

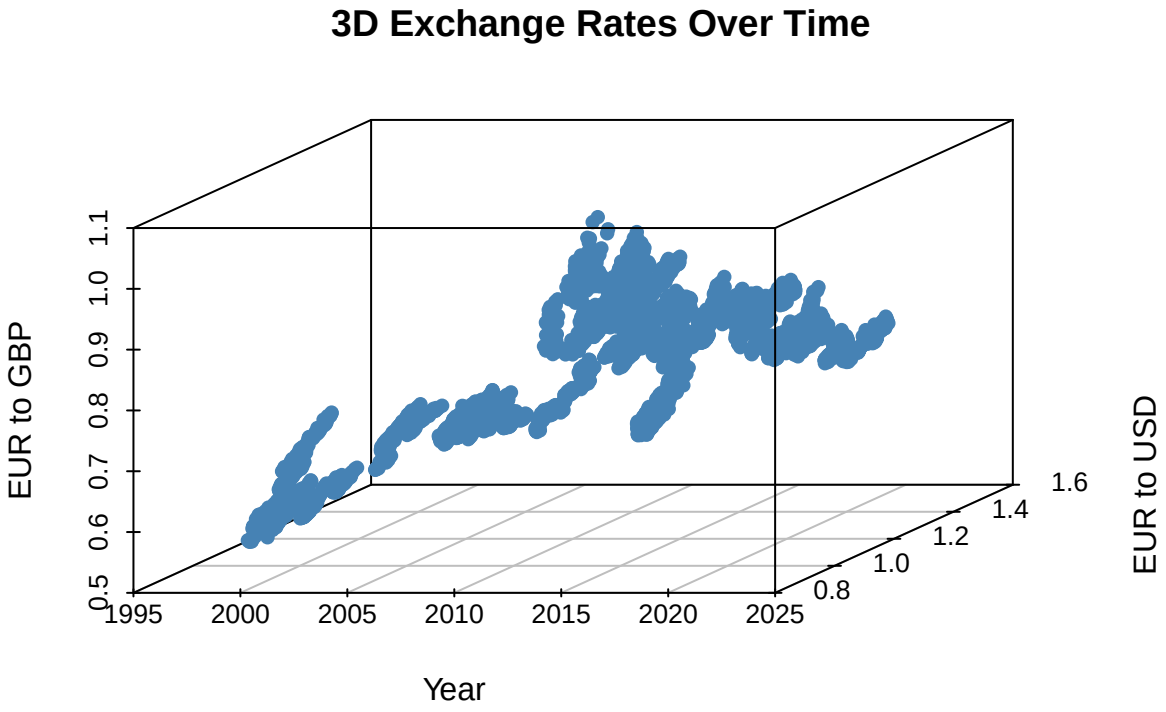
A scatterplot of the "USD" value vs "GBP" value can be found below;



Visualisation in 3 Dimensions

Adding time as an axis now, we can observe the following 3D plot.

Note that the image displayed blow is obtained by running ***scatterplot3d*** function in R, for sake of visualisation on the knitted markdown file, but have also included an interactive plot using the ***rgl*** function within the markdown file itself, which allows for better visualisation of the points..



Adding the Filter

The next stage now, was to develop a method to go from dealing with these data points to actually generating the filter.

I applied a Sequential Monte Carlo method (i.e. a Particle Filter) to estimate the evolving hidden state over time, the states here being the value of the EUR to USD and EUR to GBP exchange rates.

To begin, some assumptions were necessary, those being the fact that the state evolves according to a **stochastic transition model** (i.e. Random walk), and that observations are drawn from a **likelihood model** given the state.

Once again, the goal, as it has always been, is to approximate the **posterior distribution** of the state at each time step using weighted samples (particles).

Notation

- $x_t = \begin{bmatrix} x_t^{USD} \\ x_t^{GBP} \end{bmatrix}$: Hidden state at time t
- $\bar{y_t}$: Observed mean exchange rate at time t
- $p(x_t|x_{t-1})$: The state transition model (how model evolves)
- $p(y_t|x_t)$: The observation likelihood
- $\{x_t^{(i)}\}_{i=1}^{M}$: The particles (samples from distribution of x_t)

The Process

Step 1: Since this is the first time I am working with a real data set, not a simulated model, I had to ensure the data was 'clean', and so I implemented some code to do so, as well as ensuring the data was listed in a chronological order, to simulate time series. I also set the size of my filter at this stage, deciding the number of time stamps to be equivalent to the total number of individual years on record, and creating 1000 particles.

Step 2: As with all other particle filters that I have worked with, the next step was to now initialise my data set. Remembering that particles represent possible values of the hidden state $x_1^{(i)}$, and assuming that the true state at $t = 1$ is close to the observed mean, I was able to sample from a Bi-Variate Gaussian distribution, centered at the observed means for USD and GBP, with a small isotropic covariance, σ^2 , which I set to be $\sigma^2 = 0.1$. Mathematically speaking, we hence had;

$$x_1^{(i)} \sim \mathbb{N}(\bar{y}_1, \sigma^2 I)$$

This gave me a cloud of initial particles, that roughly approximated the true initial hidden state. Thus I was able to approximate the hidden state at $t = 1$ by using the empirical mean of the particles: $\hat{x}_1 = \frac{1}{M} \sum_i x_1^{(i)}$.

Step 3: At this stage, I I then implemented the recursive structure of the particle filter, looping from time $t = 2$ to $t = N$. Each iteration consisted of the following steps:

• Prediction Step :

I applied the Random walk transition model, which follows $x_t^{(i)} = x_{t-1}^{(i)} + \epsilon_t^{(i)}$, $\epsilon_t^{(i)} \sim \mathbb{N}(0, \sigma^2 I)$. That is, each particle evolves by adding zero-mean Gaussian noise to its previous state. This reflects an assumption of temporal continuity in the hidden process, allowing smooth evolution over time.

• Update Step :

I originally created a model where independence was assume between USD and GBP, but upon reflection have come to realise that this is an unrealistic assumption. Hence I now model their joint distribution using a Multi-Variate Gaussian, and I calculated the likelihood of each particle given the observation y_t to be, $w_t^{(i)} = p(y_t|x_t^{(i)}) = \mathbb{N}(y_t|x_t^{(i)}, \Sigma)$, where Σ is the empirical 2×2 covariance matrix of the historical observations between USD and GBP. This allows the filter to take into account the possible correlation between USD and GBP exchange rate movements.

NOTE: I am making the assumption that USD and GBP are not independent, in my *observation model*, as they are both derived from macroeconomic data, but my *latent state evolution* is making the assumption that they are.

• Normalisation of weights :

After computing the likelihoods, I normalised the particle weights s.t. $\sum_i w_t^{(i)} = 1$. This prevents division by 0, and ensures numerical stability and valid probabilistic interpretation.

• Resampling step :

This step involved systematic re-sampling, and in my initial filtering case, I went with the simple use of Multi-nomial re-sampling. Here I select new particles with replacement, proportionally to their weights. Particles with High Likelihoods (i.e. those that have better alignment with observed data) are more likely to be retained. This step helps to focus computational resources on plausible regions of the state space and prevents high particle weight degeneracy, where a small number of particles dominate the weight distribution.

• Stored Filtered Estimate :

I then approximate the posterior mean estimate of the hidden state at time t by taking the empirical mean of the re-sampled particles;

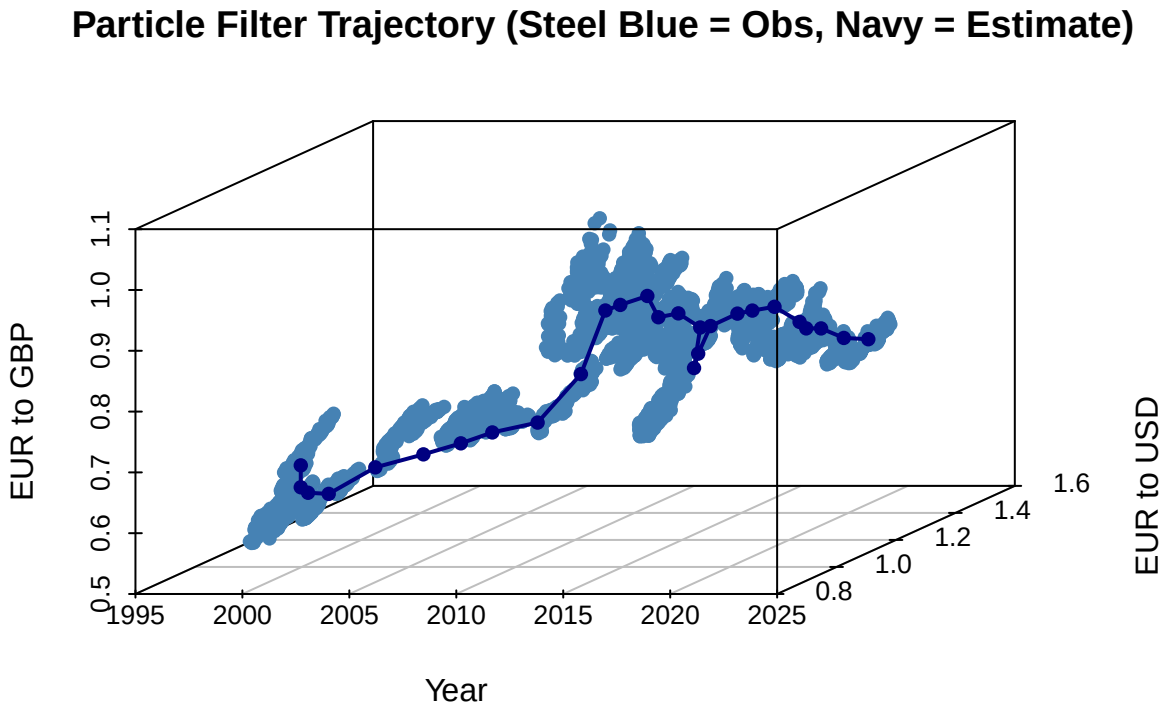
$$\hat{x}_t = \sum_i w_t^{(i)} x_t^{(i)}$$

This rough 3 step procedure results in the generation of a particle filter, of which tracks the conversion rates of USD and GBP vs EURO over the years, and reconstructs, visually, a likely hidden path from the observed data set.

Visualisation of the Impelemented Filter

A 3D visualisation of the plot and results of the particle filter can be observed below.

Note that the image displayed blow is obtained by running ***scatterplot3d*** function in R, for sake of visualisation on the knitted markdown file, but have also included an interactive plot using the ***rgl*** function within the markdown file itself, which allows for better visualisation of the filter estimate.



Measuring the success of the Filter

Following on from last weeks work, I wanted to now assess the effectiveness of this filtering method, by using various different measures.

RMSE

This measures the average magnitude of error between the filtered estimates and the observed value.

$$RMSE_t = \sqrt{(x_t^{est} - x_t^{obs})^2 + (y_t^{est} - y_t^{obs})^2}$$

Angle Error

This measures the angular difference between the estimated and observed vectors (USD, GBP) treating them as 2D vectors.

$$\theta_t = arccos\left(\frac{\vec{x}_t^{est} \times \vec{x}_t^{obs}}{||\vec{x}_t^{est}|| \times ||\vec{x}_t^{obs}||}\right)$$

Covariance Determinant/Trace

The sample covariance matrix $\Sigma_t \in \mathbb{R}^{2\times 2}$ of the particles at time t is:

$$\Sigma_t = \frac{1}{M-1}\sum_{i=1}^{M}(x_t^{(i)} - \hat{x}_t)(x_t^{(i)} - \hat{x}_t)^T$$

where $\hat{x}_t = \frac{1}{M}\sum_{i=1}^{M} x_t^{(i)}$ is the mean of the particles at time t.

The **Covariance Trace** tells us the magnitude of the uncertainty. It can calculated as $tr(\Sigma_t) = \sigma_{USD}^2 + \sigma_{GBP}^2$, and it is a total uncertainty or measure of how spread out the particles are. It is virtually the sum of the marginal variances.

The **Covariance Determinant** is related to the volume of the 2D uncertainty ellipsoid. It can be calculated as $det(\Sigma_t) = \sigma_{USD}^2\sigma_{GBP}^2 - Cov(USD, GBP)^2$. It measures the overall volume of uncertainty, and accounts for the correlation between Dimensions.

So virtually, whilst trace tells is how uncertain we are, the determinant tells us how independent and spread out that uncertainty is in a 2D space.

Building on understanding of the covariance determinant and trace

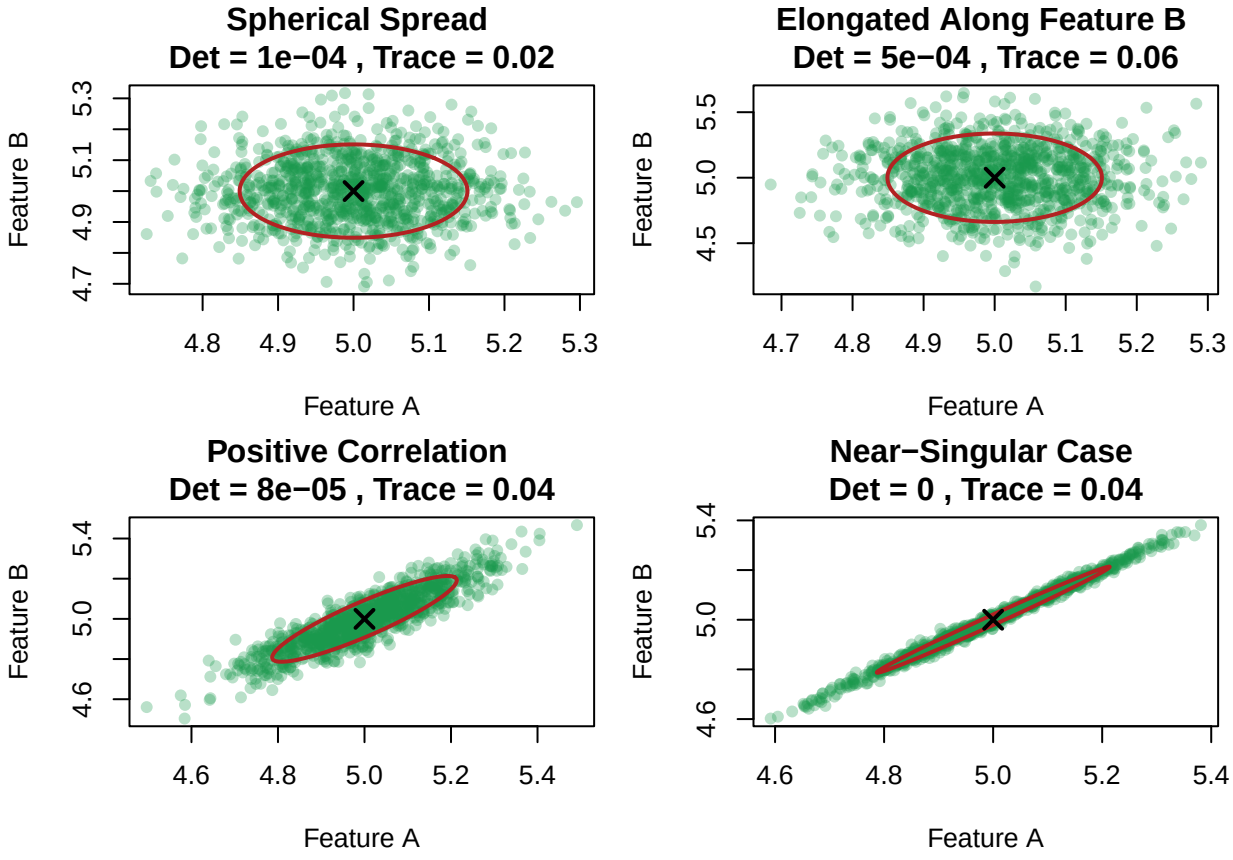
Following my meeting with Adam, I gathered that this was a rather useful and important measure of error, so wanted to devote a bit more time to understanding of these measures.

The Determinant was what appears to be the more interesting of the two, as I feel that the understanding of trace is rather simple to understand.

As addressed above, the Determinant of the covariance matrix has a very nic geometric interpretation; the idea that it is proportional to the volume of the **uncertainty ellipsoid** that surrounds the particle cloud.

So essentially, in the 2D case, (USD vs GBP), the determinant grows when the spread of particles increases in both directions, shrinks when the particles are tightly clustered or lie along a narrow direction, and equals zero when all particles lie on a straight line (perfect linear correlation - singular matrix).

I decided to simulate a simple 2D particle cloud, with a different covariance structure to understand this, computing and displaying the covariance matrix, its determinant, and plotting the corresponding ellipses. Note: Ellipse here represents the 68% confidence region (within 1 standard deviation).



Key Takeaways:

Determinant scales with the area of the ellipse.

A larger determinant $\implies$ wider, rounder ellipse $\implies$ more uncertainty.

A smaller determinant $\implies$ narrow, squeezed ellipse $\implies$ concentrated belief.

The Trace just sums the variance along the x and y axes (whatever that is) and doesnt account for correlation, unlike the determinant.

Calculating these measures for my Particle Filter

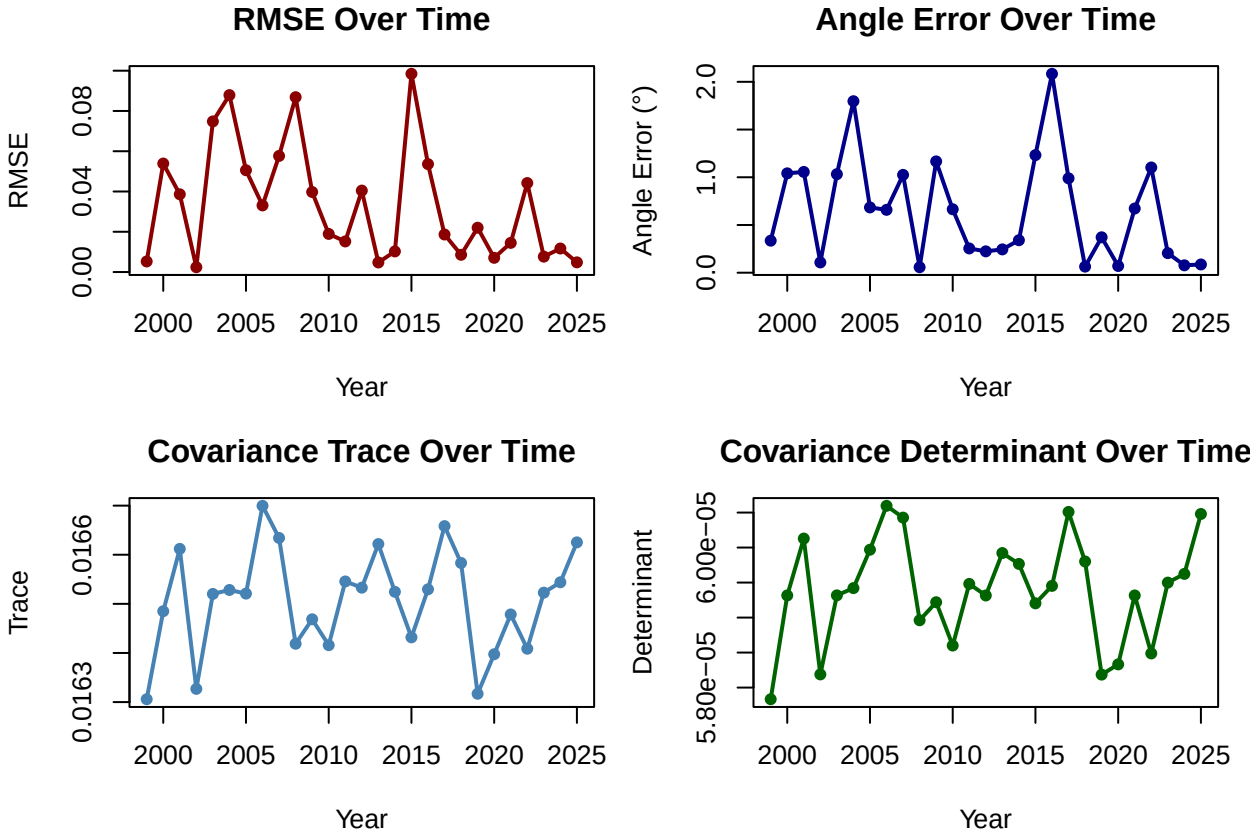
I hence went on to write some code that implemented these formulas to calculate each of these measures for my particle filtering method that was described and written above, given this particular data set.

The values for this procedure can be seen below at each time stamp (i.e. each year).

Table 1: Particle Filter Performance Metrics Over Time

Year	RMSE	AngleErrorDeg	Trace	Determinant
1999	0.0053	0.34	0.0163	5.8e-05
2000	0.0539	1.04	0.0165	5.9e-05
2001	0.0386	1.06	0.0166	6.0e-05
2002	0.0024	0.11	0.0163	5.8e-05
2003	0.0748	1.03	0.0165	5.9e-05
2004	0.0879	1.80	0.0165	5.9e-05
2005	0.0506	0.68	0.0165	6.0e-05
2006	0.0331	0.66	0.0167	6.1e-05
2007	0.0577	1.02	0.0166	6.0e-05
2008	0.0869	0.06	0.0164	5.9e-05
2009	0.0398	1.17	0.0165	5.9e-05
2010	0.0189	0.66	0.0164	5.9e-05
2011	0.0152	0.25	0.0165	5.9e-05
2012	0.0404	0.22	0.0165	5.9e-05
2013	0.0048	0.24	0.0166	6.0e-05
2014	0.0102	0.34	0.0165	6.0e-05
2015	0.0985	1.23	0.0164	5.9e-05
2016	0.0536	2.08	0.0165	5.9e-05
2017	0.0186	0.99	0.0167	6.1e-05
2018	0.0086	0.06	0.0166	6.0e-05
2019	0.0220	0.37	0.0163	5.8e-05
2020	0.0071	0.07	0.0164	5.8e-05
2021	0.0145	0.67	0.0165	5.9e-05
2022	0.0442	1.10	0.0164	5.8e-05
2023	0.0076	0.20	0.0165	5.9e-05
2024	0.0117	0.08	0.0165	6.0e-05
2025	0.0048	0.09	0.0166	6.0e-05

We can also visualise these results using graphs, as can be seen below.



Improvements to the Bootstrap Particle filter generated

Whilst this first model appeared to do well at estimating the hidden underlying true state, I definitely saw some areas of potential improvement.

First and foremost, my current Prediction step assumes that the latent state follows:

$$x_t = x_{t-1} + \epsilon_t, \quad \epsilon_t \sim \mathbb{N}(0, \sigma^2 I)$$

or in other words, assumes that USD and GBP are independent in the latent state evolution, as I used the simple random walk model. This is generally fine for exchange rates in the absence of stronger priors, but there are ways on which I could build from this.

Improved Model 1 - Mean Reverting Process + Dynamic Noise scaling

Upon some further investigation, it appears that in finance, things like interest rates or FX spreads that fluctuate but tend to return to a "central" level, and models tend to use something called a mean-reverting process.

A mean-reverting process is a stochastic process that tends to 'pull' the state back toward a long-term average or equilibrium over time.

A classic example, is the **AR(1)** process, which is as follows;

$$x_t = ax_{t-1} + \epsilon_t, \quad \epsilon_t \sim \mathbb{N}(0, \sigma^2), \text{ with } |a| < 1$$

so essentially, if $|a| = 1$ this is just a random walk process (with no mean-reversion), and if $|a| < 1$ then the process decays over time back toward zero (mean-reverting).

In most practical settings, and what I will be doing here, we want mean-reversion toward a non-zero mean, μ, and the model becomes;

$$x_t = \mu + a(x_{t-1} - \mu) + \epsilon_t$$

Given the nature of the data set, and the fact that I am looking at 3 of the arguably most stable currencies in the world, it seemed rational to calculate μ simply by taking the historical average for the conversion rates of both USD and GBP. Thus we have;

$$\mu = \begin{bmatrix} \mu_{USD} \\ \mu_{GBP} \end{bmatrix} = \begin{bmatrix} \frac{1}{T}\sum_{i=1}^{T} x_t^{USD} \\ \frac{1}{T}\sum_{i=1}^{T} x_t^{GBP} \end{bmatrix}$$

The other improvement I made was the introduction of dynamic noise scaling. This meant adjusting the magnitude of the process noise based on characteristics of the data, rather than just selecting a fixed 'noise level' like I did earlier on (saying sd=0.1), I can now adapt this to match the **historical volatility** of the data.

This seemed like a significant improvement, as it appears that in real-world data, like this exchange rate data set, volatility is not constant over time, and the different currencies of course can exhibit different levels of variability, so dynamic scaling here, allows my particle filter to better reflect the true level of uncertainty in the latent state evolution process. Furthermore, it avoids over-smoothing (which occurs if noise is too small) or excessive randomness (which occurs is noise is too large), as well as allowing the models to capture differences in volatility between the USD and GBP data.

Mathematically speaking, I implement code below to calculate the standard deviation σ, for both USD and GBP, $\sigma_{USD}, \sigma_{GBP}$ respectively, in the following manner.

Firstly I calculate the baseline noise magnitudes for each currency;

$$\hat{\sigma}_{USD} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_t^{USD} - \bar{x}_{USD})^2}$$

$$\hat{\sigma}_{GBP} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_t^{GBP} - \bar{x}_{GBP})^2}$$

and then go on to compute;

$$\sigma_{USD} = 0.5 \times \hat{\sigma}_{USD}, \sigma_{GBP} = 0.5 \times \hat{\sigma}_{GBP}$$

so the noise added to each latent component has a standard deviation proportional to its empirical volatility.

At this stage, we now have;

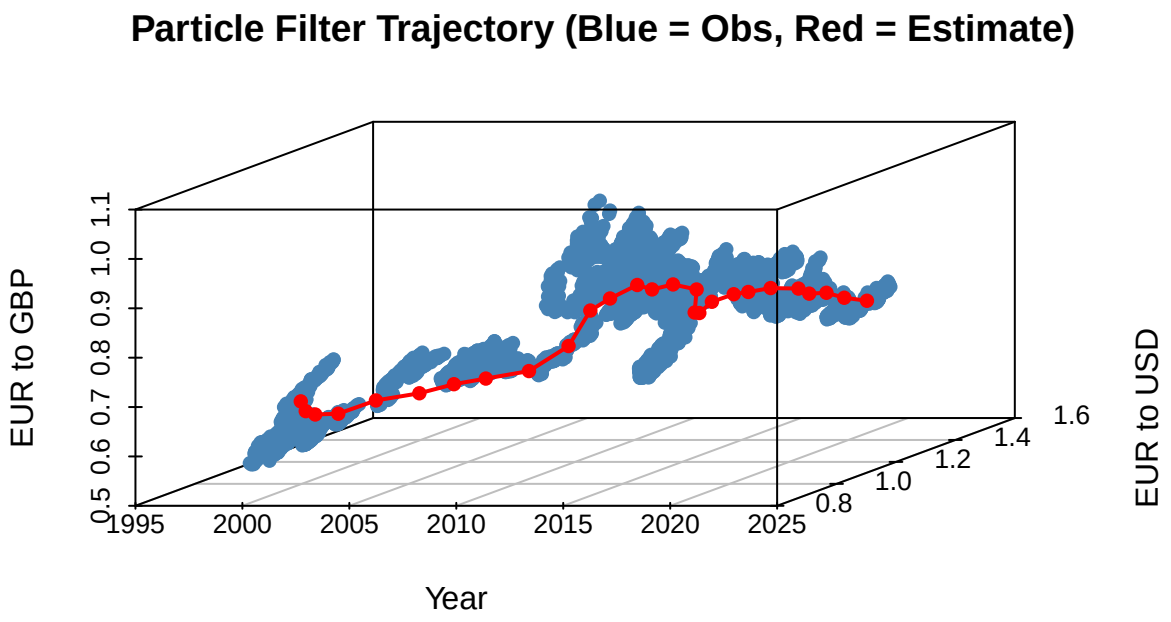
$$\epsilon_t^{(i)} \sim \mathbb{N}\left(\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}\sigma_{USD}^2 & 0\\0 & \sigma_{GBP}^2\end{bmatrix}\right)$$

for each particle, $i = 1, ..., M$.

And so the final update for each particle is now;

$$x_t^{(i)} = \mu + a \times (x_{t-1}^{(i)} - \mu) + \epsilon_t^{(i)}$$

The 3D plot of this updated filter can be seen below;

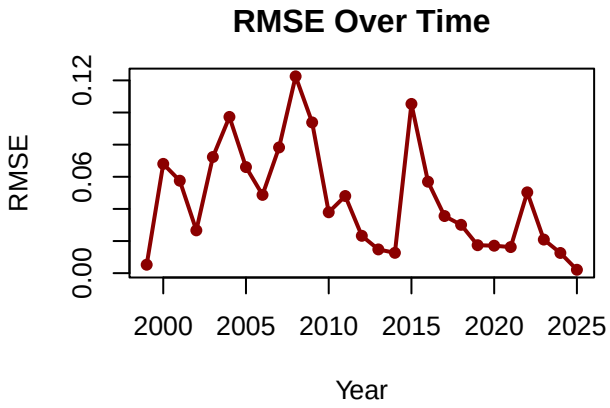


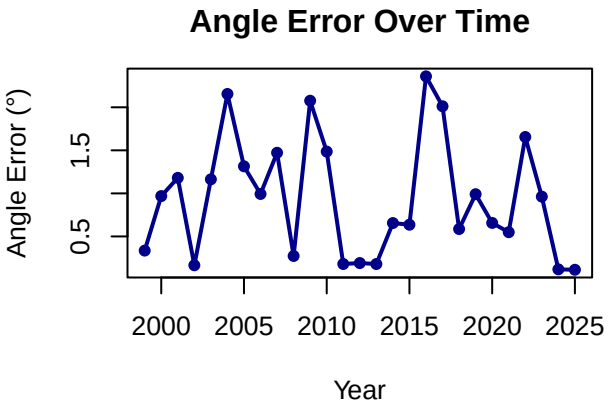
Computing Measures of Error for the Updated Filter

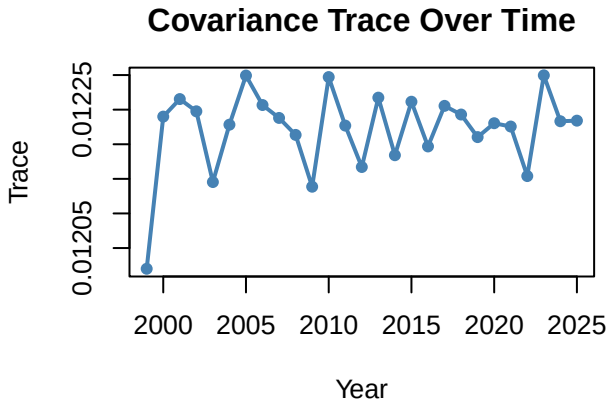
Now I compute the measurements of error as a result of implementing this filter, like before, to allow for comparison.

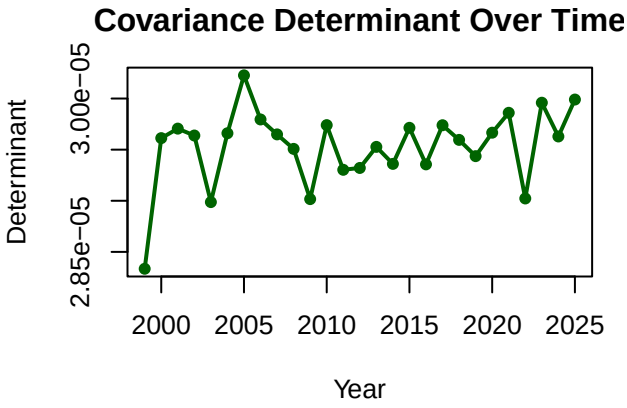
Table 2: Particle Filter Performance Metrics Over Time

Year	RMSE	AngleErrorDeg	Trace	Determinant
1999	0.0053	0.34	0.0120	2.8e-05
2000	0.0680	0.97	0.0122	3.0e-05
2001	0.0576	1.18	0.0123	3.0e-05
2002	0.0267	0.17	0.0122	3.0e-05
2003	0.0723	1.16	0.0121	2.9e-05
2004	0.0973	2.15	0.0122	3.0e-05
2005	0.0660	1.31	0.0123	3.0e-05
2006	0.0487	0.99	0.0123	3.0e-05
2007	0.0783	1.47	0.0122	3.0e-05
2008	0.1225	0.27	0.0122	3.0e-05
2009	0.0938	2.08	0.0121	2.9e-05
2010	0.0379	1.49	0.0123	3.0e-05
2011	0.0480	0.18	0.0122	2.9e-05
2012	0.0232	0.19	0.0122	2.9e-05
2013	0.0148	0.18	0.0123	3.0e-05
2014	0.0127	0.66	0.0122	2.9e-05
2015	0.1053	0.64	0.0123	3.0e-05
2016	0.0569	2.36	0.0122	2.9e-05
2017	0.0356	2.01	0.0123	3.0e-05
2018	0.0301	0.59	0.0122	3.0e-05
2019	0.0174	0.99	0.0122	2.9e-05
2020	0.0171	0.66	0.0122	3.0e-05
2021	0.0162	0.55	0.0122	3.0e-05
2022	0.0503	1.65	0.0122	2.9e-05
2023	0.0209	0.96	0.0123	3.0e-05
2024	0.0126	0.12	0.0122	3.0e-05
2025	0.0021	0.11	0.0122	3.0e-05


RMSE Over Time
RMSE
0.00
0.06
0.12
2000
2005
2010
2015
2020
2025
Year


Angle Error Over Time
Angle Error (°)
0.5
1.5
2000
2005
2010
2015
2020
2025
Year


Covariance Trace Over Time
Trace
0.01205
0.01225
2000
2005
2010
2015
2020
2025
Year


Covariance Determinant Over Time
Determinant
2.85e−05
3.00e−05
2000
2005
2010
2015
2020
2025
Year

Improved Model 2 - Changing the Resampling Strategy + ESS

As addressed earlier in the literature, my original filter used a multinomial resampling strategy, one of the more basic and commonly used resampling methods in particle filters.

We discussed how this works previously, but for sake of simplicity upon re-reads, I will explain briefly again here.

This strategy involves drawing M indices from a discrete probability distribution over $\{1, 2, ..., M\}$ where each index i is chosen with a probability w_i, the normalised weights of particle i. These weights are formed by calculation of the likelihoods of each particle given the observed exchange rates. Following this distribution, we should expect to see that high-weight particles are likely to be selected multiple times, whilst low-weight particles much fewer, if at all.

This is a Monte Carlo approximation of replacing the current particle set with a new one, that focuses computational effort on the most likely hypotheses.

We then replace the particle set with the newly selected ones, and reset all weights to uniform again, because after re-sampling, all particles are equally likely again before the next prediction step.

Issues with this Method in my Original Particle Filter

• In the original implementation, I re-sampled at every time stamp, no matter how diverse the particles were.

• This could have likley lead to sample impoverishment - a problem where diversity among particles quickly collapses, espceially in low-noise settings or with small M.

Solution to Address these Issues

There are a few solutions to this strategy, but the one I decided to focus on, was by using **adaptive re-sampling** via **Effective Sample Size (ESS)**.

Firstly, we note that **ESS** is a measure of how 'diverse' the particle weights are at a given time stamp, and essentially helps us answer the question *'how many particles are effectively contributing to the estimate?'*

ESS is defined as;

$$ESS = \frac{1}{\sum_{i=1}^{M} w_i^2}$$

Where M is the total number of particles, and w_i is the normalised weight of particle i.

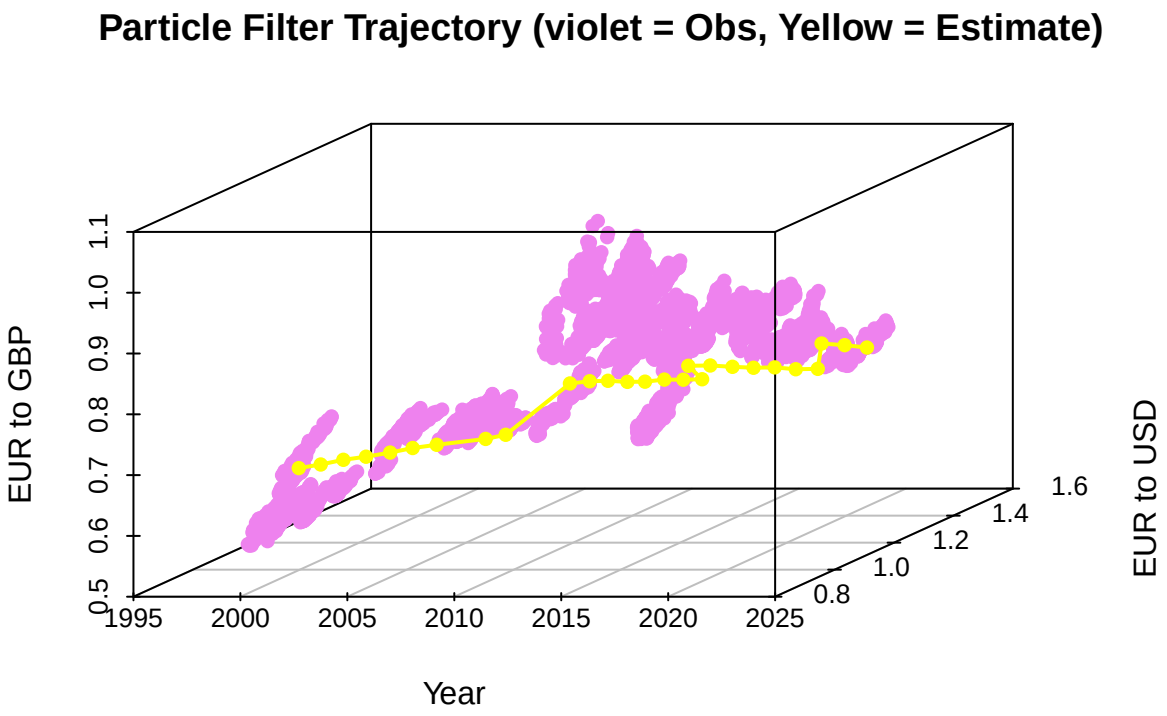
If all weights are equal: $ESS = M \implies$ no degeneracy

If only one particle has non-zero weight: $ESS = 1 \implies$ full degeneracy

So low ESS means high-degeneracy, i.e. most particles are useless.

ESS-based re-sampling avoids the issue of re-sampling at every time stamp, and only re-samples when needed, (i.e. when ESS is low). This saves computational effort, and reduces unnecessary randomness. For this strategy to work, we must define a threshold for which we decide if a particle is contributing meaningfully or not. It is common practice to set this threshold to be $\frac{M}{2}$, i.e. re-sample only if fewer than 50% of particles are contributing meaningfully. Once again, this ensures that we only replace particles when the estimate is actually degrading.

The 3D plot of this updated filter can be seen below;

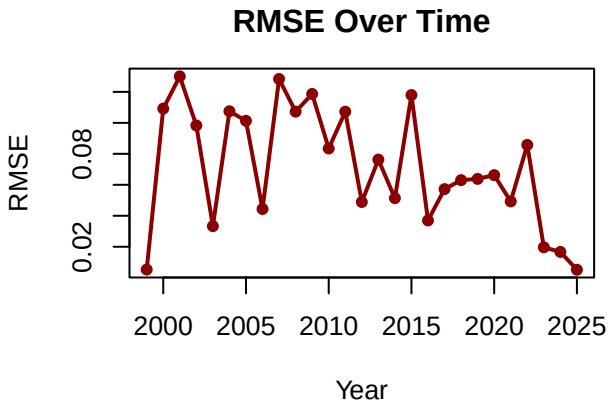


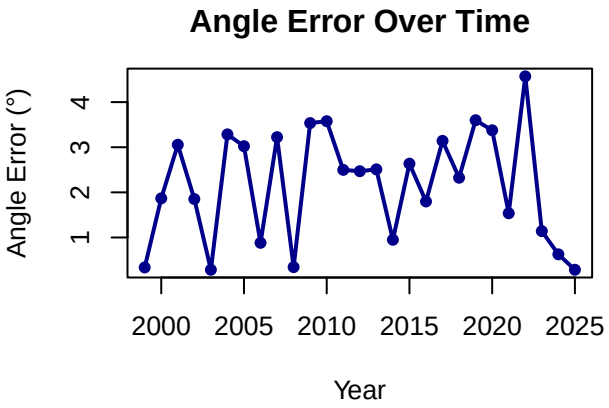
Computing Measures of Error for the Updated Filter

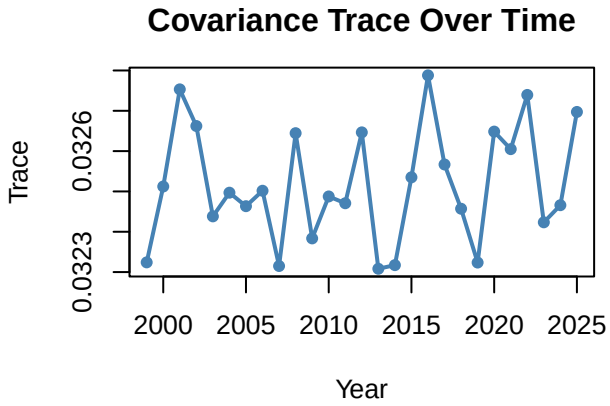
Now I compute the measurements of error as a result of implementing this filter, like before, to allow for comparison.

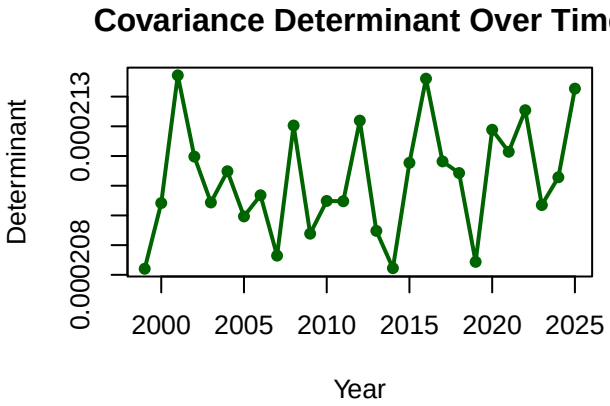
Table 3: Particle Filter Performance Metrics Over Time

Year	RMSE	AngleErrorDeg	Trace	Determinant
1999	0.0053	0.34	0.0323	0.000208
2000	0.1092	1.87	0.0325	0.000210
2001	0.1300	3.06	0.0328	0.000215
2002	0.0983	1.85	0.0327	0.000212
2003	0.0333	0.28	0.0324	0.000210
2004	0.1075	3.28	0.0325	0.000211
2005	0.1013	3.02	0.0325	0.000210
2006	0.0444	0.88	0.0325	0.000211
2007	0.1283	3.22	0.0323	0.000209
2008	0.1072	0.34	0.0326	0.000213
2009	0.1187	3.54	0.0324	0.000209
2010	0.0834	3.58	0.0325	0.000210
2011	0.1072	2.50	0.0325	0.000210
2012	0.0489	2.47	0.0326	0.000213
2013	0.0763	2.51	0.0323	0.000209
2014	0.0514	0.95	0.0323	0.000208
2015	0.1179	2.64	0.0325	0.000212
2016	0.0370	1.80	0.0328	0.000215
2017	0.0572	3.14	0.0326	0.000212
2018	0.0630	2.32	0.0325	0.000211
2019	0.0638	3.60	0.0323	0.000208
2020	0.0662	3.38	0.0326	0.000213
2021	0.0493	1.54	0.0326	0.000212
2022	0.0857	4.57	0.0327	0.000214
2023	0.0196	1.14	0.0324	0.000210
2024	0.0167	0.63	0.0325	0.000211
2025	0.0051	0.28	0.0327	0.000214


RMSE Over Time
RMSE
0.02
0.08
2000
2005
2010
2015
2020
2025
Year


Angle Error Over Time
Angle Error (°)
1
2
3
4
2000
2005
2010
2015
2020
2025
Year


Covariance Trace Over Time
Trace
0.0323
0.0326
2000
2005
2010
2015
2020
2025
Year


Covariance Determinant Over Time
Determinant
0.000208
0.000213
2000
2005
2010
2015
2020
2025
Year