DAML Assignment 2023

Imogen Stephenson 30566835

As part of this assignment, I answered questions 1, 2, and 5.

1 Question 1

The general formula for the true value of the data is:

$$\hat{N}_i = N_{0_i} e^{-\alpha(t_i - t_{0_i})}, \tag{1}$$

Where $N_i = N_i(t)$ is the mass of radioactive material in the i-th widget at t_i seconds from when it was manufactured, with uncertainty σ_{N_i}. Both t_{0_i} (actual time when widget was manufactured) and N_{0_i} (initial radioactive mass when manufactured) are constants for each widget.

The relevant data obtained for all widgets during the investigation were N, t and σ_N.

The following are known variables in the system:

Number of widgets = 100

Number of detectors = 3

Maximum initial mass of radioactive material = 20g

Time taken to manufacture widgets = 35 days = 35 × 24 × 60 × 60 seconds = $30, 240, 000$ seconds.

Time delay between final manufacture and first measurement = $14days = 14 \times 24 \times 60 \times 60$ seconds =$1, 209, 600$ seconds.

Number of observations per widget = 1.

First, I plotted the raw data. This provided a bit of context to the data I was working with.

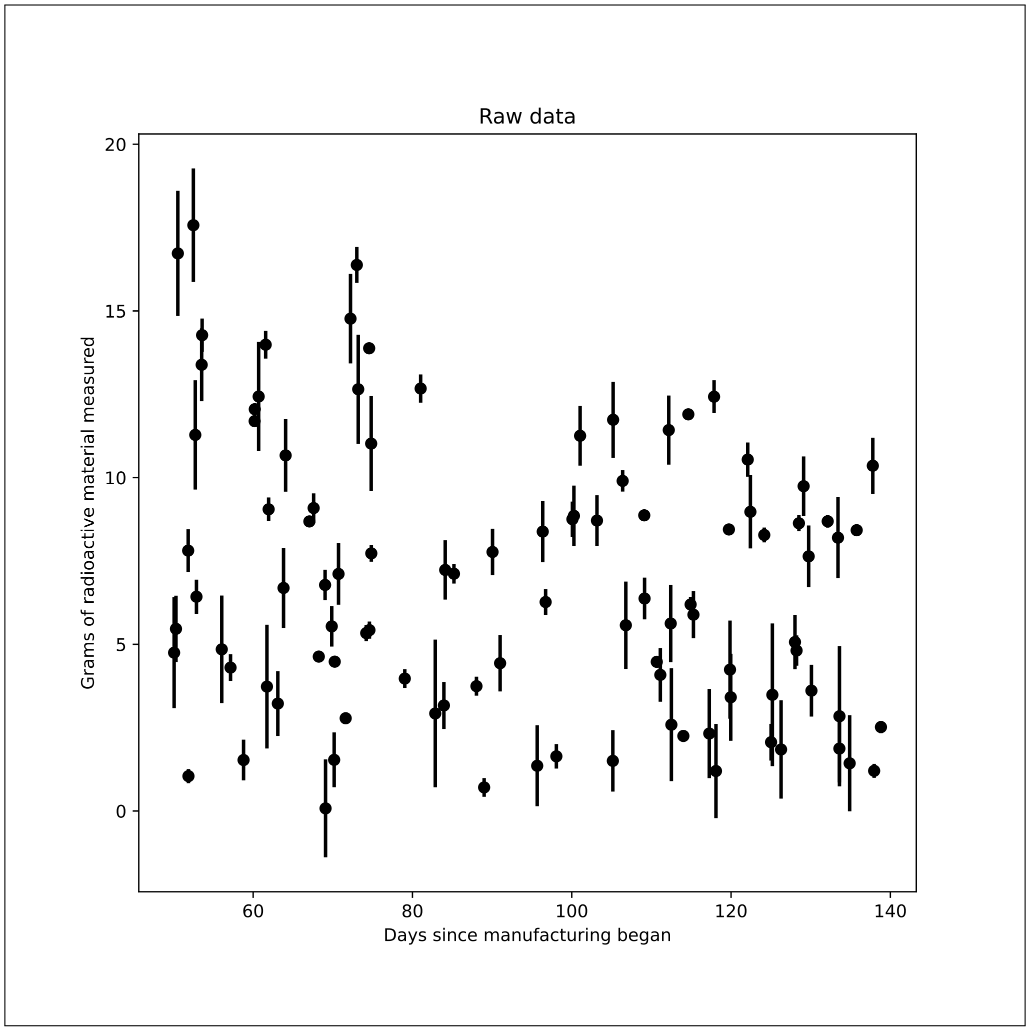


Figure 1

I then considered the model for the data, to run inference on α. This parameter was constant for the entire dataset. However, each widget measured had its own N_0 and t_0 values.

In general, we assume the measurement time has no uncertainty, meaing the likelihood for the data can be written as:

$$\mathcal{L} = p(N, \theta | t, \sigma_{N(t)}) = p(N | t, \sigma_{N(t)}, \theta) \times p(\theta), \tag{2}$$

where θ are the model parameters α, N_0 and t_0.

I note that:

$$p(N_i | t, \sigma_{N_i}), \theta) = \frac{1}{\sqrt{2\pi\sigma_{N_i}^2}} e^{-\frac{\left(N_i - N_{0_i} e^{-\alpha(t_i - t_{0_i})}\right)^2}{2\sigma_{N_i}^2}}, \text{and} \tag{3}$$

$$p(\theta) = p(N_{0_i}) \times p(t_{0_i}) \times p(\alpha). \tag{4}$$

1.1 Setting the priors

We are able to set an upper and lower limit on N_{0_i} that are similar for each widget - N_{0_i} cannot be less than 0, and we know it cannot be greater than 20g=$N_{0_{max}}$. Therefore, I set the prior

$$N_{0_i} \sim \mathcal{U}(0, N_{0_{max}}). \tag{5}$$

Further, time of manufacture t_{0_i} is also bound between two values, 0 and $t_{0_{max}} = 35\text{days} = 30,240,000\text{seconds}$. Therefore,

$$t_{0_i} \sim \mathcal{U}(0, t_{0_{max}}). \tag{6}$$

Finally, we know that α must be positive, as the radioactive material can only decay. Further, we can estimate the order of magnitude for α by assuming the material won't completely decay within a few seconds.

If N_0 were 20g, we will assume it would take more than one day to decay to less than 1 gram (considering there were 14 days delay between manufacture and observation, if this were not the case, one would expect a lot more "0" measurements).

In one day, there are 86,400 seconds. Assuming $N_0 = 20g$, for there to be 1 gram left after 86,400 seconds from manufacture, α would need to be 3.46728×10^{-5}. To simplify this, the upper bound for α was set to be $\alpha_{max} = 10^{-5}$.

As there was no other guidance as to the value of α, I set its prior to be a uniform distribution between 0 and this upper limit:

$$\alpha \sim \mathcal{U}(0, \alpha_{max}). \tag{7}$$

Therefore,

$$p(t_{0_i}) = \frac{1}{t_{0_{max}}} \tag{8}$$

$$p(N_{0_i}) = \frac{1}{N_{0_{max}}} \tag{9}$$

$$p(\alpha) = \frac{1}{\alpha_{max}}. \tag{10}$$

The total likelihood for **all** the data is therefore:

$$\mathcal{L} = \prod_{i=1}^{N_{\text{widgets}}} p(N_i|t, \sigma_{N_i}), \theta) \tag{11}$$

$$\mathcal{L} = \prod_{i=1}^{N_{\text{widgets}}} \left(\frac{1}{t_{0_{max}} N_{0_{max}} \alpha_{max}} \frac{1}{\sqrt{2\pi\sigma^2_{N_i}}} e^{-\frac{\left(N_i - N_{0_i} e^{-\alpha(t_i - t_{0_i})}\right)^2}{2\sigma^2_{N_i}}} \right). \tag{12}$$

1.2 Part I: Inferring α

To conduct inference on α (and other parameters in θ), I used cmdstanpy.

The relevant stan model was as follows:

```
data {
int<lower=0> Num_widgets;
vector[Num_widgets] t_measured;
vector[Num_widgets] N_measured;
vector[Num_widgets] sigma_N_measured;
real N_0_max;
}
parameters {
real<lower=0,upper=0.01> alpha;
vector<lower=0, upper=N_0_max> [Num_widgets] N_0;
vector<lower=0,upper=35*24*60*60> [Num_widgets] t_0;
}
model {
alpha   normal(0,0.00001);
for (i in 1:Num_widgets) {
N_0[i] uniform(0,N_0_max);
t_0[i] uniform(0,35*24*60*60);
N_measured[i]   normal(N_0[i] * exp(-alpha * (t_measured[i]- t_0[i])),sigma_N_measured[i]);
}
}
```

I sampled the parameters 1000 times, after an initial 1000 step burn-in. The chain plot for α is shown below.

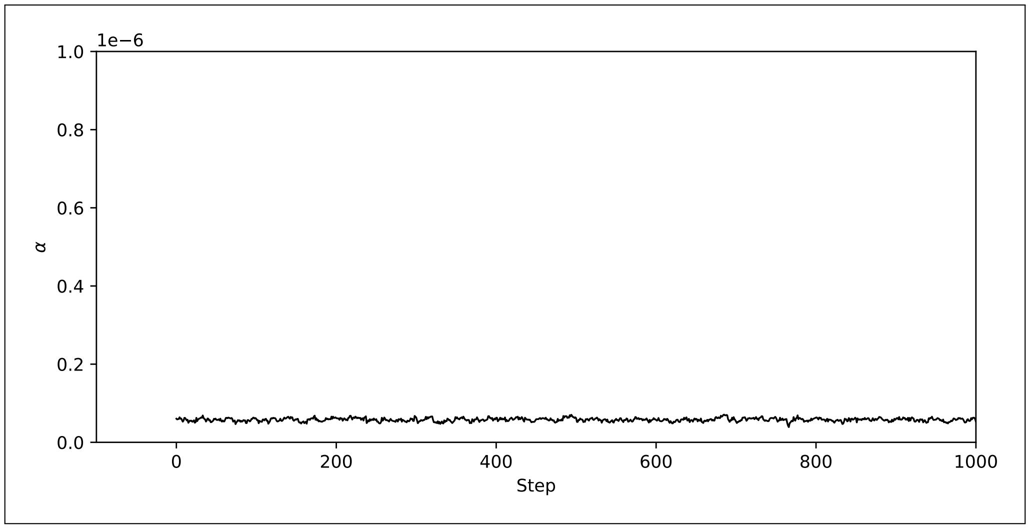


Figure 2

Upon inspection, and considering the much higher bounds, the chain appears to converge. Further, the $\hat{R}$ value for this chain is 1.001550, which supports its reliability. Therefore, I considered the chain's mean value and standard deviation as reliable representations of the most likely value of α and its uncertainty.

Mean value $= 5.795162325 \times 10^{-8}$,

Standard deviation $= 4.433872207 \times 10^{-9}$.

1.3 Part II: Including a bias

1.3.1 Which detector is biased? (joint inference)

First, I established which detector was most likely the one with a bias.

To do this, I modified the model from above to contain another dimension - I introduced a new bias term that, unlike N, N_0, t and t_0, did not depend on the widget but instead depended on the widgets' detector.

This meant that the stan code had an additional level to the model - within the 'for' loop over the widgets, there were 3 possible models allocated depending on the detector. Given the different models for the detectors A, B and C, the total likelihood can now be written using an index for each widget that shows which detector was used: $i_D = 1_B, 2_A, 3_B, 4_C, 5_C....$

These models reflected the new true model:

$$\hat{N_{i_D}} = N_{0_i} e^{-\alpha(t_i - t_{0_i})} + b_D \tag{13}$$

Where b_D is the bias on a detector $D = \text{detector} A, B, \text{or} C$.

The total likelihood of the data therefore becomes:

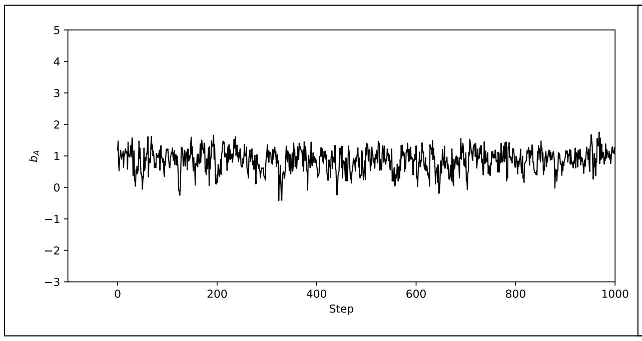
$$\mathcal{L} = \prod_{i_D}^{N_{\text{widgets}}} \left(\frac{1}{t_{0_{max}} N_{0_{max}} \alpha_{max}} \frac{1}{\sqrt{2\pi\sigma_{N_i^2}}} e^{-\frac{\left(N_i - \left(N_{0_i} e^{-\alpha(t_i - t_{0_i})} + b_D\right)\right)^2}{2\sigma^2_{N_i}}} \right). \tag{14}$$

This was implemented into the stan code as below.

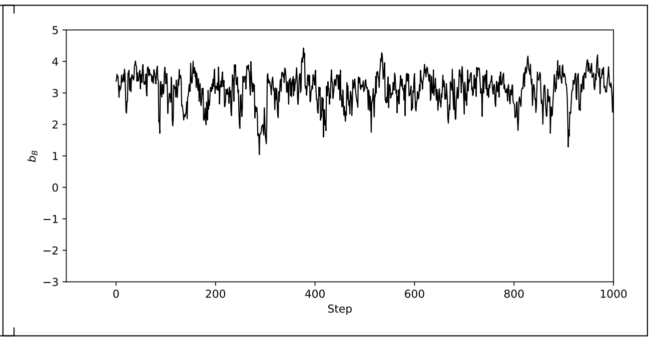
```
data {

int<lower=0> Num_widgets;

vector[Num_widgets] t_measured;

vector[Num_widgets] N_measured;

vector[Num_widgets] sigma_N_measured;

real N_0_max;

vector[Num_widgets] detector;

}

parameters {

real<lower=0,upper=0.000001> alpha;

real b1;

real b2;

real b3;

vector<lower=0, upper=N_0_max>[Num_widgets] N_0;

vector<lower=0,upper=35*24*60*60>[Num_widgets] t_0;

}

model {

b1  normal(0,5);

b2  normal(0,5);

b3  normal(0,5);

alpha  uniform(0,0.00001);

for (i in 1:Num_widgets) {

if (detector[i]==1) {

N_0[i] uniform(0,N_0_max);

t_0[i] uniform(0,35*24*60*60);

N_measured[i]  normal(N_0[i] * exp(-alpha * (t_measured[i]- t_0[i])) + b1,sigma_N_measured[i]);

}

if (detector[i]==2) {

N_0[i] uniform(0,N_0_max);
t_0[i] uniform(0,35*24*60*60);

N_measured[i]   normal(N_0[i] * exp(-alpha * (t_measured[i]- t_0[i]))+b2,sigma_N_measured[i]);

}

if (detector[i]==3) {

N_0[i] uniform(0,N_0_max);

t_0[i] uniform(0,35*24*60*60);

N_measured[i]   normal(N_0[i] * exp(-alpha * (t_measured[i]- t_0[i]))+b3,sigma_N_measured[i]);

}

}

}
```

1.3.2 Results

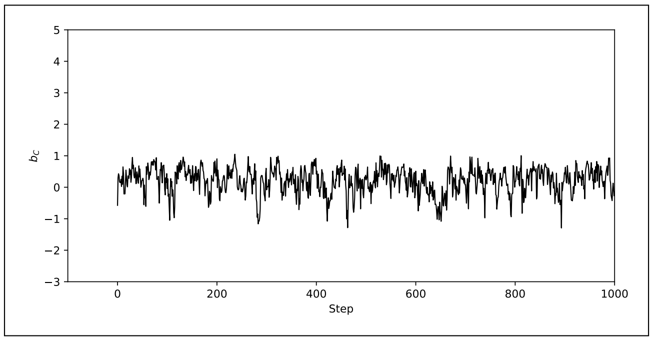
The chain plots for bias terms $b_{D=A,B,C}$ are below.



(a)



(b)



(c)

These chains appear to be less convergent, with values seeming to consistently span more than one unit. however, the chains do appear consistently around a single value, and their overall reliability is reflected in their $\hat{R}$ values: 1.000070 for b_A, 1.007750 for b_B, and 1.001150 for b_C. Therefore, we can consider these chains to be significant.

The bias terms sampled for each detector had the following means and standard deviations:

$$b_A = 0.850713733115 \pm 0.3798374628073978g \tag{15}$$
$$b_B = 3.06492728 \pm 0.505629990014439g \tag{16}$$
$$b_C = 0.22553132105419996 \pm 0.4319163857390285g. \tag{17}$$

While I could have implemented a mixture model with two models (one with a bias and one without), this preliminary method seems to have already confirmed which detector has a bias. The bias term for detector C is the only one that does not extend to 0 within three standard deviations. Therefore, I can confidently say that detector B is the biased detector (given that only one is biased).

It is worth noting that our mean α value is $8.150583315 \times 10^{-8}$ with a standard deviation of $6.662486347768082 \times 10^{-9}$, which are both slightly greater than their previous values. Therefore the presence of this bias has clear impact on the interpretation of the data.

However, I will not call these values the "final" values for b_B nor α yet, as my models so far have not been quite correct. Now, I know to use the original, without-bias model for detectors A and C, and the with-bias model for B.

1.3.3 Final model (bias on detector B)

My new likelihood equation is the same as before, but now $b_A = b_C = 0$, and I define $b_B = b$.

Implementing this new model, my stan code becomes:

```
data {

int¡lower=0¿ Num_widgets;

vector[Num_widgets] t_measured;

vector[Num_widgets] N_measured;

vector[Num_widgets] sigma_N_measured;

real N_0_max;

vector[Num_widgets] detector;

}

parameters {

real¡lower=0,upper=0.000001¿ alpha;

real b;

vector¡lower=0, upper=N_0_max¿[Num_widgets] N_0;

vector¡lower=0,upper=35*24*60*60¿[Num_widgets] t_0;

}

model {

b   normal(0,5);
alpha   uniform(0,0.00001);
for (i in 1:Num_widgets) {
if (detector[i]==1) {
N_0[i] uniform(0,N_0_max);
t_0[i] uniform(0,35*24*60*60);
N_measured[i]   normal(N_0[i] * exp(-alpha * (t_measured[i]- t_0[i])),sigma_N_measured[i]);
}
if (detector[i]==2) {
N_0[i] uniform(0,N_0_max);
t_0[i] uniform(0,35*24*60*60);
N_measured[i]   normal(N_0[i] * exp(-alpha * (t_measured[i]- t_0[i]))+b,sigma_N_measured[i]);
}
if (detector[i]==3) {
N_0[i] uniform(0,N_0_max);
t_0[i] uniform(0,35*24*60*60);
N_measured[i]   normal(N_0[i] * exp(-alpha * (t_measured[i]- t_0[i])),sigma_N_measured[i]);
}
}
}
```

This stan code provided the following chain plots for α and b.

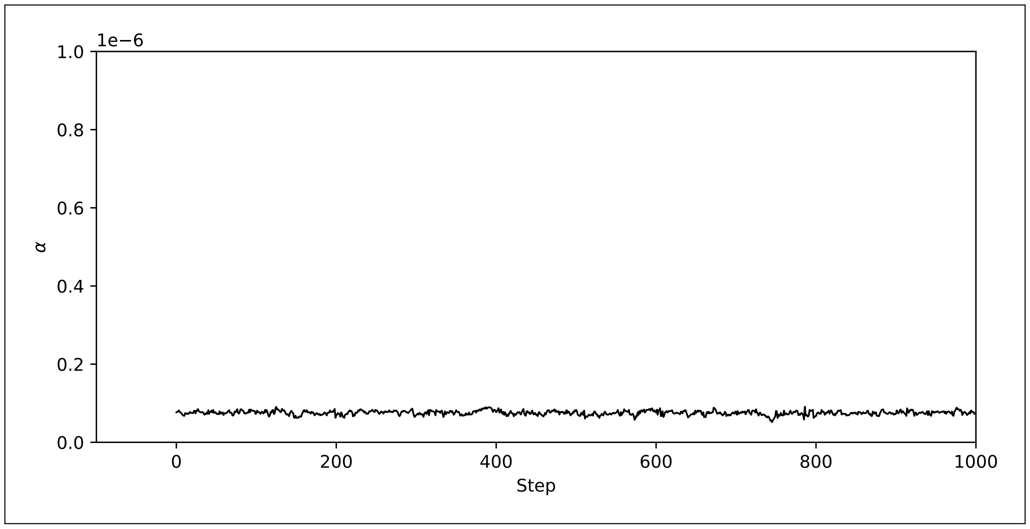


Figure 4

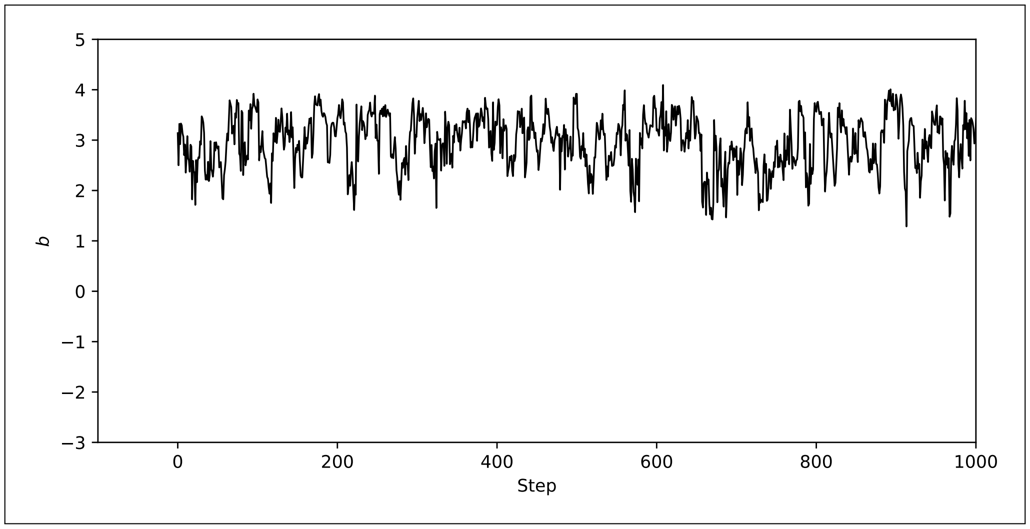


Figure 5

The chain for α appears to converge, and this is supported by the fact that its mean $\hat{R}$ value is 1.008210.

However, the chain for b appears significantly less convergent than the chain for α. To assess whether it is reliable or not (as the visual effect may involve artefacts of the axis limits), we can consider the $\hat{R}$ value, which is 1.023030. While this is not ideal, it is below 1.05 so it has some degree of credibility. Further, a relatively large uncertainty is anticipated for these bias values, as we are also sampling other parameters in the model.

Choosing to rely on our inference for b, we ge the following results:

- The mean bias value from the chain was $2.865356627 \pm 0.5660971450111738$; and
- The mean value for α was $7.503173255 \times 10^{-8} \pm 5.8378395522877715 \times 10^{-9}$.

Therefore, my inferred answers to this question are that detector B had the bias of $2.87g \pm 0.57g$, and the radioactive material's rate of decay α was $(7.50 \pm 0.58) \times 10^{-8}\ s^{-1}$.

The final value for the rate of decay, α, was larger than originally anticipated by about 50%. This makes sense, as if one of the detectors was measuring more radioactive mass than truly existed, this means more of the mass must have decayed within the same time period, meaning the decay was faster and α is greater.

2 Question 2

2.1 Part I: Finding periodicity

I first plotted the raw data in the given file, which containted the brightness of stars over a certain period of time. The time units are assumed to be seconds.

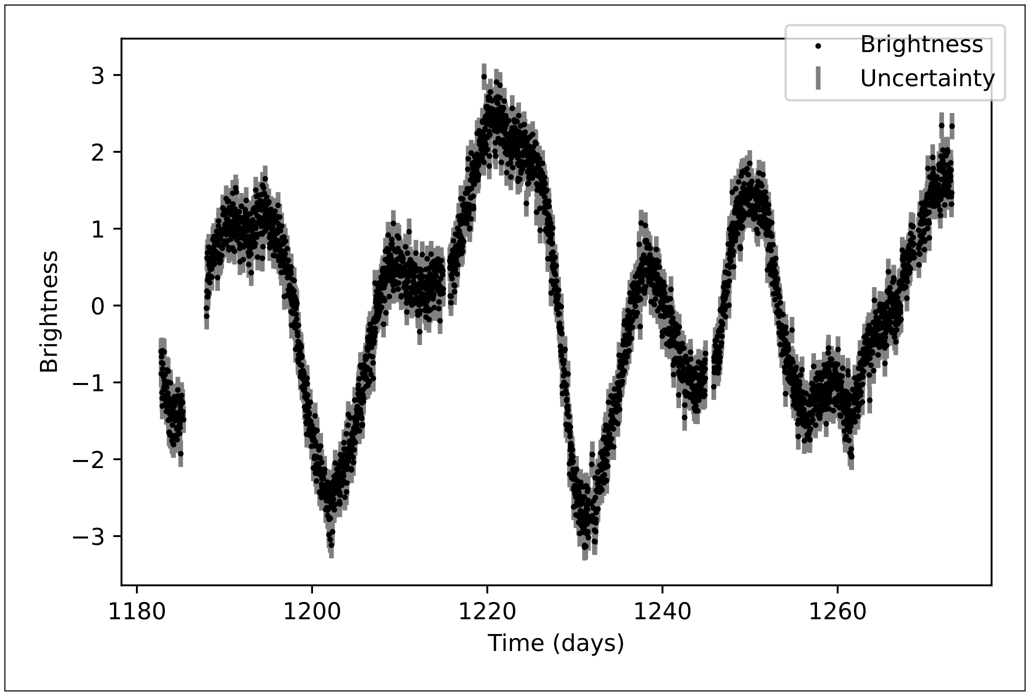


Figure 6

Upon inspection, there appears to be more than one form of periodicity in the data. Further, the data seems to be incomplete - there is a 'break' in the data near the start of the plot. Therefore, instead of using a Fourier transform to detect present periodicities, I will convert this data into a Lob-Scargle Periodogram.

2.1.1 Lomb-Scargle Periodogram

The mean time difference between consecutive data points is ~ 0.0236, which means the data can't reliably present a periodic trend with a period less than this amount (or an angular frequency greater than $\frac{2\pi}{0.0236} \approx$ 266.24 days^{-1}.

Further, the data cannot demonstrate periodicity with a period greater than the range. Therefore, I chose the upper limit for the period to be 100 s (a frequency of $\frac{2\pi}{100} \approx 0.063$ days^{-1}

Therefore, for my initial Lomb-Scargle periodogram, I will approximate these limits for possible periods to be 0.02 s and 100 s (for frequencies, 0.06 days^{-1} and 270 days^{-1}).

To actually create the Lomb-Scargle periodogram, I used the scipy.signal.lobscargle function in python. This process involved the calculation of periods from angular frequencies, so I generated 5000 angular frequency points within the afore-mentioned range. This is the resulting plot of the possible frequencies of periodicity, and their associated powers in the data.

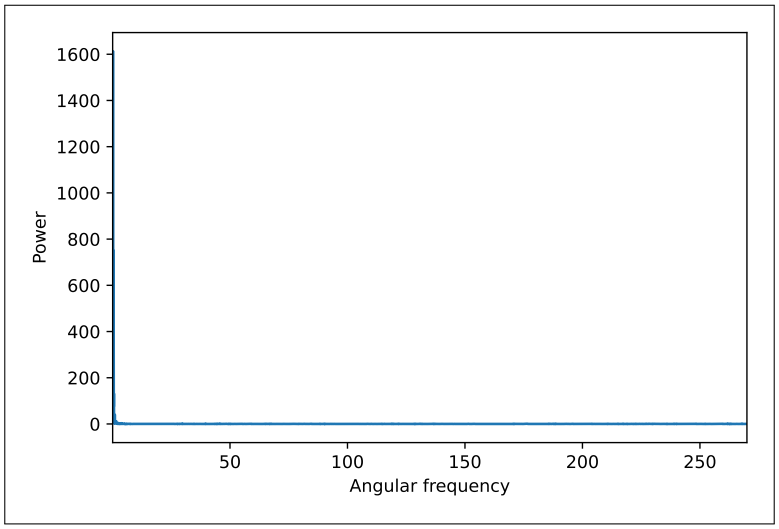


Figure 7

The limits set out above have proven to be rather extreme. Simply by inspection, I can comfortably claim that there is no significant periodicity with a frequency higher than 10 days^{-1}. By instead generating 5000 angular frequency points in the range 0.06 to 10 days^{-1}, the following plot was obtained:

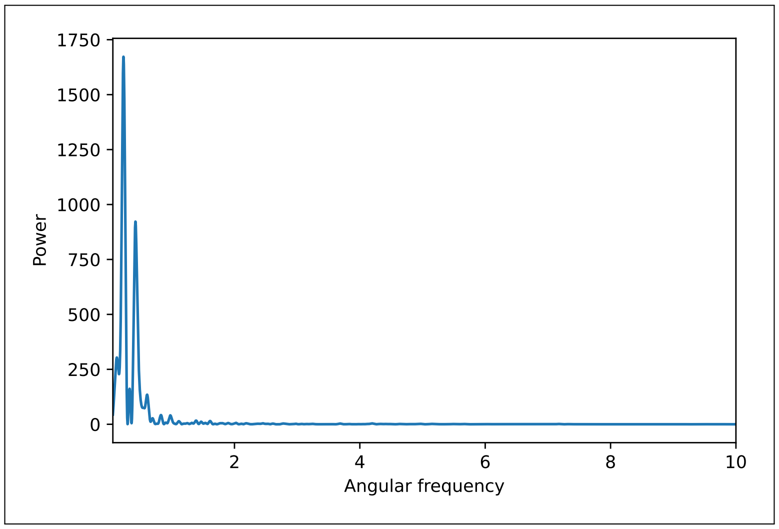


Figure 8

Again, it seems we can refine our bounds to refine the more significant frequency values. Therefore, I finally generated 5000 angular frequency points in a range of 0.06 to 2 days^{-1}.

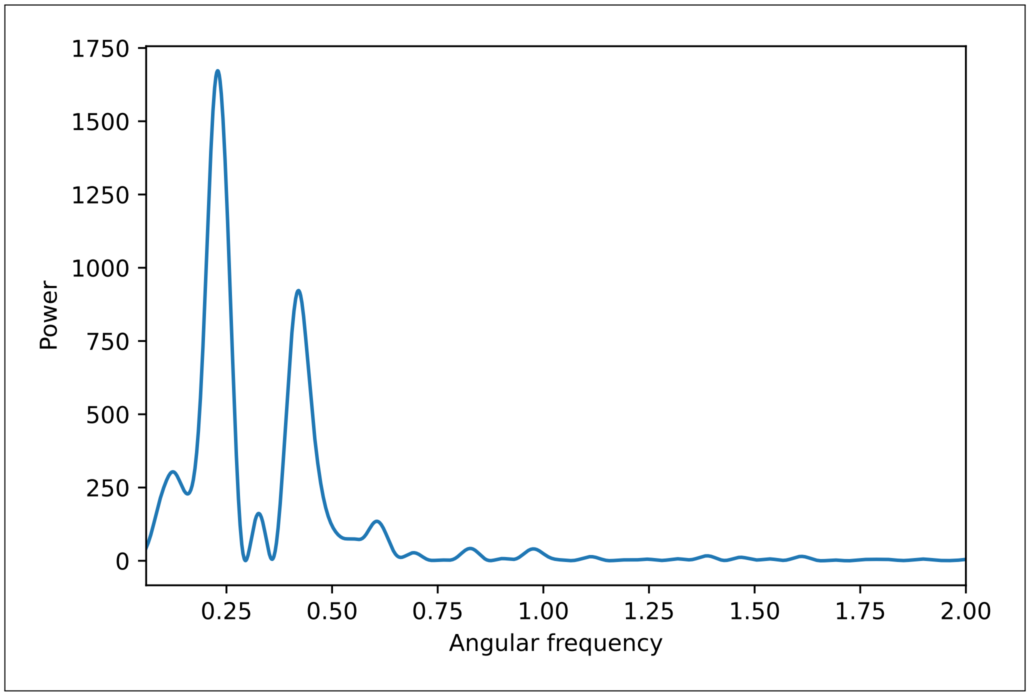


Figure 9

Using these frequency points, the associated period for each was found. The powers were then plotted against the periods to obtain the final Lomb-Scargle periodogram. From this plot, we can see there are several strong peaks, suggesting there is periodicity in our data.

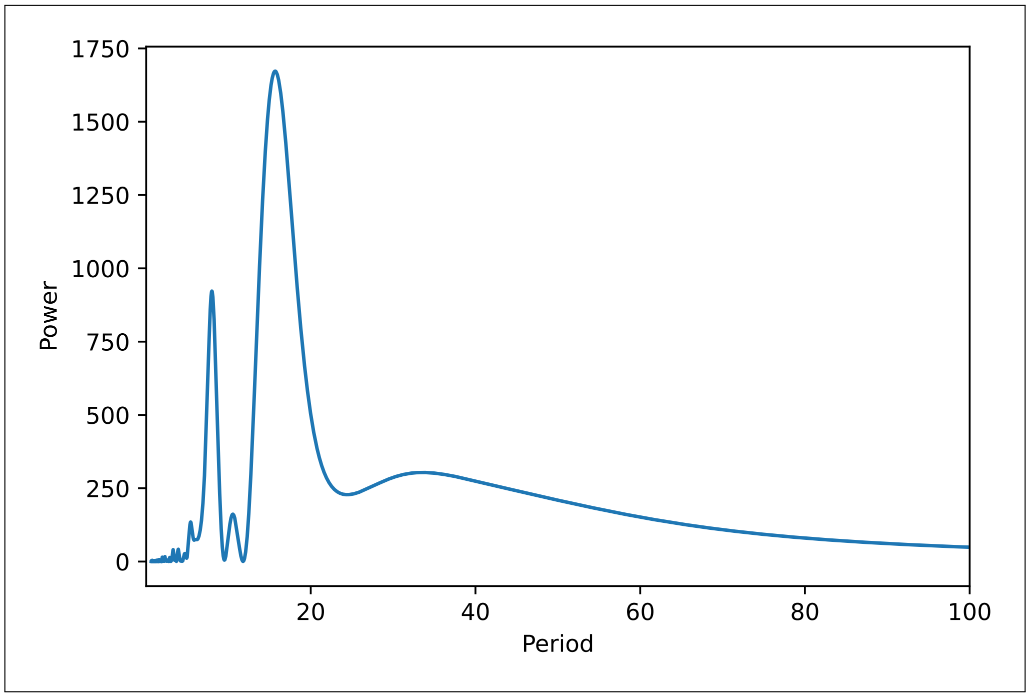


Figure 10

2.1.2 Fitting Gaussians to the Periodogram

I noticed that there were four primary peaks in the periodogram, at different angular frequencies. To establish which frequencies these were associated with, I fitted four Gaussians to the data using scipy.optimize.curve_fit in python.

The 4-Gaussian model is given by:

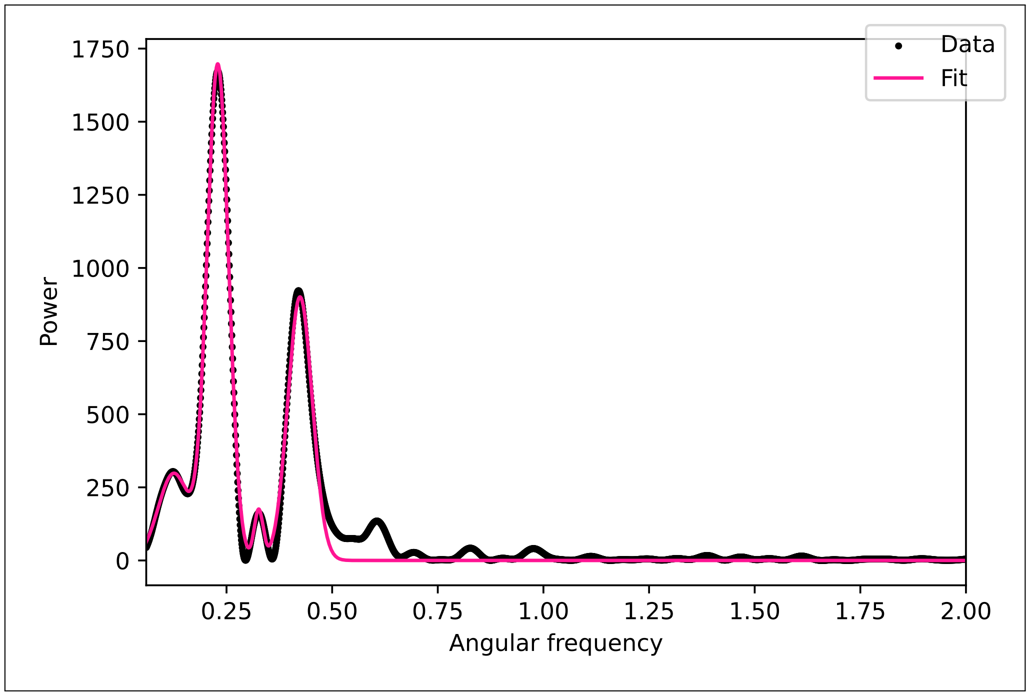
$$\text{power} = A_1 e^{-\frac{(p-p_{0_1})^2}{2\sigma_1^2}} + A_2 e^{-\frac{(p-p_{0_2})^2}{2\sigma_2^2}} + A_3 e^{-\frac{(p-p_{0_3})^2}{2\sigma_3^2}} + A_4 e^{-\frac{(p-p_{0_4})^2}{2\sigma_4^2}}, \tag{18}$$

where p is the period value, p_0 is the mean of the Gaussian, A is the amplitude and σ is the standard deviation.

By inspecting the periodogram, I made the following initial guesses for the Gaussian parameters, for power against angular frequency:

$$A_1 = 260 \tag{19}$$
$$t_{0_1} = 0.12 \tag{20}$$
$$\sigma_1 = 0.01 \tag{21}$$
$$A_2 = 1600 \tag{22}$$
$$t_{0_2} = 0.2 \tag{23}$$
$$\sigma_2 = 0.01 \tag{24}$$
$$A_3 = 100 \tag{25}$$
$$t_{0_3} = 0.35 \tag{26}$$
$$\sigma_3 = 0.005 \tag{27}$$
$$A_4 = 900 \tag{28}$$
$$t_{0_4} = 0.4 \tag{29}$$
$$\sigma_4 = 0.01. \tag{30}$$

The fitted Gaussians are shown in the following plot.



The results of the curve_fit approach are below.

First Gaussian parameters:

$$A = 297.90999236202117 \pm 3.271300874806592 \tag{31}$$
$$mean = 0.1254673539758214 \pm 0.0005453348857795788 \tag{32}$$
$$std = 0.03720313838632927 \pm 0.0006776891283775317. \tag{33}$$

Second Gaussian parameters:

$$A = 1691.9354021230115 \pm 3.7341305912351053 \tag{34}$$
$$mean = 0.2294096695160523 \pm 8.233631114494472e-05 \tag{35}$$
$$std = 0.025580126901335 \pm 8.222358484156754e-05. \tag{36}$$

Third Gaussian parameters:

$$A = 171.29302550704844 \pm 5.751137821734896 \tag{37}$$
$$mean = 0.3260593910140013 \pm 0.00041231487085792033 \tag{38}$$
$$std = -0.010588258593880722 \pm 0.000413845932962637. \tag{39}$$

Fourth Gaussian parameters:

$$A = 901.0970982370443 \pm 3.471372927439471 \tag{40}$$
$$mean = 0.4239780930875808 \pm 0.00012975486565853546 \tag{41}$$
$$std = 0.029192207340780145 \pm 0.00013089817575411708. \tag{42}$$

By inspection, I would conclude that there are 3 primary periodicities in the data, the periods of which are $\approx 27.806 \pm 2.844$ days, 14.911 ± 0.992 days, 16.931 ± 0.134 days (in order of signal height). The other fitted

Gaussian appeared rather insignificant, and is seen to have a small prominence in the fitted line (not much more than the small peak on the left side). Therefore, I neglect this as a significant signal and I would estimate there to only be 3 primary periodicities.

2.2 Part II: Gaussian Process

To implement a Gaussian process, I had to first choose which/how many kernels to use.

As I had identified 3 primary periodicities using the Lomb-Scargle periodogram, I decided to use three periodic kernels. I decided to not include a linear offset, as I had no reason to believe the brightness of the starts would experience significant offsets over the course of a few months.

Therefore, I used three kernels k_1, k_2 and k_3 using the following formulation:

$$k(x, x') = \exp\left[-\frac{2\sin^2\left(\frac{\pi(x-x')}{P}\right)}{l^2}\right] \tag{43}$$

Where l is the length scale of the kernel, and P is the period of the oscillation. Each kernel was initialised with the periods found through Gaussian fitting in the previous part ($P = 27.806, 14.911, 16.931$ days.

To perform the Gaussian processing, I also initialised the scaling factors for each kernel (dictating the strength of each kernel). I did this by inspecting the normalised plot of the Lomb-Scargle periodogram - where the height of the peaks correspond to the "strength" or "significance" of the periodicity.

The result of these decisions was the following kernel governing the Gaussian process:

$$K = 0.2 \times k_1 + 1 \times k_2 + 0.6 \times k_3. \tag{44}$$

The relevant parameters that were obtained through the Gaussian processing were the periods of the sub-kernels P_1, P_2, P_3 and their length scales l_1, l_2, l_3.

After simply using the scipy minimize function (L-BFGS-B method) to optimise the kernel parameters, the following plot was obtained for the predicted brightness 2 months into the future. The prediction plot is coloured between one standard deviation above and below the actual predictd line.

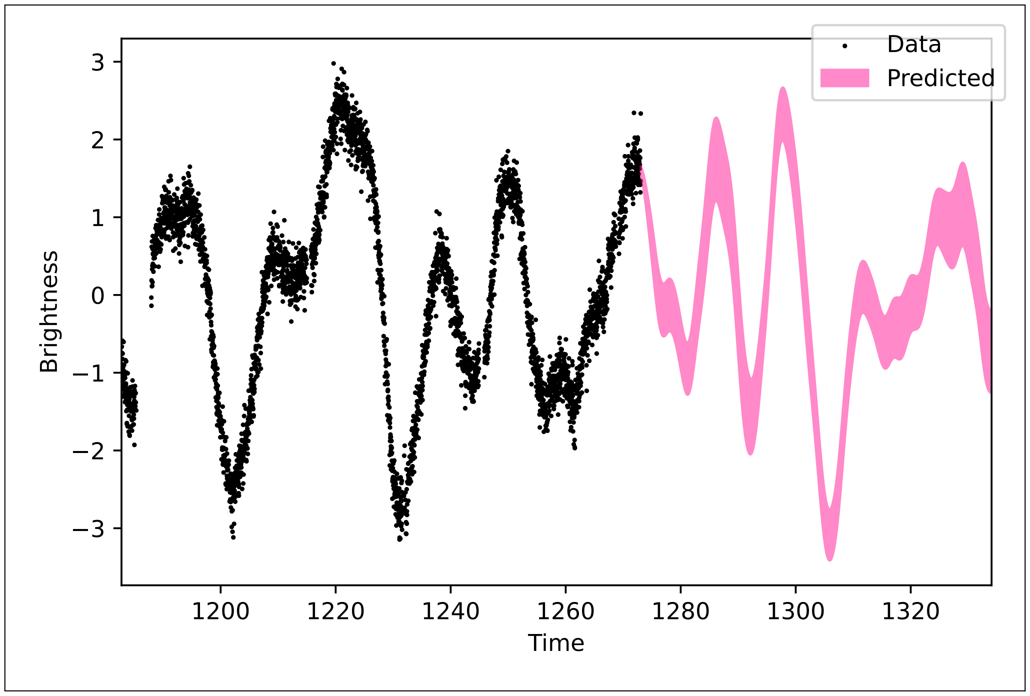


Figure 11

This plot appears to include a somewhat realistic-looking prediction. However, it still doesn't appear entirely consistent with the actual data, meaning we should ameliorate our parameters through the use of sampling. The sampling method will allow us to also plot the "spread" of the possible modelling parameters, which will give a better understanding of possible outcomes (and their likelihood) in the future.

2.2.1 Sampling

I made use of cmdstanpy sampling to find the idea parameters for the kernel. I used the output parameters from the minimization method as initialisations for the sampling.

Due to time constraints, I was only able to conduct 200 samples total (100 for the burn-in, and 100 for the actual sampling). The resulting chains were stored for each of the kernel parameters, and

This involved randomly choosing 50 points in the chain plots to extract parameters from, and apply to a prediction. Each prediction was plotted on the same axes as the data, as shown below.

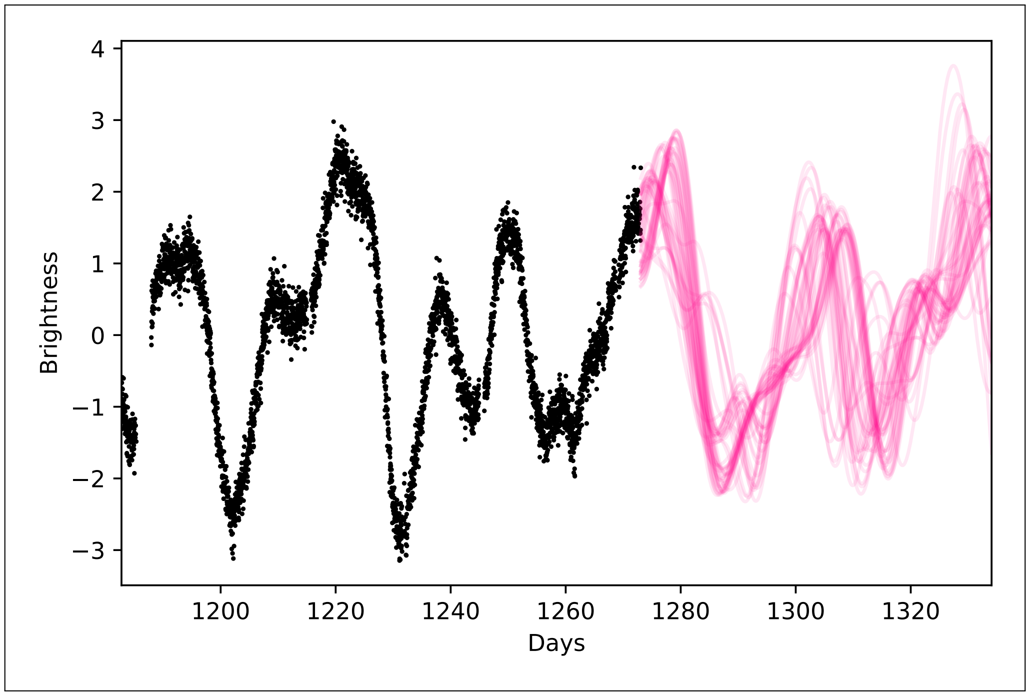


Figure 12

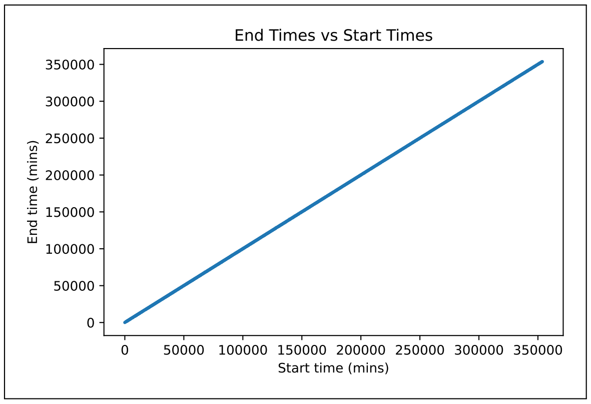
Although this prediction looks messier, it reflects the vague nature of our uncertainty. I note that if I were able to run more samples, the data may seem slightly neater - while I'm aware of this, the process was very time-expensive. Visually, it seems that the chains did provide values that make predictions appear consistent with the data. However, ideally one would conduct more samples (with a better computer than mine) to get a more accurate distribution of possible predictions.

3 Question 5

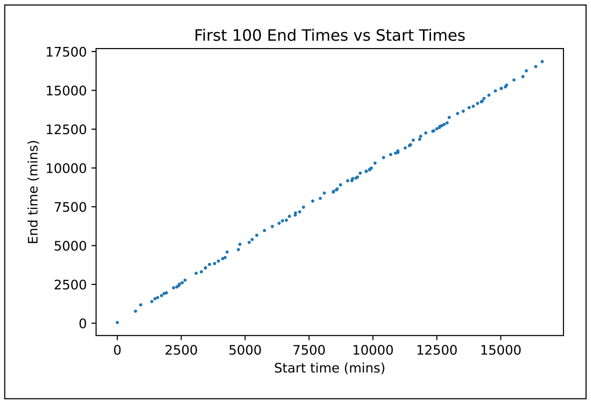
The first thing I noted about this data was that the data was ordered from latest time to earliest. Before even beginning the proper "cleaning" process, I sorted the data to be in chronological order, to make comparisons slightly easier when cleaning the data.

3.1 Before cleaning the data

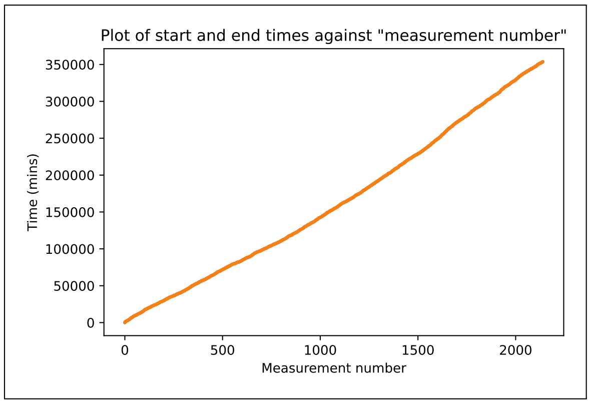
To get a sense of the data, I first plotted... everything against... everything. The following plots are my initial foray into the (now sorted) dataset, which only consisted of a row of "starting times" and "ending times" over the course of several months. Note that I plotted the "starting time" values in terms of time passed from the first measurement's starting time (i.e. starting from 0). I also plotting the "durations" as the difference between the end and start times, and the "delays" as the time between an end time and the subsequent startime time.


End Times vs Start Times
End time (mins)
Start time (mins)

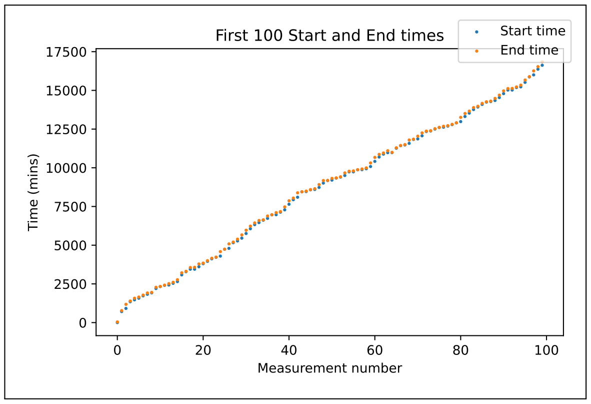
(a)


First 100 End Times vs Start Times
End time (mins)
Start time (mins)

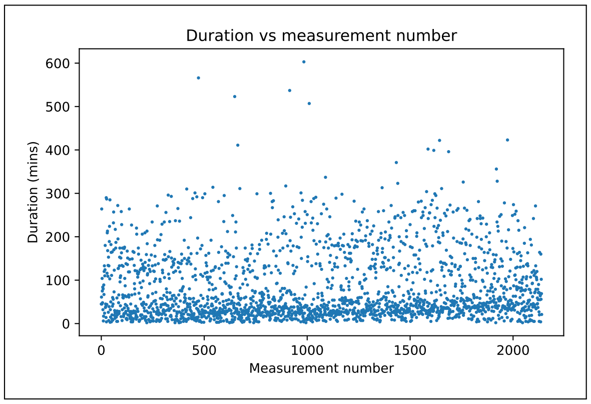
(b)


Plot of start and end times against "measurement number"
Time (mins)
Measurement number

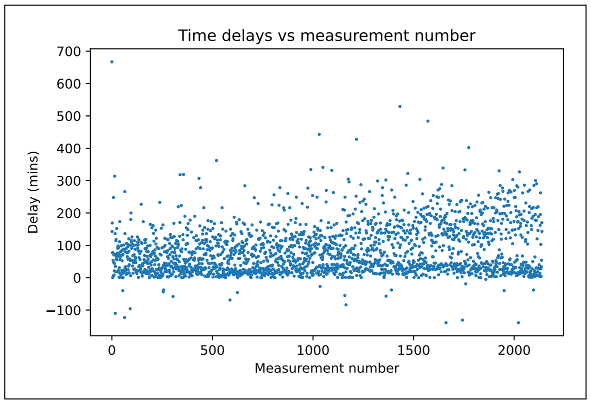
(c)


First 100 Start and End times
Start time
End time
Time (mins)
Measurement number

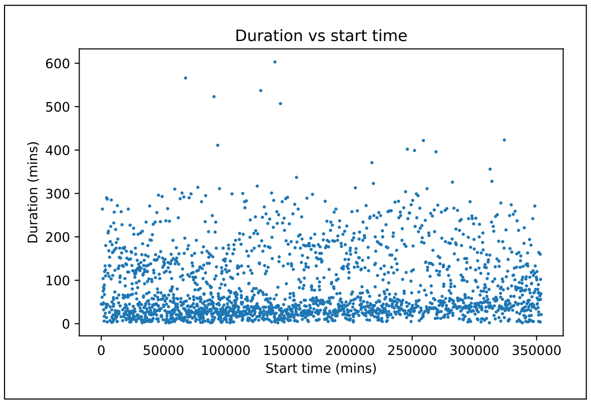
(d)


Duration vs measurement number
Duration (mins)
Measurement number

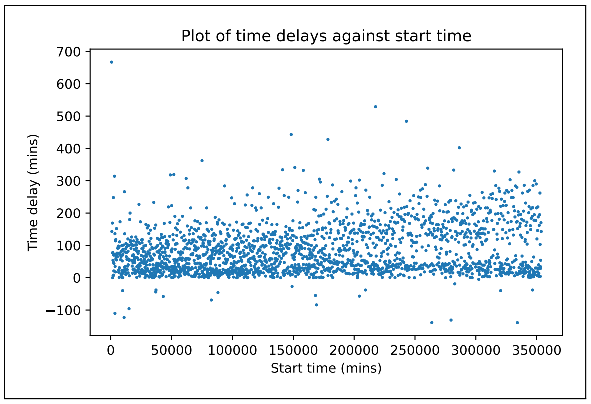
(e)


Time delays vs measurement number
Delay (mins)
Measurement number

(f)


Duration vs start time
Duration (mins)
Start time (mins)

(g)


Plot of time delays against start time
Time delay (mins)
Start time (mins)

(h)

Comments

- Durations and delays appear to have some constant value with spread to higher values;
- There appears to be a very strong linear relationship between end and start time;
- There is a less obvious relationship between start/end time and measurement number;
- Durations don't have a strong relationship with the start time nor experiment number, but there is a gathering of points at values around 40. There appears to be a slight increase in the value of this cluster, but it's not very significant.
- The time delays" - not completely similarly spread, like with duration. There seems to be an increase.
- to model this, I looked at the relationship between the start time and measurement number.

3.2 Cleaning the data

I first removed any duplicates from the data and any empty entries in the csv file.

I then looked to filtering out the outliers for the duration values. I did this by implementing a mixture model of two possible models: the relevant model, and the outlier model.

I conducted inference for the proportion of the duration data that were outliers, by using cmdstanpy to sample the parameters using the assumption of a constant value model. As seen in Figure 13e and the apparent linear relationship between start and end times, this is a sensible model to apply.

While no uncertainties were provided in the data, I made the assumption that the time measurements had some uncertainty associated with them. However, given this assumption, I marginalised over these uncertainties such that the the likelihood for this outlier model was:

$$\mathcal{L} = p(y, |x, \mu_{0_y}, \sigma, Q) \propto \prod_{i=1}^{N} \left(\frac{Q}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - \mu_{0_y})^2}{2\sigma^2}} + \frac{(1-Q)}{2\pi\sqrt{\sigma^2 + \sigma_o^2}} e^{-\left(\frac{(y_i - \mu_{0_y})^2}{2(\sigma^2 + \sigma_o^2)}\right)} \right), \tag{45}$$

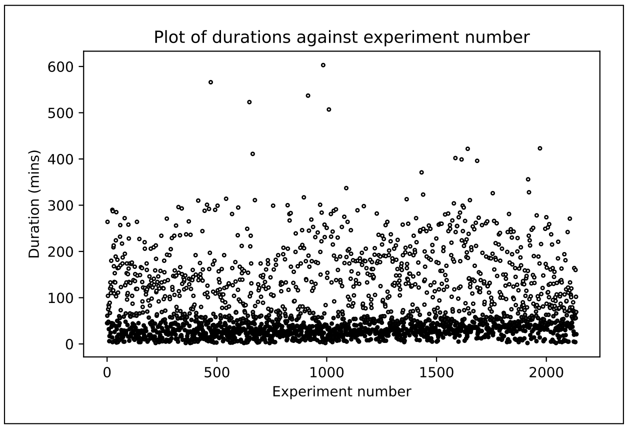
Where x and y are measurement number and durations respectively, σ is the marginalised uncertainty for the data, and σ_o is the added uncertainty associated with the outlier model (which is "blurrier", or more widely spread).

The stan model for this mixture model was accordingly:

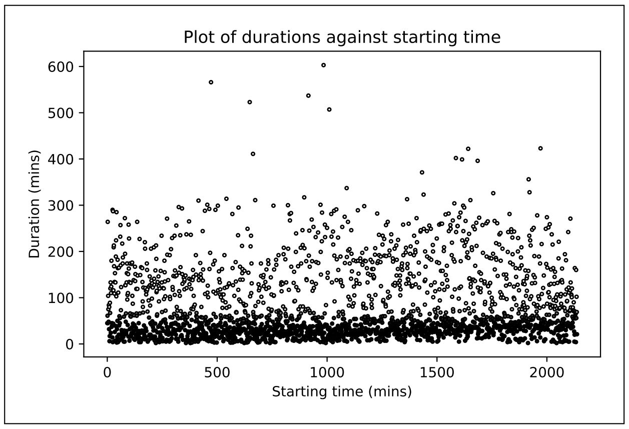
```
data {

int<lower=0> l;

vector[l] y;

}

parameters {

real<lower=0> sigma_real;

real<lower=0> sigma_out;

real<lower=0,upper=1> q;
real mu_real;
real mu_out;
}
model {
for (i in 1:l){
real log_prob1 = normal_lpdf(y[i] — mu_real, sigma_real);
real log_prob2 = normal_lpdf(y[i] — mu_out, sigma_real+sigma_out);
target+= log_mix(q,log_prob1, log_prob2);
}
}
generated quantities {
vector[l] lp1;
vector[l] lp2;
for (i in 1:l){
lp1[i] = normal_lpdf(y[i] — mu_real, sigma_real);
lp2[i] = normal_lpdf(y[i] — mu_out, sigma_real+sigma_out);
}
}
```

It was not possible to really assess the outliers for the delay times, as during the cleaning process, entries in the dataframe were removed. This altered the effective recorded time difference between readings. Therefore, I didn't engage in any "cleaning" based on the distribution of the time delays.

The following plot identifies (by transparency) which duration values are outliers.



(a)



(b)

To be able to remove appropriate points, I chose a threshold for ignored points. Points with a likelihood of belonging to the main model less than 10^{-8} were ignored. The result of this threshold is plotted in the following figure, illustrating the new clean data.

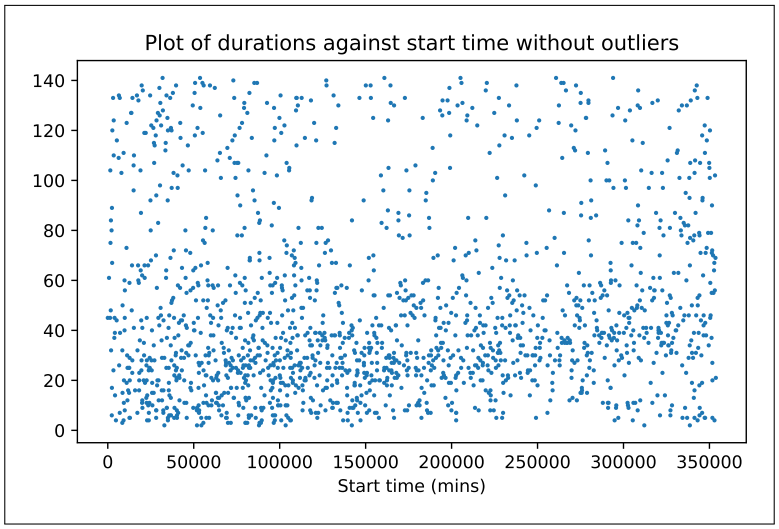


Figure 15

While I did not conduct inference on outliers for the delay times between measurements, Figure 13h shows that some delay times in the raw data are negative. I made the conclusion by inspection that this data must not be ”clean” data, as there were very few instances of this occurring, and it was unlikely that such separate durations could possibly occur concurrently. If the data involved the measurement of durations that could be concurrent, more negative values would be anticipated.

Therefore, I removed the data that included start times occuring before the previous measurement (in chronological order) was finished. The result of this, compared with Figure 13h, is plotted below. It can be seen that the sparse negative values have been removed.

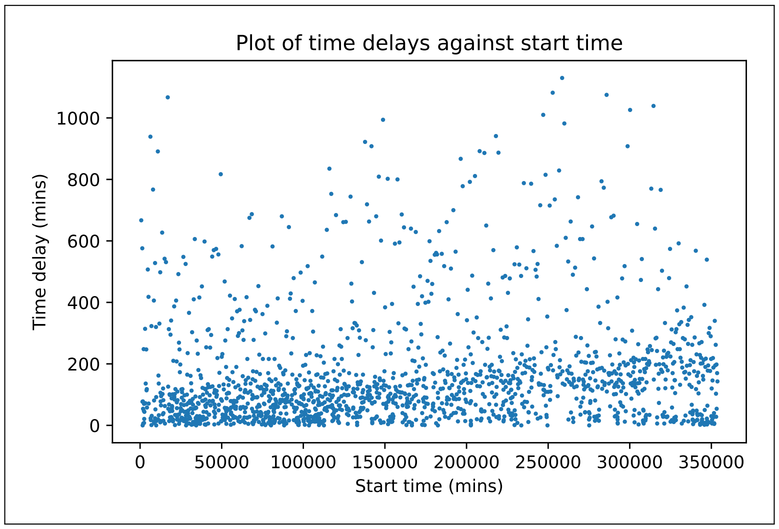
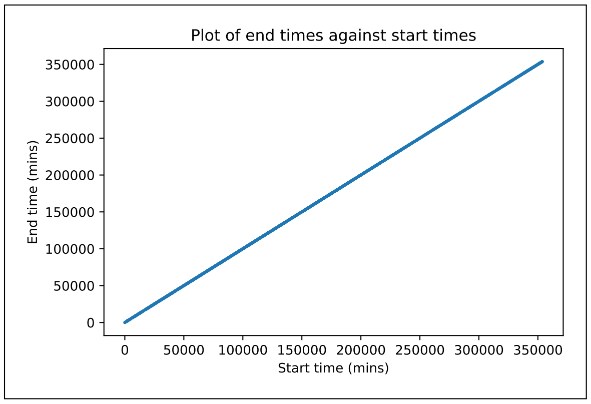


Figure 16

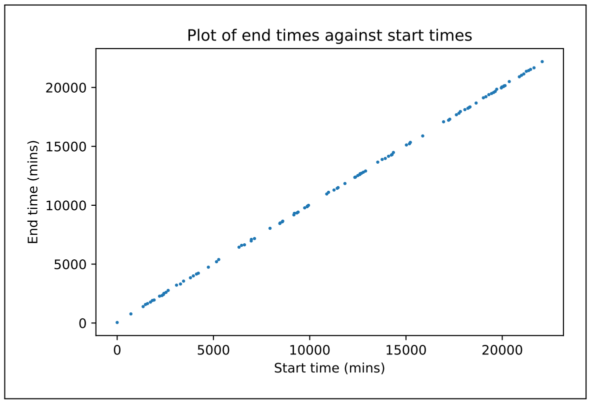
At this point, I declared the data was ”clean”.

3.3 Modelling the clean data

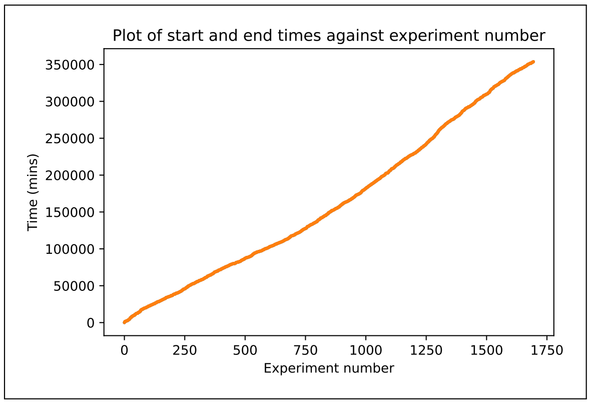
Now using the clean data, the relevant plots that reveal patterns in the data are found below.



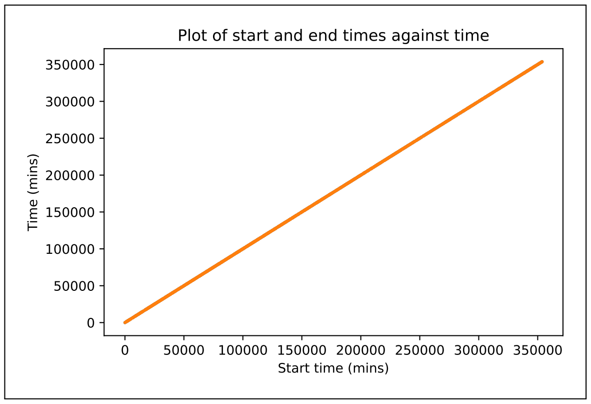
(a)



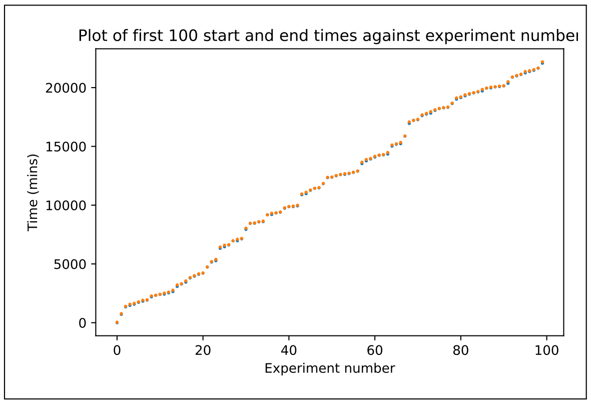
(b)



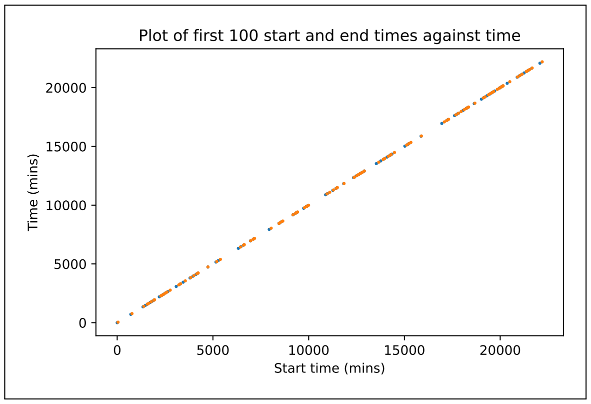
(c)



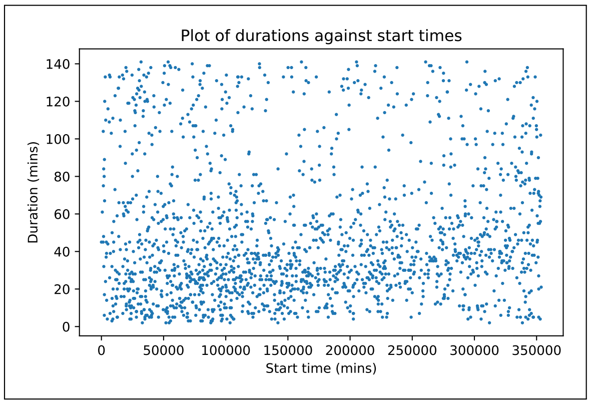
(d)



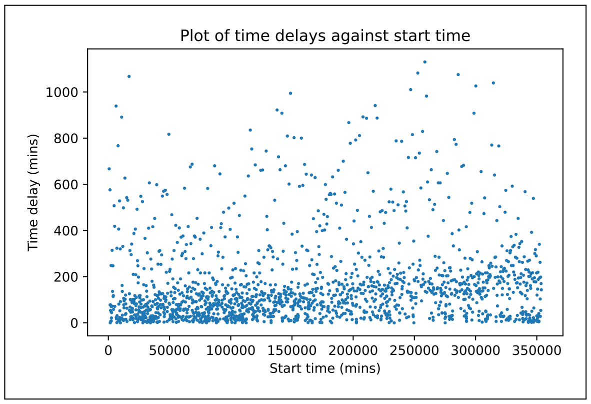
(e)



(f)



(g)



(h)

Figure 17

At this point, I will set out to model three relationships that I can see in the data. Firstly, the relationship between end time and start time, such that an end time can be predicted for a given start time (effectively modelling the duration).

3.3.1 Modelling end time vs start time

As mentioned before, one of the stronger patterns seen in the data is the relationship between start times and end times, seen in 17a. The linearity of this relationship is preserved even when zooming in, as shown in 17b.

If this relationship is linear, that would mean that "duration" is effectively a constant. This somewhat agrees with the plot of durations against start times, shown in Figure 15, which despite having a large variance in the data (ranging from 0 to 140 minutes), appears to be slightly more concentrated around a single value (around 20 mins). This concentration dilutes slightly at higher starting times, but there isn't a particularly distinct increase.

Therefore, it seemed sensible to use a **linear** generative model for the relationship between end and start time.

I modelled the data in a linear fashion, such that:

$$y_i \; \mathcal{N}(mx_i + c, \sigma_i) \tag{46}$$

Where y_i and x_i are the end and start times respectively, m is the gradient and c is the offset, and σ_i represents the uncertainty of the model. I marginalised over this uncertainty, so it can be said that the model is:

$$y_i \; \mathcal{N}(mx_i + c, \sigma), \tag{47}$$

Where σ is the convolved uncertainty of the Gaussian as a result of marginalisation over the system's own uncertainty.

The data itself did not contain uncertainties, but I included them as parameters to be sampled in the inference.

Using conditional probability:

$$p(y_i, \sigma_{y_i}, m, c|x_i) = p(y_i|x_i, \sigma_{y_i}, m, c)p(\sigma_{y_i})p(m, c) \tag{48}$$

$$\tag{49}$$

where

$$p(y_i|x_i, \sigma_{y_i}, m, c) = \frac{1}{2\pi\sigma_{y_i}^2} e^{-\frac{(y_i-(mx_i+c))^2}{2\sigma_{y_i}^2}}. \tag{50}$$

I did not immediately care about finding the uncertainty σ_{y_i} for each data point. Therefore, I marginalised over this parameter using a half-normal prior:

$$f \sim \mathcal{N}(\mu_\sigma, \sigma_\sigma) \tag{51}$$

$$\sigma_{y_i} = |f|. \tag{52}$$

Following from the fact that convolution of Gaussians results in a Gaussian, our marginalised probability then becomes:

$$p(y_i, \sigma_y, m, c|x_i) = \frac{1}{2\pi\sigma_T^2} e^{-\frac{(y_i-(mx_i+c))^2}{2\sigma_T^2}}. \tag{53}$$

Where $\sigma_T^2 = \sigma_{y_i}^2 + \sigma_\sigma^2$.

Further, to ensure the prior distribution of m is equal over all the possible gradients, the prior probability for m and c is given by:

$$p(m, c) = (1 + m^2)^{-3/2}. \tag{54}$$

The total probability of the system is therefore (where N is the number of points):

$$p(y_i, \sigma_y, m, c|x_i) = \prod_{i=1}^{N} (p(y_i|x_i, \sigma_y, m, c)p(m, c)) \tag{55}$$

$$p(y_i, \sigma_y, m, c|x_i) = \prod_{i=1}^{N} \left(\frac{1}{2\pi\sigma_T^2} e^{-\frac{(y_i-(mx_i+c))^2}{2\sigma_T^2}} (1 + m^2)^{-3/2} \right) \tag{56}$$

$$\tag{57}$$

The log probability of the system (where x is the experiment number) is therefore:

$$\ln \mathcal{L} = -\frac{3}{2} N \ln\bigl(1 + m^2\bigr) - N \ln 2\pi\sigma_T^2 - \sum_{i=1}^{N} \frac{\bigl(y_i - (mx_i + c)\bigr)^2}{2\sigma_T^2}. \tag{58}$$

Coding steps I first used the least squares method in linear algebra to approximate m and c. I then used curve fit to get a refined initialisation for m and c.

The fitted parameter values were:

$$c = 44.09554 \tag{59}$$

$$m = 1.000025. \tag{60}$$

Finally, I used these initialisations to sample the parameters, using cmdstanpy. As the line seems very strongly linear, I could have chosen to not use stan model inference, and instead leave it at curve_fit. However, as I was going to make predictions, it was worth considering the standard deviations in the sampling, and to allow these to propagate into the future prediction.

I used the following stan model to sample the parameters of this system.

data {

int<lower=0> N;

vector[N] x;

vector[N] y;

}

parameters {

real m;

real c;

real<lower=0> sigma;

}

model {

c   uniform(-100,100);

sigma   normal(0,1000);

y   normal(c + x * m, sigma);

}

The resulting chains from this sampling are plotted below, for all three parameters.

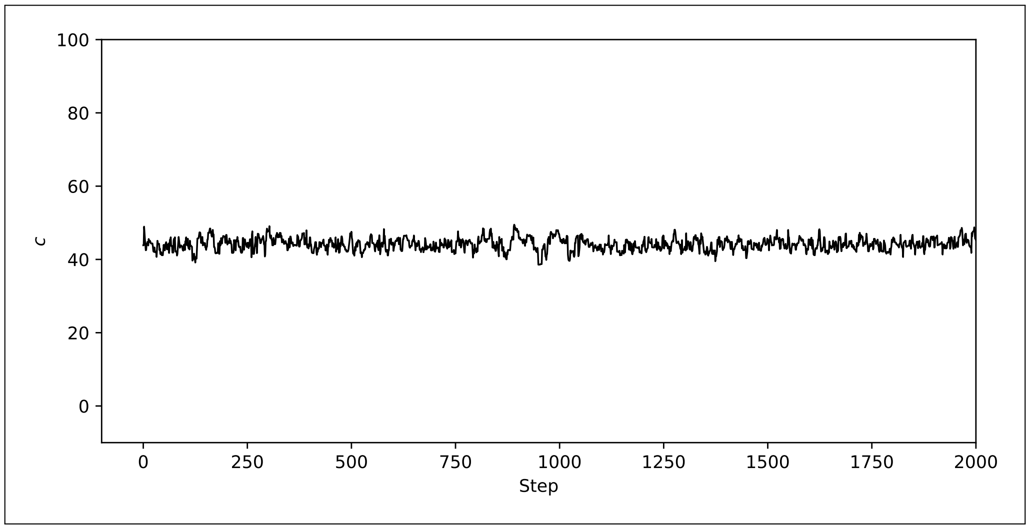


Figure 18

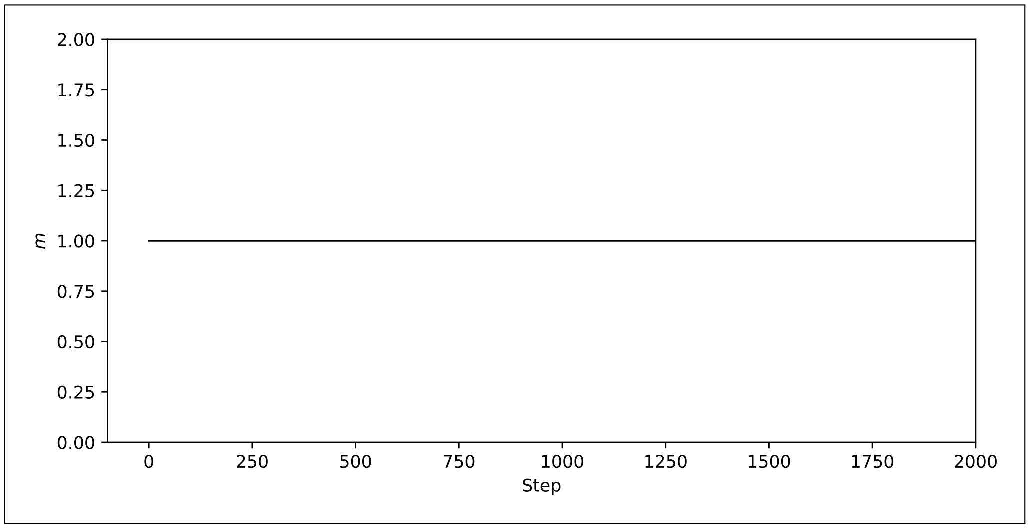


Figure 19

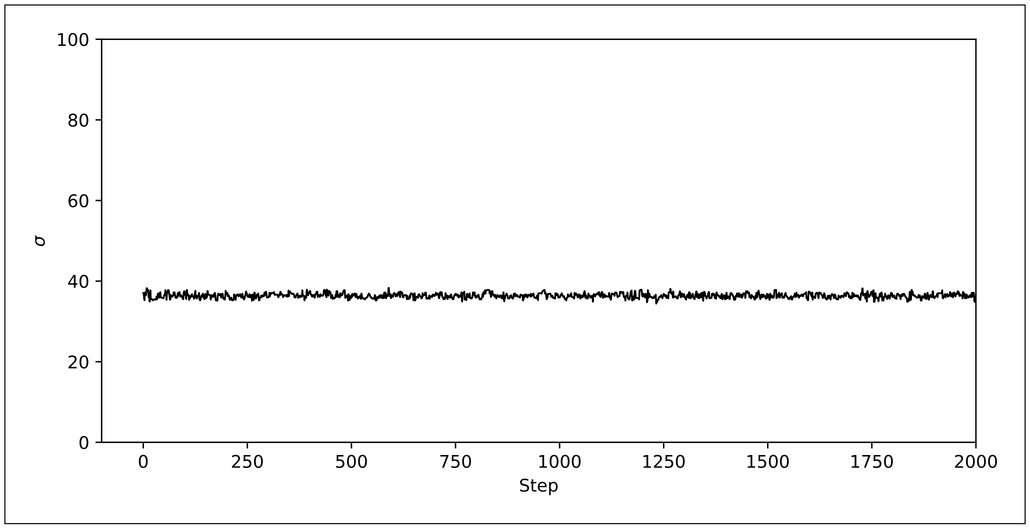


Figure 20

The above chains not only appear to have converged, but their $\hat{R}$ values are each close to 1. The resultant parameters from sampling are:

$$c = 44.2601 \text{ with a standard deviation: } 1.7776 \tag{61}$$

$$m = 1.000025 \text{ with a standard deviation: } 9.6987 \times 10^{-6} \tag{62}$$

$$\sigma_T = 36.3364193 \text{ with a standard deviation: } 0.61837. \tag{63}$$

We will return to this model later, when making general predictions.

3.3.2 Modelling the start time (AKA time-delays)

I thought it would be useful to be able to predict the start time on its own, in addition to the end times. For example, if this data were recorded for something turning on an off, it would be useful to be able to model the delay between these ”ons” and ”offs” (or simply, when the ”ons” occur).

For this purpose, I plotted starting times alone against the experiment number, such that it can be predicted that the n-th starting time would be a certain value.

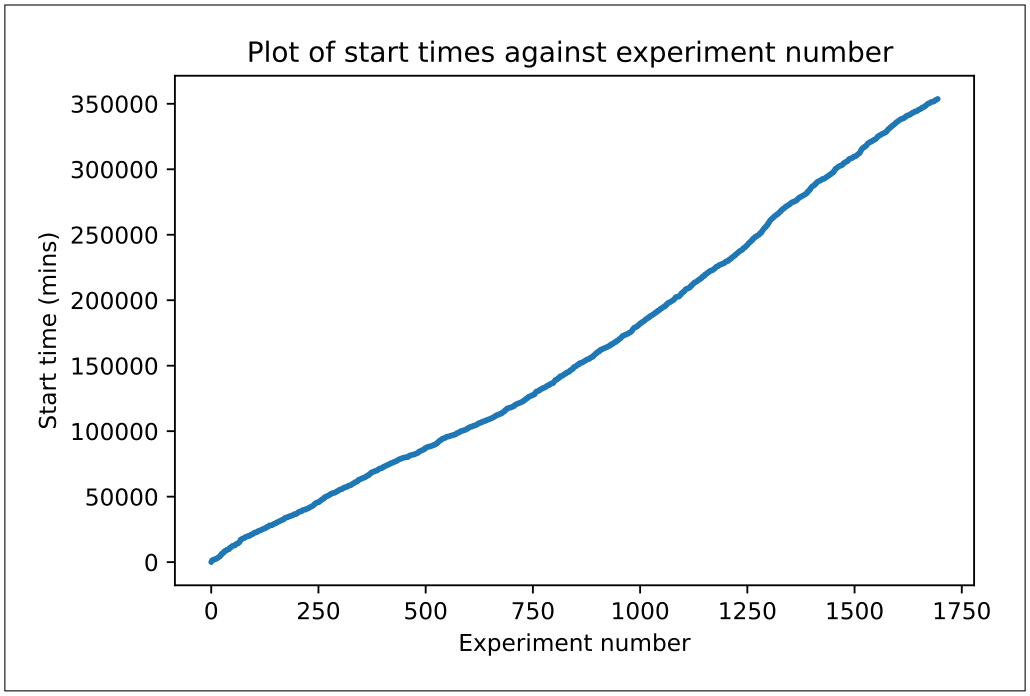


Figure 21

It is not immediately obvious what model would fit this data. In the interests of applying the simplest model possible as a starting point, I decided to only explore polynomial trends.

As a first point of call, I applied the Bayesian Information Criterion to assess which polynomial model would be ideal.

The formula for BIC is given by:

$$BIC = D \ln N - 2 \ln \hat{\mathcal{L}}, \tag{64}$$

Where D is the number of parameters in the model, theta are the parameters and $\mathcal{L}$ is the likelihood.

To engage in the linear algebra approach to point estimating the parameters, I assumed that there was some uncertainty associated with the starting times (again, despite the lack of given uncertainties in the data). I assumed that both start and end times would share similar uncertainties. To be safe, I still only imposed their uncertainties to be a fifth of the uncertainties previously established in the previous sampling method (σ).

Applying the likelihoods for the different polynomial models, the BIC for each number of parameters is plotted below.

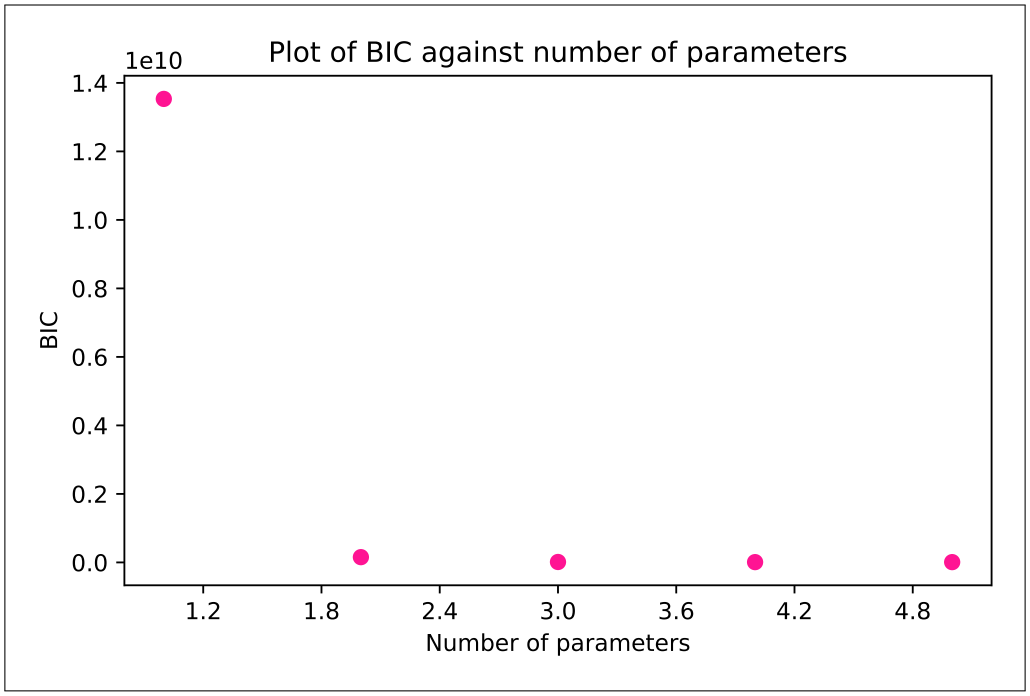


Figure 22

The number of parameters with the lowest BIC (therefore the optimal model) was 5 parameters (a quartic polynomial), with the BIC value $\sim 8.5 \times 10^6$.

While this does provide some guidance, we have found this BIC value using an arbitrarily chosen uncertainty in the data. Further, we do not have a grasp of what model may be more appropriate in this case (one with more or less parameters). In the interests of simplicity (”always start with the simplest model!”) I first conducted inference using a linear model, and confirmed this was likely not correct.

The BIC values for a quadratic, cubic and quartic model were $1.25587245e{+}07, 8.49188002e{+}06, 8.48986786e{+}06$ respectively. I first noticed that the cubic and quartic BIC were nearly the same (differing by $\sim 0.02\%$), meaning I was more inclined to determine the cubic model than the quartic (simper is better). Further, the quadratic BIC was sufficiently close for me to consider it, and determine its suitability as a model accordingly.

LINEAR MODEL

Using a similar model to the previous inference, just for completeness, I used a linear model on the data for start times vs measurement number.

Using the curve_fit function, the initialisation values for the parameters m (gradient) and c (offset) were:

$$c = -13908.21 \text{ with a standard deviation: } 531.853 \tag{65}$$
$$m = 208.63 \text{ with a standard deviation: } 0.5437188688468032. \tag{66}$$

The stan model for the linear modelling of start time against measurement number is the following.

data {

int<lower=0> N;

vector[N] x;

```
vector[N] y;
}
parameters {
real m;
real c;
real¡lower=0¿ sigma;
}
model {
c   uniform(-100000000,100);
sigma   normal(0,1000);
y   normal(c + x * m, sigma);
}
```

The chain plots for this linear model are plotted below.

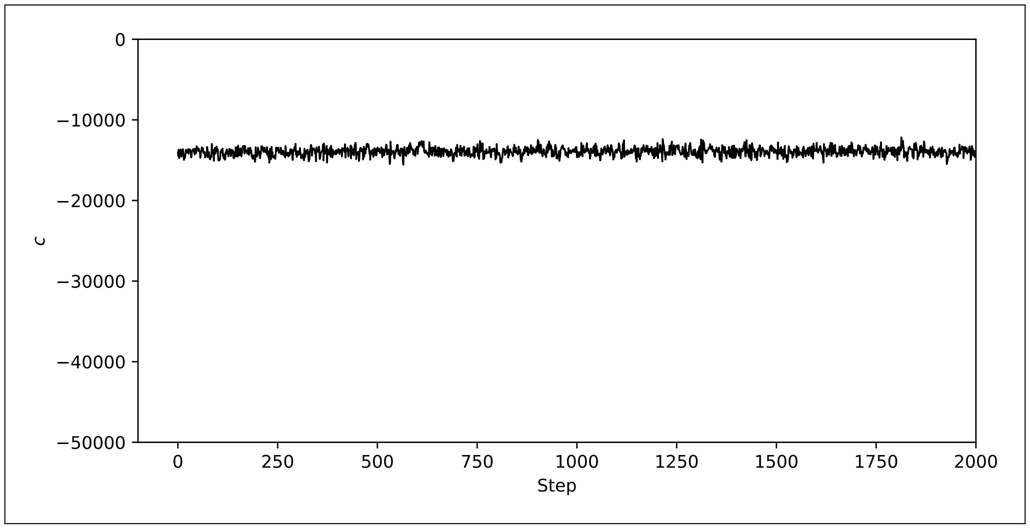


Figure 23

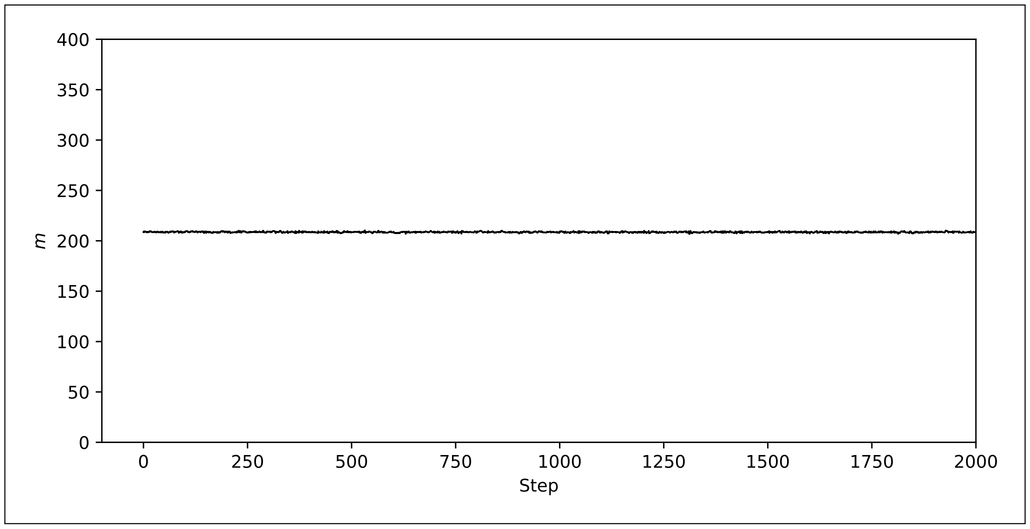


Figure 24

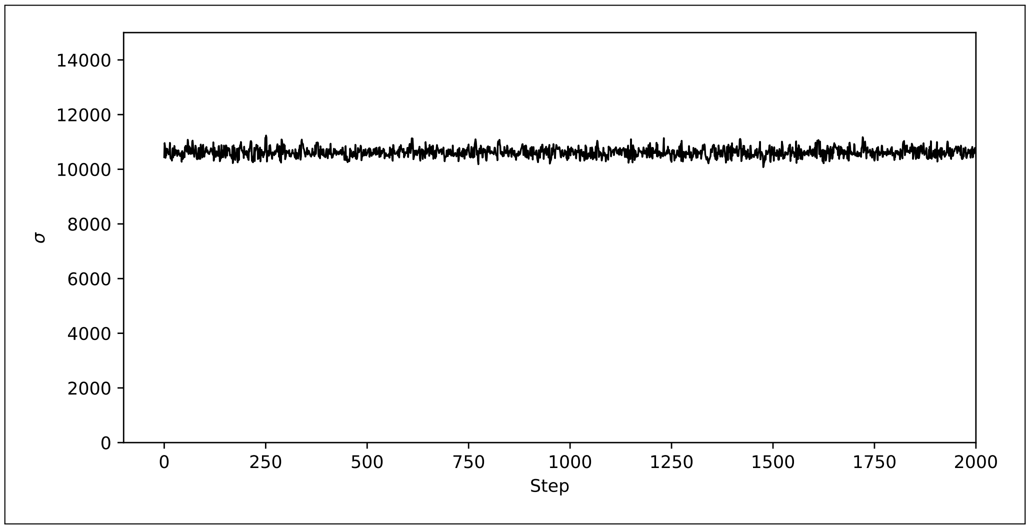


Figure 25

These chains all appear to converge, supported by their $\hat{R}$ values close to 1.

The resultant mean values for the parameters and their standard deviations, derived from the chain plots, were:

$$c = -13921.1376 \pm 507.8847 \tag{67}$$
$$m = 208.6395 \pm 0.5173 \tag{68}$$
$$\sigma_T = 10616.492 \pm 170.255. \tag{69}$$

The extended prediction for the starting times using this model is plotted below, with the actual data. The plot is filled within the first two standard deviations σ_T.

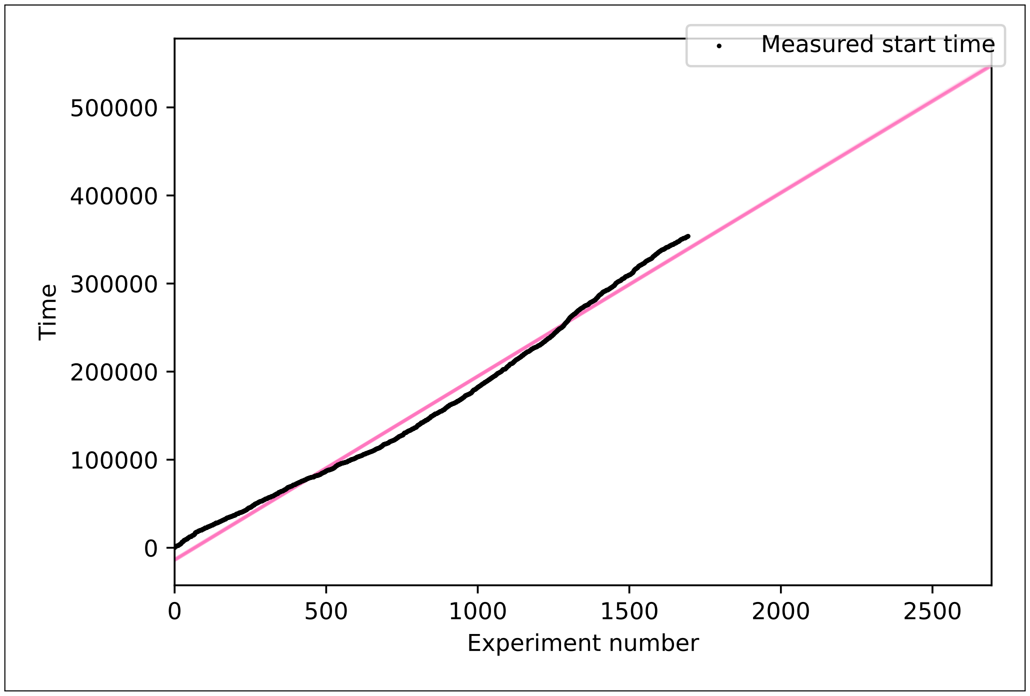


Figure 26

Visually, we can inspect the graph and conclude that our BIC value was rightfully higher for this model, as it fails to incorporate the deviations from a straight line that are clearly not due to noise. Rather, the relationship between start time and measurement number is clearly not linear. This is particularly notable when considering the sheer number of points that have followed a similar curve - if this were simply an artefact of the normal distribution of the data probability, one would expect the spread of data to be more random, rather than following a curved line. Therefore, it is not accurate to chalk those trends up to a large standard deviation, and the actual model is more likely to be a higher order polynomial.

QUADRATIC MODEL

Similar to the linear model, the quadratic likelihood is simply:

$$p(y_i, \sigma_y, m, c|x_i) = \prod_{i=1}^{N} \left(\frac{1}{2\pi\sigma_T^2} e^{-\frac{\left(y_i-(ax_i^2+bx_i+c)\right)^2}{2\sigma_T^2}} (1+m^2)^{-3/2} \right). \tag{70}$$

I again used curve_fit to initialise the values, and implemented the following stan model to conduct sampling for the parameters.

The stan model for the quadratic model is as below.

data {

int<lower=0> N;

vector[N] x;

vector[N] y;

}

parameters {

real a; real b; real c; real<lower=0> sigma; }

model {

sigma   normal(0,1000);

y   normal(a * x .* x + b * x + c, sigma);

}

The chain plots for parameters a, b, c and σ are given below. There all appear to converge, and have $\hat{R}$ values around 1.00.

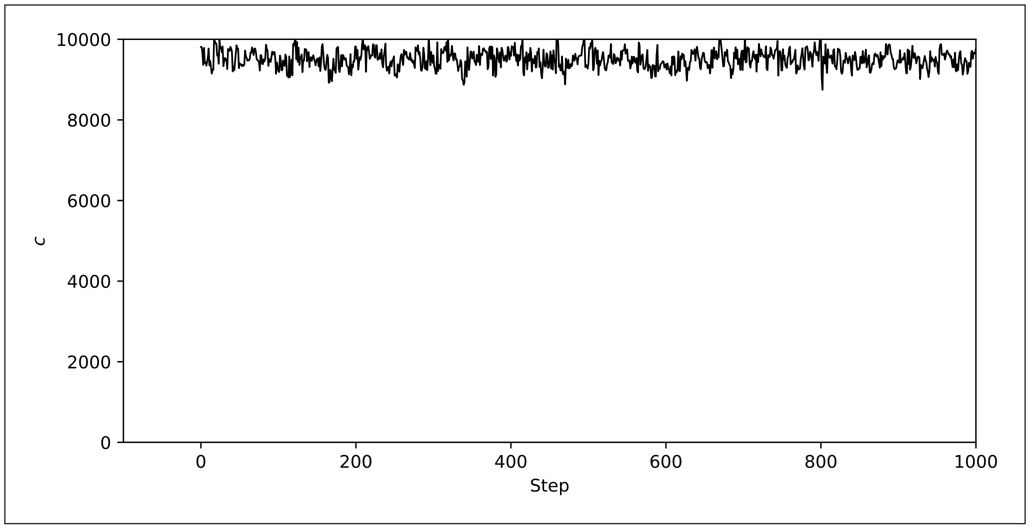


Figure 27

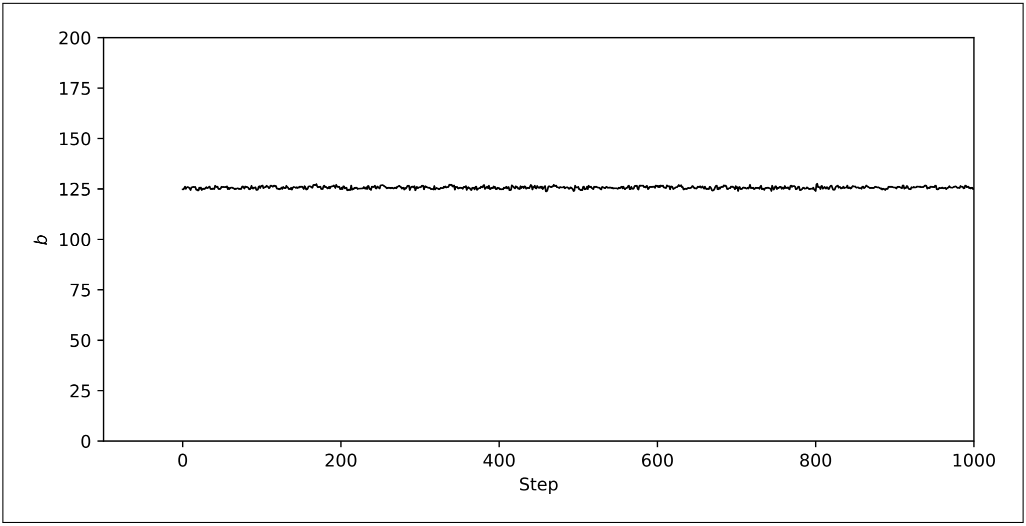


Figure 28

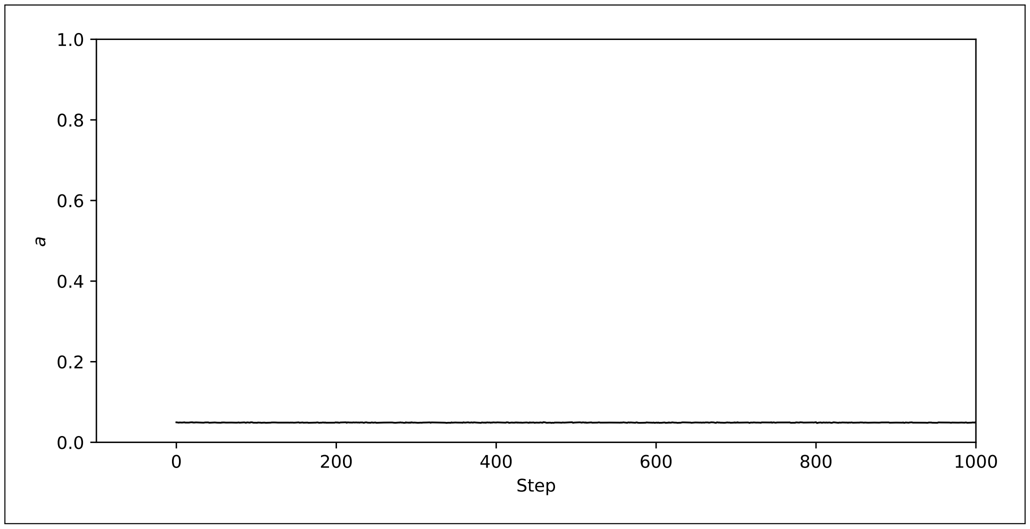


Figure 29

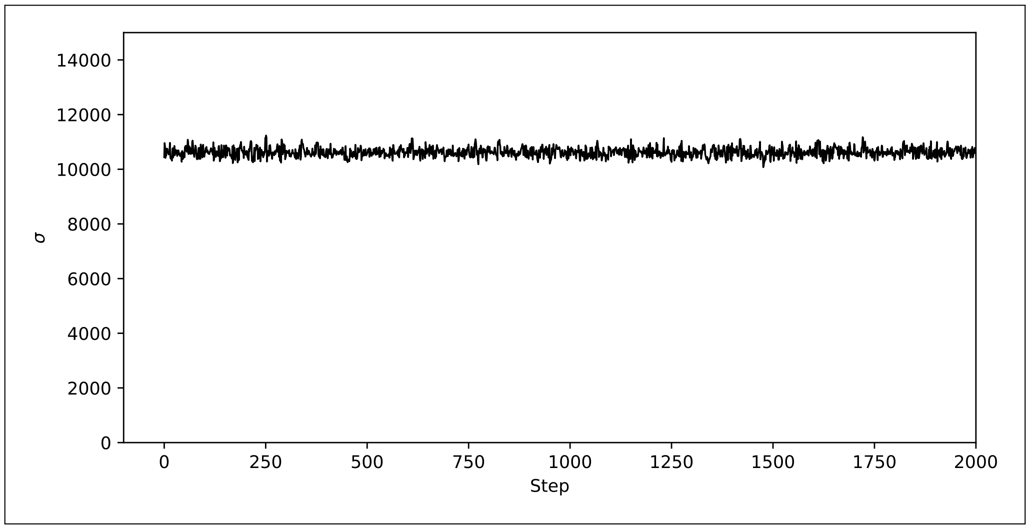


Figure 30

The final parameter values (their mean and standard deviations) were:

$$a = 0.0489911638 +/- 0.00035053 \quad (71)$$
$$b = 125.6456955 +/- 0.610032796 \quad (72)$$
$$c = 9503.207445 +/- 221.7859999 \quad (73)$$
$$\sigma_T = 3122.110465 +/- 54.0340. \quad (74)$$

(I note that while σ_T appears large, this is still on the order of the "thickness" of the depicted data line as the axes scale is very large. I have no reason for why I chose to do this in minutes, I made that decision at the beginning and never got around to changing it).

I plotted the quadratic model using the parameters' mean values, filled within **two** standard deviations, with the original data.

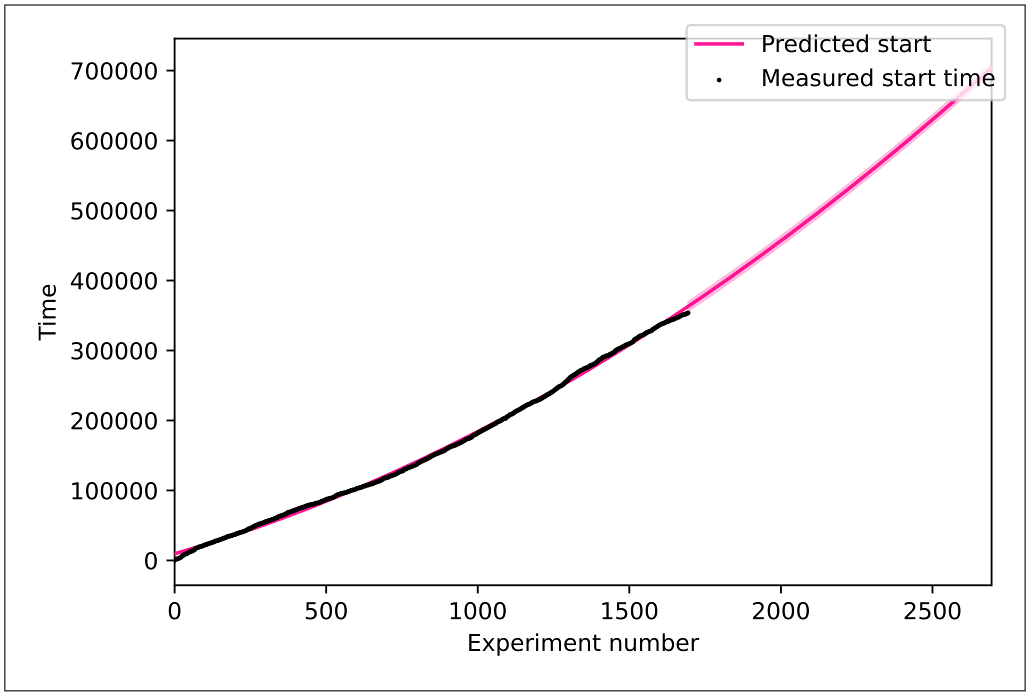


Figure 31

This fit appears much better than the linear fit. Further, the standard deviations appear to incorporate the full ”thickness” of the line, corresponding to the smaller-scale variance. However, there are two places the data seems to coherently deviate from the quadratic trend - at the beginning, and at the end of the data. Therefore, there is cause to explore a cubic trend, although a quadratic model appears to be in good standing to be the generative model for this data.

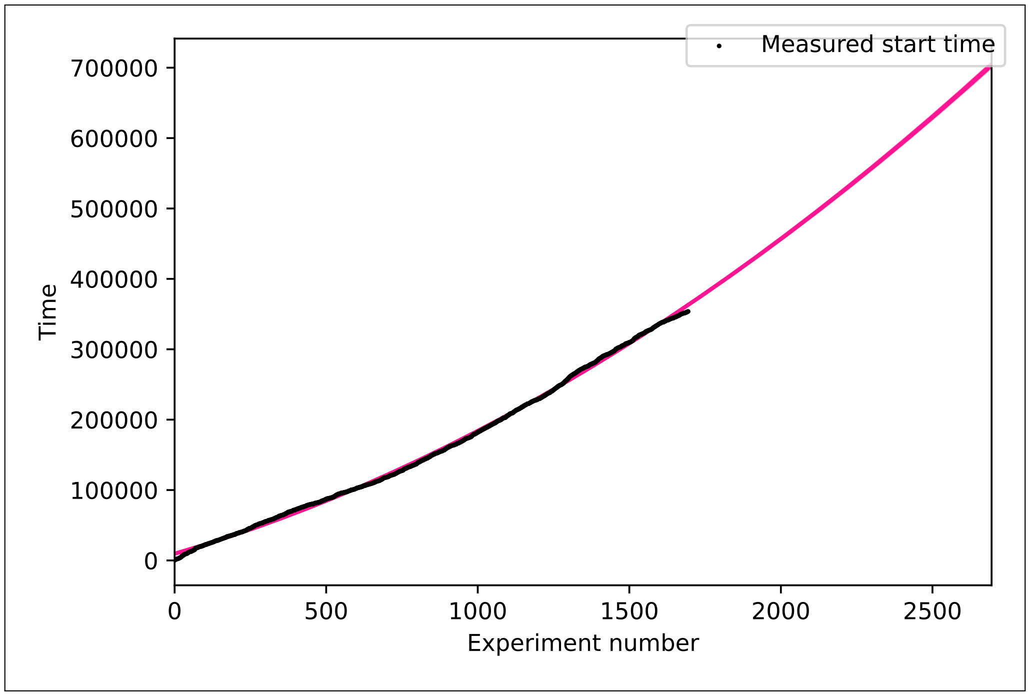


Figure 32

For this reason, I only used curve_fit to explore the cubic model.

CUBIC MODEL

I used the following cubic model, and applied it to the data using scipy.curve_fit to find the parameters.

$$y = ax^3 + bx^2 + cx + d. \tag{75}$$

The resultant parameters were:

$$d = 8595.61758733002 + / - 301.5555543832021 c = 132.1208993314244 + / - 1.5421006908315316 b = 0.03943043034902863 + \tag{76}$$

The result is plotted in pink below.

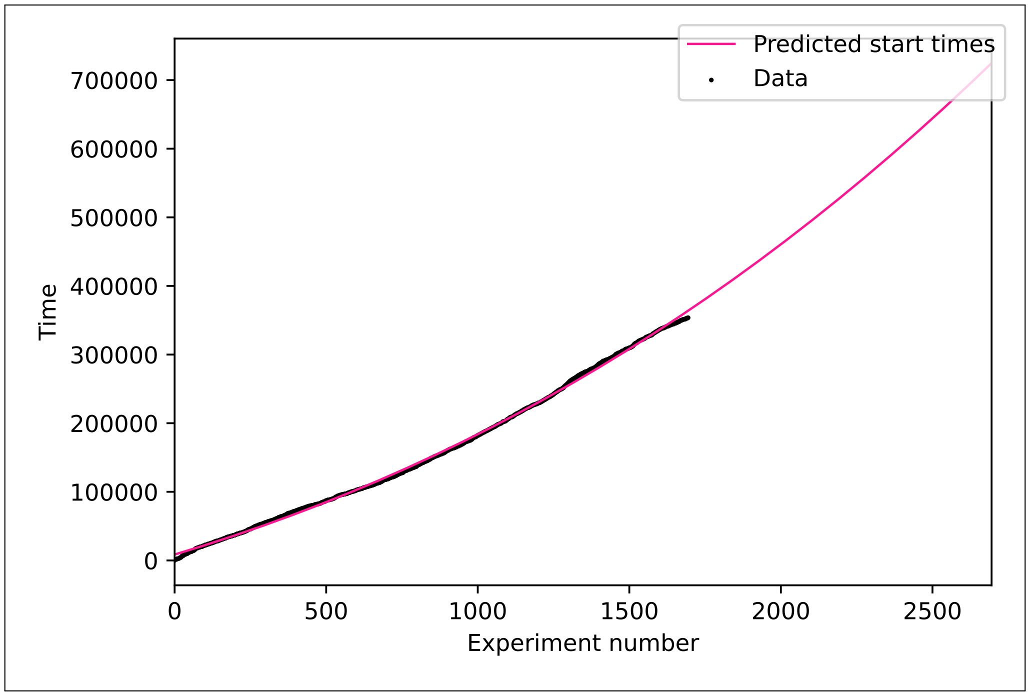


Figure 33

When compared with Figure 31 and 32, the cubic model does not seem to incorporate any major differences to the quadratic model - in fact, the final value for ”a” is relatively small, seeming to prefer a quasi-quadratic fit.

Without further information about the source or expected trend of the data, this model does not seem to provide a big advantage over the quadratic model. Therefore, I would choose the model the data with the quadratic model, in the interest of sticking to the simplest model.

3.4 Final predictions

Through the above methods, we have developed a quadratic model for the starting time as a function of ”measurement number”, and a linear model for the ending time as a function of starting time. Therefore, both of these can be predicted for future measurements.

It is easier to explain these predictions in the context of something turning ”on” at the start time, and ”off” at the end time. Using the quadratic model, we are able to predict when it will turn ”on” next (ie the start time of the following ”measurement number”. The general trend is that it will increase quadratically.

A visualisation of this prediction is given below, using the method of plotting 100 lines with parameters taken from random points in the chain plots. This was done to provide a representation of the spread of possible predictions, which was possible due to the sampling method.

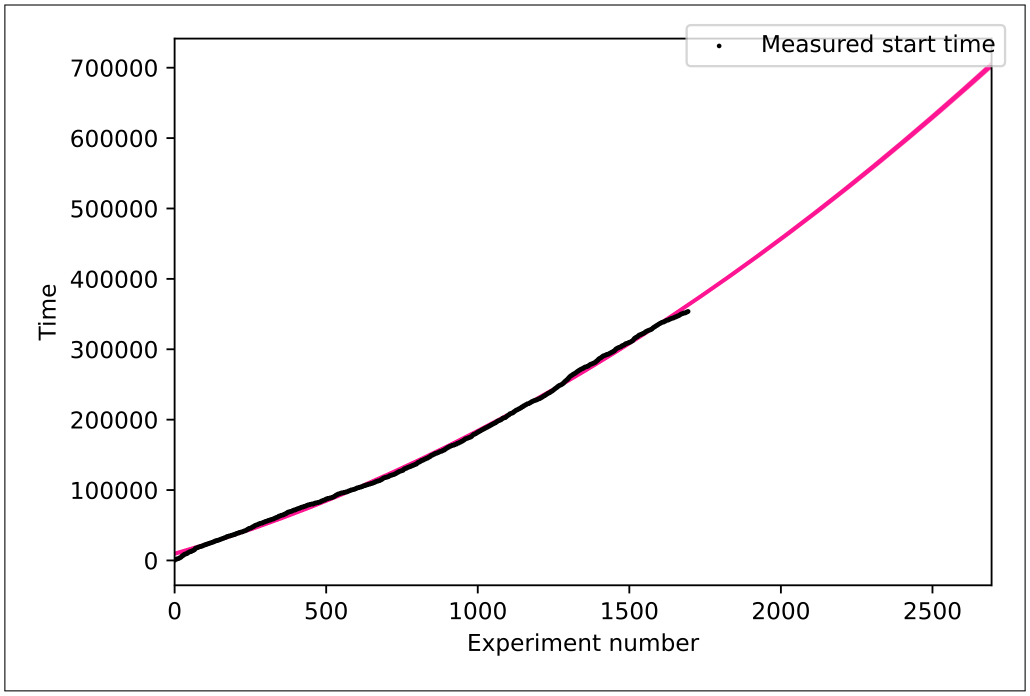


Figure 34

I do strongly note that there seems to be a deviation from this predicted increase near the end of the data. If this is due to an upcoming down-turn in the true data values, a different model may very well be more accurate. However, as we do not have insight into this ”future” data, the quadratic model suffices.

Finally, from this, we can also predict when the thing will turn ”off”, as this value is linearly dependent on when it turns ”on”. This prediction for end times is plotted below, again as a function of measurement number, and again using the method of picking 100 sets of parameters from random points in the sampled chain plots. This is effectively predicted to be some constant duration after whatever the starting time was.

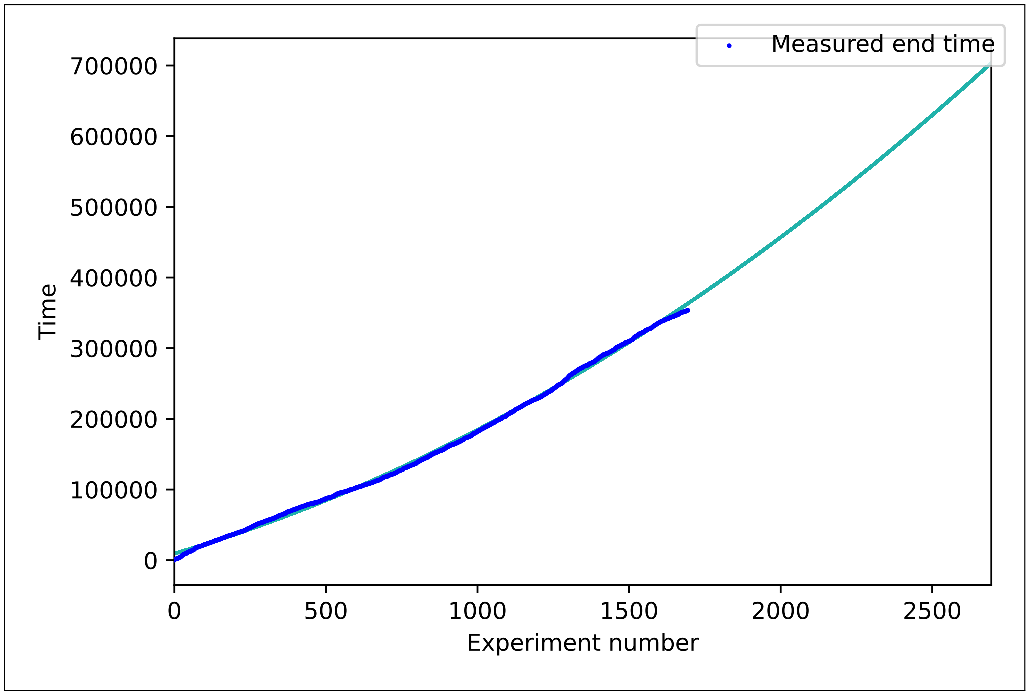


Figure 35

3.5 Further ideas

I was limited in this question by time (and skill), but there are a few things I noted that I would've liked to attempt to model further.

Firstly, there appears to be a periodic element in the data, as shown by the "zoomed in" plot in Figure 17e. There are repeated "bumps" to the start (and therefore end) times. I chose to discount these "bumps" as being encompassed in the uncertainty of a linear model. However, there may be scope to explore periodic trends. For example, inspired by Question 2, one could implement Gaussian processing to model this periodicity.

Another formulation of this artefact is by considering a histogram of the number of "starting times" for each hour of the day. This is shown below, and on first inspection, it appears possible this could be fitted by some periodic function.

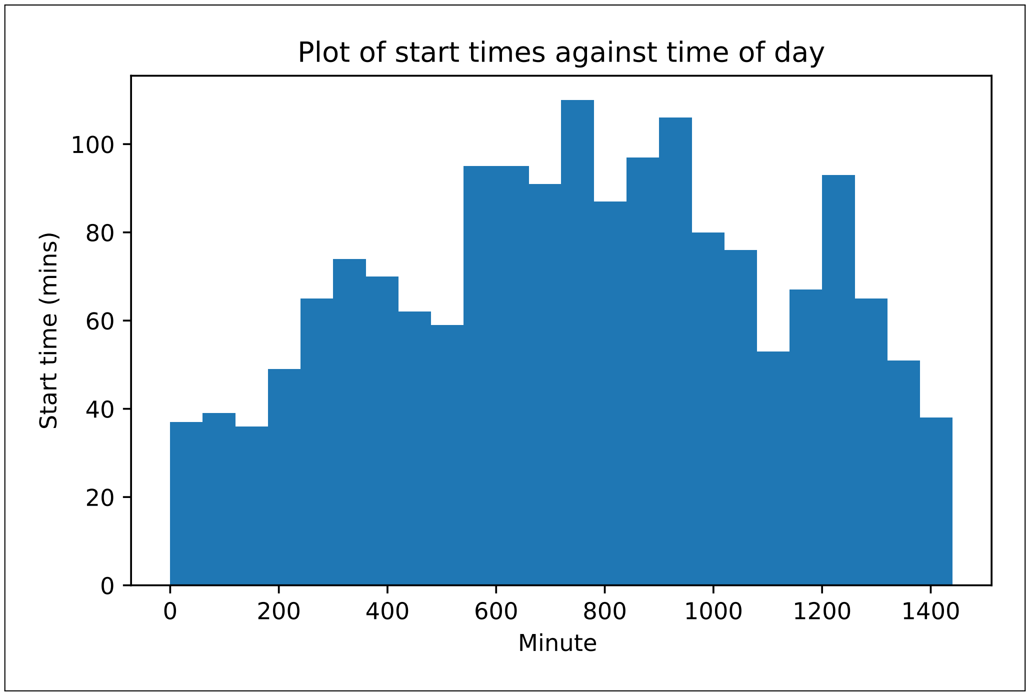


Figure 36

In addition, I plotted the total number of starting times (so, measurements) per day in the data, as seen below.

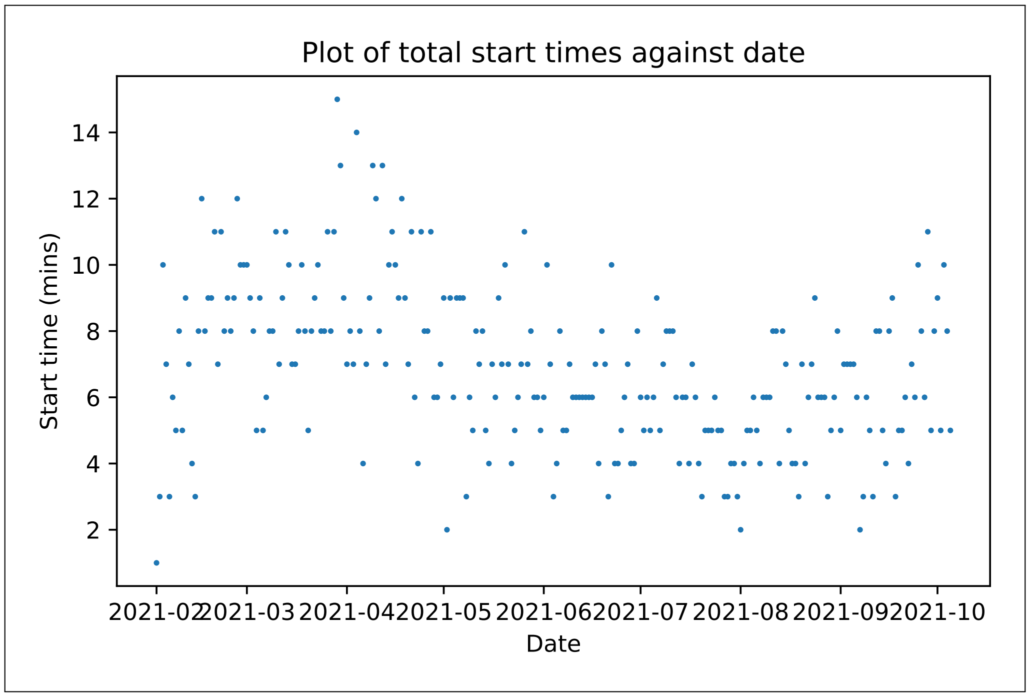


Figure 37

The number of start times per day appears to be experiencing a slight decreasing trend over the course of the months. This could be another modelled trend, to predict the number of measurements that would occur in future days.