EMERGING MINDS
JOURNAL FOR STUDENT RESEARCH

ISSN 2976-9094
EMJSR 2026,Vol 4:M93-M110
https://doi.org/10.59973/emjsr.420

Received: 2026-07-20
Accepted: 2026-08-28
Published: 2026-09-15

Modelling Name Dynamics

Bethany Delves[1,*]

[1]University of Portsmouth, School of Mathematics and Physics, Portsmouth, PO1 3QL, UK

[*]Corresponding author: beth.delves03@gmail.com

Abstract - This paper investigates the mathematical modelling of name dynamics, using a stochastic model applied to data from the Office for National Statistics (ONS). Building on the Wright-Fisher model from population genetics, bias is incorporated to represent preferential selection, while mutation is introduced to account for random individual variation in naming behaviour. The parameters are fitted to the ONS name data and multiple formulations of the bias function are evaluated to determine the best representation of observed trends. The results demonstrate that incorporating mutation and a time-dependent bias substantially improves the fit to data and produces realistic forecasts of future trends. The proposed model captures key features of changing name popularity while remaining computationally efficient, illustrating the potential of stochastic population models to capture the dynamics of cultural evolution in naming practices.

Keywords - Onomastics; Wright-Fisher Model; Population Genetics; Stochastic Modelling; Time-dependent bias.

1 Introduction

Throughout history the popularity of a name has changed with each generation, often reflecting the time around them and what is popular. Throughout this project, the aim is to create a mathematical model that can predict the number of babies born with a specific name in the next five or ten years. The study of name dynamics is known as onomastics [1], something that has been studied for centuries to see the social and linguistic impacts names have.
Another similar study of linguistics is dialectology, this is the study of dialect and dialects [2]. Dialects are subdivisions of a particular language, commonly the differences in pronunciations of a word between accents. Dialectology studies how these linguistic variants spread through populations, and how factors such as geography, migration and social networks can change language. These same factors can affect naming practices, as names may rise and fall in popularity through cultural diffusion and social transmission. As a result, the process controlling name dynamics may have similarities with linguistic evolution. There are also studies into social networks which find that dense or multiplex networks tend to be resistant to linguistic change [3]. The structure of a social network can therefore influence the effective population size [4] associated with linguistic transmission because speakers in these networks tend to reinforce existing linguistic variants and resist linguistic change. This will be important later on in the project, as the popularity of a name is often affected by specific influential individuals, such as celebrities, where the name they choose for their child often rises in popularity after that child is born, which can reduce the effective population size.

Onomastics is a subject within the topic of linguistics, which means we can use a model of linguistics and relate it to name dynamics. However, the most common way to show linguistic evolution is to follow genetic models [5]. This is because the variant of a word and the allele of a gene can be treated in the same way, where they are passed through generations, depending on how popular the variant (allele) is. For this project we will be using population genetics to imitate name dynamics as population genetics model frequencies of an allele within a population.
The main models of population genetics are the Hardy-Weinberg equilibrium [6] (frequencies in a population will remain constant), Wright-Fisher model [7, 8] (stochastic model of genetic drift) and deterministic selection models [7] (frequency changes are fixed and predictable). There have been many studies into creating stochastic models of linguistic evolution [9], and in this project we are going to use stochastic processes to model name dynamics. We will start with the Wright-Fisher model [7, 8], and develop it to more accurately fit our name and its dynamics.

2 The Neutral Model

The best place to start our model is by assuming parents name their children following the theory of neutral evolution. This is the theory that all variations in populations, such as different alleles of a gene, have the same probability of being reproduced. The variation is all equally random, and the frequencies of the alleles are moving randomly up or down.
The Wright-Fisher model of neutral evolution [7, 8] is a stochastic model where we assume the entire population reproduces simultaneously, resulting in discrete and non-overlapping generations, within a population of constant size N [10]. Let i be the number of individuals with a given allele, and j the number in the next generation. Then we can simulate the Wright-Fisher model by using the binomial sampling probability

$$P_{ij} = \binom{N}{j} (i/N)^j (1 - i/N)^{N-j} \quad 0 \leq i, j \leq N. \tag{1}$$

Adjusting this model from genetics to linguistics, we first start with two variants A and B, where A is the name of interest and B is every other name. Then let X_t = number of people with name A at time t, so $P_t = \frac{X_t}{N}$, which is the probability of selecting name A at time t. This probability replaces (i/N) in Equation 1. We also need a population size, N, and time, t, in years.
Following Wright-Fisher's model, this is binomially distributed with

$$X_{t+1} \sim \text{Binomial}(N, P_t).$$

We can now rewrite Equation 1 as:

$$\mathbb{P}(X_{t+1} = j | X_t = i) = \binom{N}{j} (P_t)^j (1 - P_t)^{N-j} \quad 0 \leq X_t, X_{t+1} \leq N. \tag{2}$$

However, the neutral model alone is not an accurate model of linguistic evolution, there are many outside influences to language in the real world. One of these influences is 'bias', where one word is the preferred choice over the other due to prestige, identity or linguistic factors. Prestige refers to the social status or value associated with a word, often resulting in a variant chosen that is linked to socially influential groups [3]. Another way language is influenced is through 'conformity' (or 'accommodation'), which is the idea that an individual is more likely to use the more popular word, studies have found that tightly connected social networks promote conformity to existing social norms [3]. The last influence to language that we will model is 'mutation', where an individual picks randomly between two variants, unrelated to the rest of the population. The probabilities for each of these models are, letting $f = X_t/N$:

$$\begin{aligned}
&\text{Neutral: } P_t = f \\
&\text{Biased: } P_t = bf(1 - f) + f, \text{where } b = \text{bias} \\
&\text{Conformity: } P_t = kf(1 - f)(2f - 1) + f, \text{where } k = \text{conformity} \\
&\text{Mutation: } P_t = \varepsilon(0.5 - f) + f, \text{where } \varepsilon = \text{mutation rate}
\end{aligned} \tag{3}$$

The probabilities for the four models, plotted against $\frac{X_t}{N}$, are shown in Figure 1.

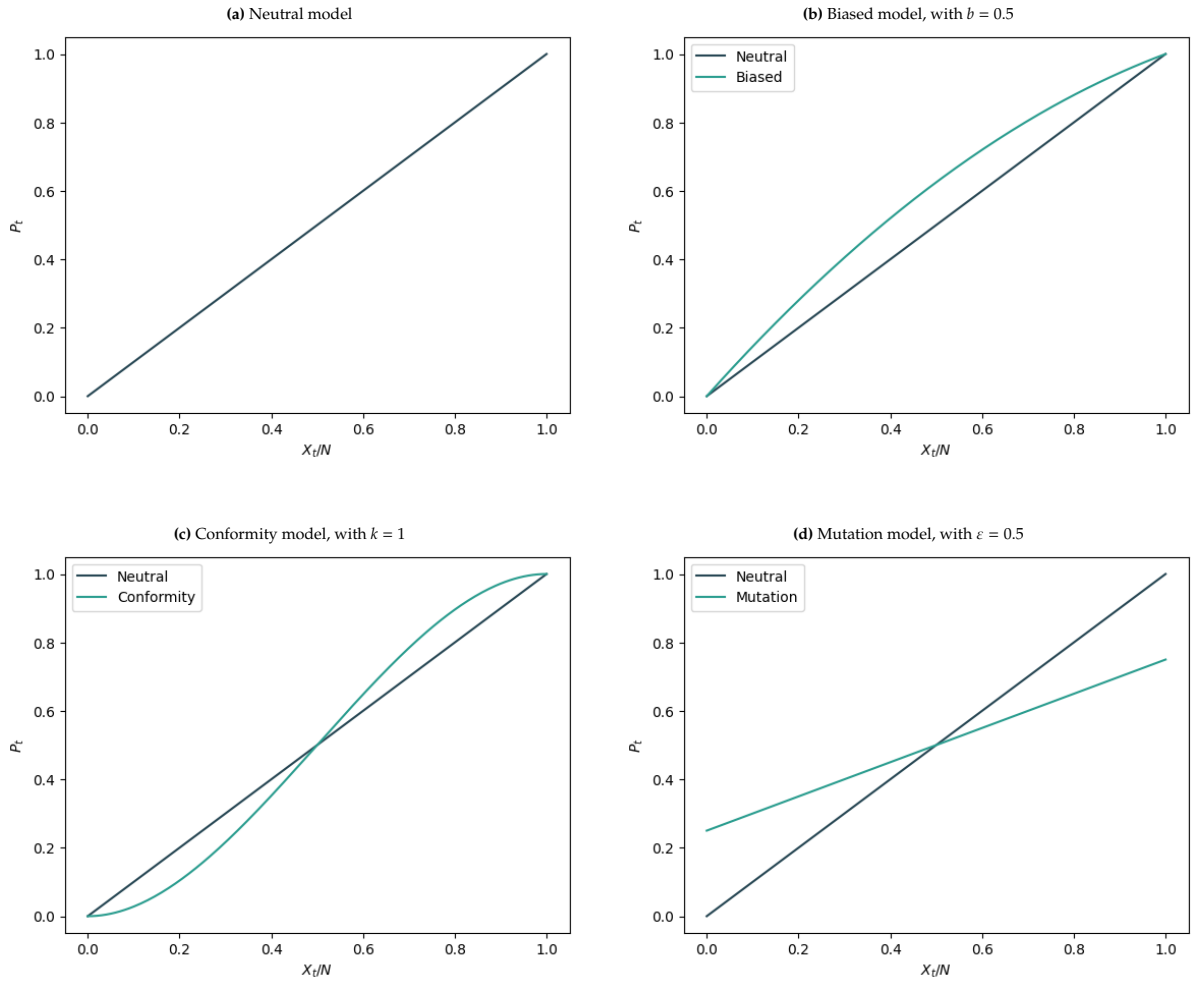


Figure 1: Plots of X_t/N by P_t, for each of the language influences

2.1 Inferring parameters at micro-model

Starting with just the bias model, from Equation 2, the probability is:

$$\begin{aligned}\mathbb{P}(X_t = j | X_{t-1} = i) &= \binom{N}{j} (P_{t-1})^j (1 - P_{t-1}))^{N-j} \quad \text{where} \quad P_t = bf(1-f) + f \\ &= g(X_t | X_{t-1}, b)\end{aligned}$$

Here we introduced g to shorten notation. To guess what b is from a dataset, we use the principle of maximum likelihood, where the likelihood is

$$\begin{aligned}\mathcal{L}(b; X) &= \mathbb{P}(X | given\ b) \\ &= \prod_{t=1}^{T} g(X_t | X_{t-1}, b)\end{aligned}$$

Then to find an estimation of b, we take the argmax of the likelihood function

$$\hat{b} = \underset{b}{\operatorname{argmax}}\, \mathcal{L}(b; X).$$

However putting the long binomial formula into the likelihood function will make the calculations numerically unstable due to having to multiply many probabilities. Instead we can take the natural log of the likelihood function, called the log likelihood:

$$
\begin{aligned}
l(b;X) &= \log \mathcal{L}(b;X_t) \\
&= \log\left(\prod_{t=1}^{T} g(X_t|X_{t-1},b)\right) \\
&= \sum_{t=1}^{T} \log g(X_t|X_{t-1},b) \\
&= \sum_{t=1}^{T}\left(\log\binom{N}{X_t} + \log p(X_{t-1},b)^{X_t} + \log(1-p(X_{t-1},b))^{N-X_t}\right) \\
&\text{using log rules we can write this as:} \\
&= \sum_{t=1}^{T}\left(\log\binom{N}{X_t} + X_t \log p(X_{t-1},b) + (N-X_t)\log(1-p(X_{t-1},b))\right) \\
&= \sum_{t=1}^{T}\left(X_t \log p(X_{t-1},b) + (N-X_t)\log(1-p(X_{t-1},b)) + \text{constant}\right)
\end{aligned}
$$

We can rename $\log\binom{N}{x}$ to a constant with respect to b, because it does not include b, the parameter we are interested in. This also means it can be ignored as we only need values relating to b to find an estimate of b. This gives us the simplified log likelihood as

$$l(b;X) = \sum_{t=1}^{t}\left(X_t \log p(X_{t-1},b) + (N-X_t)\log(1-p(X_{t-1},b))\right). \tag{4}$$

The way of estimating b is the same for both the likelihood function and the log likelihood function, we find the argmax.

$$\hat{b} = \underset{b}{\operatorname{argmax}}\, l(b;X)$$

2.2 Testing the micro-model

To test the accuracy of the log likelihood function in finding bias, we can use a fixed bias chosen by us ($b = 0.01$), and the numpy random binomial with the biased probability (Equation 3) $P_t = bf(1-f)+f$ to generate 100 datasets. For each dataset we can find the log likelihood for a range of guesses of b and take the argmax of the likelihoods to find a singular estimate of $\hat{b}$. These 100 $\hat{b}$'s we can plot into a histogram, Figure 2, along with our fixed b to see how often the correct value of $\hat{b}$ was found.
As you can see from the histogram, this basic bias model is fairly good at estimating bias, as the highest frequencies are all around the true value of b. However, this model breaks down when the path generated from the binomial distribution reaches the total population, known in genetics as 'fixation' [8]. Since this would be very unlikely to happen in the real world, i.e. everyone in England choosing to give their babies the same name, we can counteract reaching total population size by introducing another probability, mutation (ε).

2.3 Improving the micro-model

Mutation is the idea that an individual acts independently of the rest of the population when selecting either A or B. This means that when there is mutation then the probability name A is chosen is 0.5, however if there is no mutation the probability is the same as the neutral model (f). This can be written into the equation below, which rearranges to the probability shown in Equation 3.

$$
\begin{aligned}
P_t &= (1-\varepsilon)f + \frac{\varepsilon}{2} \\
&= \varepsilon\,(0.5-f) + f
\end{aligned}
$$

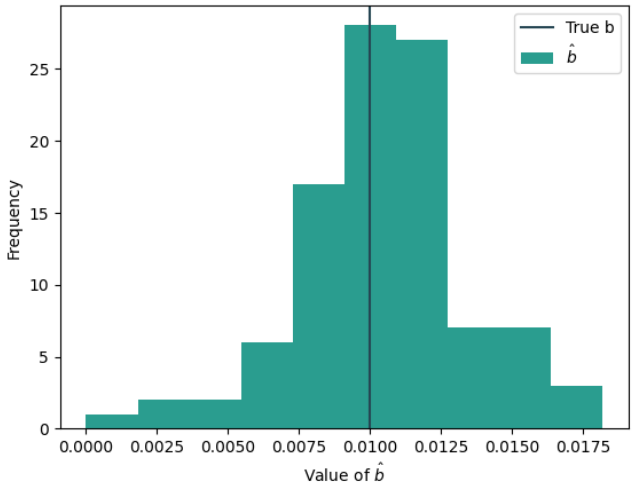


Figure 2: Histogram of $\hat{b}'s$, with the true b

This equation allows us to see how the mutation rate affects the probability of using A, where the term $(0.5 - f)$ determines the direction the probability is pulled. If $f < 0.5$, then $(0.5 - f) > 0$, and $P_t > f$, so the data is pulled up away from 0, towards the centre. If $f > 0.5$, then $(0.5 - f) < 0$, and $P_t < f$, so the data is pulled down away from the total population, N, towards the centre again. As $(0.5 - f)$ is multiplied by ε, this means that the bigger the mutation rate is, the stronger the data is pulled towards the centre.
To add mutation into our model, all we need to do it add the bias and mutation probabilities together, which gives:

$$\begin{aligned} P_t &= bf(1-f) + f + \varepsilon(0.5 - f) + f \\ &= bf(1-f) + \varepsilon(0.5 - f) + 2f \end{aligned}$$

However, we have just made the last term $2f$, which if we kept this, it would takeover the probability, as we are counting the base probability twice. This would mean that the probabilities would become much too large, (over 1), while also making b and ε ineffective. To correct this all we need to do is ignore the extra f, giving us the new combined probability of:

$$P_t = bf(1-f) + \varepsilon(0.5 - f) + f. \tag{5}$$

Using this new probability we can keep every other aspect of the model the same, where we still find the log likelihood and maximise it, but this time with two unknown parameters, $\hat{b}$ and $\hat{\varepsilon}$. So the log likelihood now looks like this:

$$\begin{aligned} l(b, \varepsilon; X) &= \log \mathcal{L}(b, \varepsilon; X) \\ &= \sum_{t=1}^{T} X_t \log p(X_{t-1}, b, \varepsilon) + (N - X_t) \log(1 - p(X_{t-1}, b, \varepsilon)) \end{aligned}$$

Where we can estimate

$$\hat{b}, \hat{\varepsilon} = \underset{b,\varepsilon}{\operatorname{argmax}}\, l(b, \varepsilon; X).$$

We can repeat the same method used to create the histogram in Figure 2, this time finding estimates for $\hat{b}$ and $\hat{\varepsilon}$, the code used to obtain these estimates is in Appendix A. These parameters can be plotted into the histograms in Figure 3. The estimates of $\hat{\varepsilon}$ are close to the true value of ε (0.01), with very little variation around the true value, while the histogram for $\hat{b}$ has more variance, but still the majority of estimates are close to the true value (0.03). Now that we have a model that is fairly accurate using simulated data from the binomial distribution, we need to test it on real world data. It is unlikely our model will be as accurate for real world data as it is for simulated data; we will probably need to adjust the model to improve the fit to name dynamics.

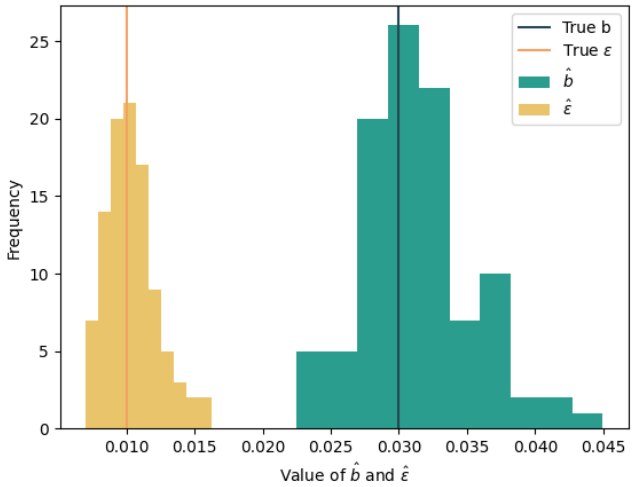


Figure 3: Histogram of $\hat{b}$'s and $\hat{\varepsilon}$, with the true $b = 0.03$ and $\varepsilon = 0.01$

3 Testing the model with data

Now that we have a basic model for finding and estimating bias within data, we can input real world data to see how well our model works. Using a dataset with counts of baby girl names from babies born every year in England and Wales between 1996 and 2024, from the Office for National Statistics [11], we can start with fitting the model to only one name. For example, we can choose the most popular name for 2024, Olivia, and take an array of these counts for each year. This array will become our dataset, $\mathcal{D}$, where we can fit the model with probabilities as either 'Olivia' or 'not Olivia'.
Then, all we need to do is put $\mathcal{D}$ into the log likelihood function to find a $\hat{b}$ and $\hat{\varepsilon}$ for the real world data. The likelihood is

$$l(b, \varepsilon; \mathcal{D}) = \sum_{t=1}^{T} \mathcal{D}_{t+1} \log p(\mathcal{D}_t, b, \varepsilon) + (N - \mathcal{D}_{t+1}) \log(1 - p(\mathcal{D}_t, b, \varepsilon)).$$

Where

$$p(\mathcal{D}_t, b, \varepsilon) = b\left(\frac{\mathcal{D}_t}{N}\right)\left(1 - \frac{\mathcal{D}_t}{N}\right) + (1 - \varepsilon)\left(\frac{\mathcal{D}_t}{N}\right) + \frac{\varepsilon}{2}.$$

For this model we assume the population is $N = 600,000$, the average number of babies born each year in England and Wales, and let $t = 29$ years.

3.1 Inputting the neutral model

Before we can input data into our model made in Chapter 2, we need to split our data, $\mathcal{D}$, into training and test sets. We can take the first 19 years (1996 to 2014) to train the model, $\mathcal{D}_{train}$ and the last 10 years (2015 to 2024) as our test set, $\mathcal{D}_{test}$. The reason we do this is to train our model on part of the data, $\mathcal{D}_{train}$, finding an estimate for bias and mutation. We can then use our $\hat{b}$ and $\hat{\varepsilon}$ to predict the next ten years, and compare that with our test data, $\mathcal{D}_{test}$, to see how accurate our model is at predicting unseen data.
Implementing this into python using the code given in Appendix A, we get values of $\hat{b} = -0.364$ and $\hat{\varepsilon} = 5.75 \times 10^{-3}$ to 3 significant figures. We can use these estimates to make a prediction of $\mathcal{D}_{test}$ with the given initial value, the name count at 2015, which for 'Olivia' was 4853. Repeating this ten times to get ten predicted paths, then plotting on the same graph as the true data, $\mathcal{D}$, we can see how well our model holds up. We shall call this model A, as we are going to develop this model further, so labelling each model will make comparisons easier.
As you can see from Figure 4, model A performs poorly when predicting $\mathcal{D}_{test}$. This is possibly due to there not being enough noise in our current approach, as real life is a lot more unpredictable than our current model assumes. One way to introduce more noise is making the population size smaller, creating an effective population size, N_e [4].

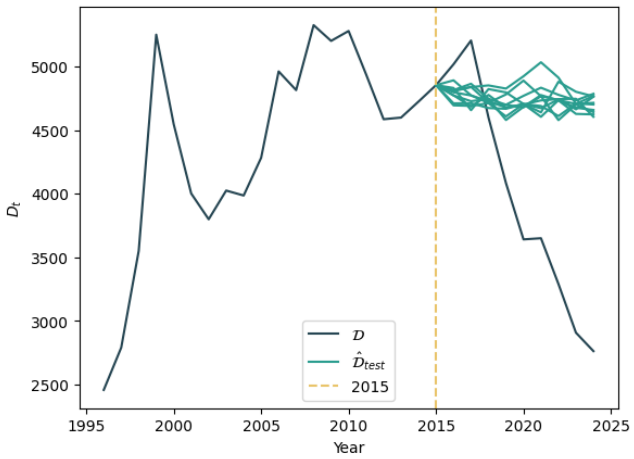


Figure 4: Ten predicted paths of Model A, using $\hat{b} = -0.364$ and $\hat{e} = 5.75 \times 10^{-3}$

3.2 Cross-Entropy loss

Before improving our model, we need a mathematical way to evaluate the accuracy of predictions from model A, and any future models we will create. One way to do this is to use the cross-entropy loss function [12], which measures the difference between the probability distributions of the true data and the predicted data. The smaller the value of cross-entropy, the more closely the probability distributions match, indicating that the model's predictions are more accurate.
First, we assume that y is our test data, and let $q(y)$ be the probability distribution of the true data, $\mathcal{D}$, with $p(y)$ as the probability distribution of the predicted data. Then we can find the cross-entropy using the following formula, which is equal to the negative log-likelihood [12],

$$h(p, q) = -\sum_{y} q(y) \log p(y; \theta)$$

This formula for cross-entropy sums over all classes, but as we only have two classes, we can use the binary cross-entropy loss, which is defined as:

$$h(p, q) = -\frac{1}{n} \sum_{y} q(y) \log p(y) + (1 - q(y)) \log(1 - p(y))$$
$$\text{where } n = \text{size of } \mathcal{D}_{test}$$

If we have a model of many predicted paths, such as in Figure 4, we can find the cross-entropy for each predicted path then find the mean of those to give us an average of how well our model fits the test data. Just finding the cross entropy for one path gives us a value of $h_A = 0.2460$, for model A, while finding the average of 100 paths gives us the a mean of $\bar{h}_A = 0.2459$, so there is very little difference between one path and the average of many. While this value of $\bar{h}_A$ is fairly small, implying model A's predicted probability distribution is close to the observed one, we want cross entropy to be as close to 0 as possible, so there is plenty of room for improvement.
We can compare cross-entropy between different models, as long as we are using the same prediction of unseen data and the same population size. We will come back to this later once we have more models with the same N, to help us determine the most accurate one.

3.3 Introducing effective population size

In model A we chose the population size to be the amount of babies born each year, $600,000$, however, the maximum value of $\mathcal{D}$ (5325), is very small in comparison. This may be a factor in the poor predictions of model A, as an extremely large N will make the noise very small. Therefore, a simple way we can try to improve our model is simply by reducing the size of N from $600,000$ to

10,000, this new model we will refer to as model B. This introduces the idea of an effective population size N_e [4], due to a small number of influential individuals.
Nothing else in our method is affected, we simply just change the population size to $N_e = 10^4$, which gives us parameters of $\hat{b} = -0.0450$ and $\hat{\varepsilon} = 0.365$, which we can use to make ten predicted paths. These are plotted below.

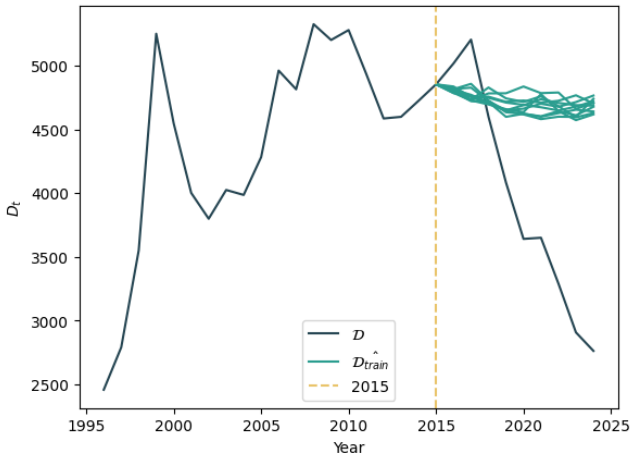


Figure 5: Plot of ten predicted paths of model B with $\hat{b} = -0.0450$, $\hat{\varepsilon} = 0.365$ and $N_e = 10^4$

It is clear from Figure 5 that a smaller population size does not significantly improve the model. Both models A and B do not appear to have enough noise to cause the random variations the actual data has. The average cross-entropy for this model is $\bar{h} = 0.6801$, which is actually higher than model A, implying that the predicted probability distribution is not as close to the observed probability distribution. This means that we need a different way to develop our model, just reducing the population size has not made any improvement from model A.

3.4 Introducing a scale factor

Another way we can improve our model is to introduce a scale factor, α, this is going to be our next model, model C. We can divide both the population size, N and our data, $\mathcal{D}$ by this scale factor, using floor division to keep both the data and the population size as integers. This changes the distribution to

$$\mathcal{D}_t \sim \text{Binomial}\left(\left\lfloor \frac{N}{\alpha} \right\rfloor, P_{t-1}\right).$$

Scaling by α leads to more noise within the model as the probabilities do not change, but the effective population size, N_e, will get smaller as α increases. The smaller N_e is, the larger variance will be, which increases noise in our model. This is clear from the formula for variance for a binomial distribution.

$$\begin{aligned}
&\sigma^2 = np(1-p)\\
&\text{replacing } n \text{ and } p \text{ with our population and probability}\\
&\quad = \left\lfloor \frac{N}{\alpha} \right\rfloor P_t(1-P_t)\\
&\quad = N_e P_t(1-P_t)
\end{aligned}$$

As you can see from the equation above, we have renamed $\left\lfloor \frac{N}{\alpha} \right\rfloor$ to N_e to make future notation shorter. We can also rename $\left\lfloor \frac{\mathcal{D}_t}{\alpha} \right\rfloor$ to $\tilde{\mathcal{D}}_t$.

3.4.1 Finding the optimal scale factor (α)

Before we can input α into model C, we need to find the optimal value that α can be. This is done by finding the α value that gives the closest variance to our observed data. We shall start with a

range of values of α, between 10 and 100. For each value of α, we scale $\mathcal{D}$ and N by α, then find the corresponding $\hat{b}$ and $\hat{\varepsilon}$. We are going to find the variance per step of the model, where each step is

$$\Delta\tilde{\mathcal{D}}_t = \tilde{\mathcal{D}}_{t+1} - \tilde{\mathcal{D}}_t.$$

From here we can find the probability of each step using the formula derived in Equation 5, then find the variance of each step using the formula

$$\mathbb{V}(\Delta\tilde{\mathcal{D}}_t) = N_e P_t(1 - P_t).$$

However, finding just this variance is the randomness or noise of our model, but we also have drift, reliant on bias and mutation. Drift is the expected change of our model per step, where the formula is

$$\mathbb{E}(\Delta\tilde{\mathcal{D}}_t) = P_t N_e - \tilde{\mathcal{D}}_t.$$

Now combining this through the law of total variance

$$\begin{aligned}\mathbb{V}(\Delta\tilde{\mathcal{D}}_t) &= \mathbb{E}(\mathbb{V}(\Delta\tilde{\mathcal{D}}_t|\tilde{\mathcal{D}}_t)) + \mathbb{V}(\mathbb{E}(\Delta\tilde{\mathcal{D}}_t|\tilde{\mathcal{D}}_t)) \\ &= \mathbb{E}(N_e P_t(1 - P_t)) + \mathbb{V}(P_t N_e - \tilde{\mathcal{D}}_t)\end{aligned}$$

Now that we have the variance using our estimated parameters $\hat{b}$ and $\hat{\epsilon}$, we can find the observed variance for the value of $\Delta\tilde{\mathcal{D}}_t$. This is much simpler as we can just use the numpy variance function, we can call this $\mathbb{V}^{obs}$. Now that we have the variances, we can find the differences, and square them to ensure all values are positive,

$$\eta(\alpha) = (\mathbb{V}^{obs} - \mathbb{V})^2.$$

Finding a value of $\eta(\alpha)$ for each value of α, we can plot this in a graph, Figure 6. With our list of $\eta(\alpha)$ values, we can use the numpy argmin function to find the index of the minimum value of $\eta(\alpha)$, the code for this is in Appendix A. The minimum value is $\eta(\alpha) = 0.238$ to 3 significant figures and corresponding optimal value of $\alpha = 56$.

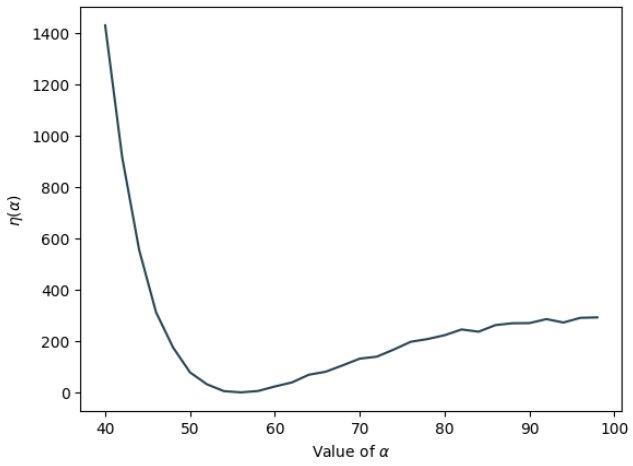


Figure 6: Plot of values of α

In Figure 6 we have only plotted the values of $\eta(\alpha)$ between $\alpha = 40$ and $\alpha = 100$, this is because the minimum value of $\eta(\alpha)$ is extremely small, while values of $\eta(\alpha)$ are extremely large the smaller α is.

3.4.2 Testing the model

Now that we have an optimal value of α (56), we can simulate many paths, like we did for our earlier models. The parameter estimates are $\hat{b} = -0.362$ and $\hat{\epsilon} = 5.69 \times 10^{-3}$, and the cross-entropy is $\bar{h} = 0.0399$.

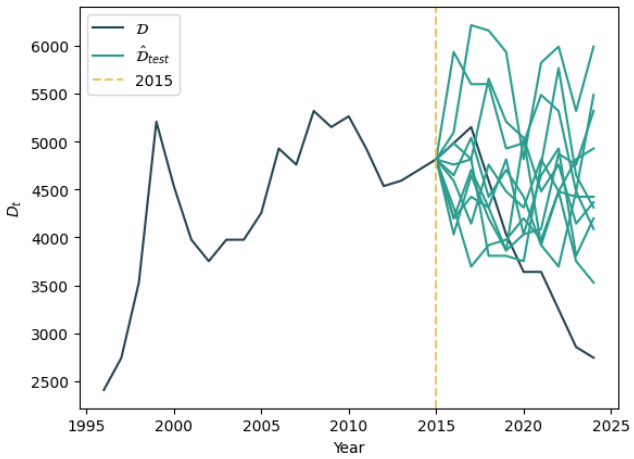


Figure 7: Plot of ten predicted paths for model C where $\hat{b} = -0.362$, $\hat{\varepsilon} = 5.69 \times 10^{-3}$, scaled with $\alpha = 56$

To visualise the simulated paths, we can plot them as they are with the scaled data, $\tilde{\mathcal{D}}$, but to make predictions of the true counts, we have to scale it back up to match $\mathcal{D}$. To do this all we need to do is multiply the predicted paths by α, and plot these with $\mathcal{D}$, in Figure 7. From these plots, we can see that model C has a lot more noise in the predicted paths than models A and B, this should mean that the predictions for this model are also much more accurate. This is shown in the mean cross entropy for our ten simulated paths, $\bar{h}_C = 0.0399$, which is extremely small and means model C is very accurate in the predictions of unseen data.

3.4.3 Visualising more simulated paths

As our models get more noisy, it gets harder to visualise predicted paths clearly, particularly if we want to simulate more than ten paths at a time. A good way to visualise this is to instead take the 10th and 90th percentiles to show the area where the majority of simulated paths fall at each time step. This is shown below in Figure 8.

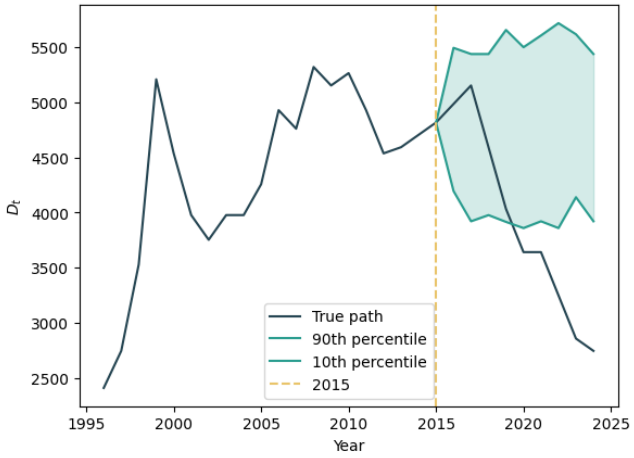


Figure 8: Plot of percentiles for model C where $\hat{b} = -0.362$, $\hat{\varepsilon} = 0.006$, scaled with $\alpha = 56$

This graph shows us model C is more accurate at predicting data within the first five years, getting less and less accurate as we get further from the initial value, as the path of the true data does not stay within the percentile range after 2019. Finding the mean cross-entropy for the 100 paths is $\bar{h}_C = 0.0399$, the same as for our ten simulated paths.

4 Time dependant bias

Now that we have a model that appears to match the noise of the true data, we can develop it further to make predictions match the unseen data more accurately. We will keep using the scale factor, α,

however as we introduce time-dependant bias we will need to find the optimal α for each model as changing the bias will affect the value of α. Our b parameter now changes from a constant to a polynomial that looks like

$$b(t) = b_0 + b_1 t + b_2 t^2 + \cdots + b_k t^k.$$

This allows the bias to vary over time, which should allow for more movement in the predictions unrelated to noise.
Changing bias also affects the probability in Equation 5 such that it is now

$$P_t = b(t)f(1-f) + \varepsilon(0.5 - f) + f.$$

In the next sections we are going to consider three degrees of polynomial (linear, quadratic and cubic) to see if they provide more accurate models than model C.

4.1 Implementing Linear Bias

Starting polynomial bias at the first degree, we can allow bias to look like:

$$b(t) = b_0 + b_1 t.$$

Inputting this linear bias into our model is very simple, we still find an optimal α, then scale our data by that α. Next we find the log likelihood, with the parameters to estimate being $\hat{b}_0$, $\hat{b}_1$ and $\hat{\varepsilon}$. Finally taking the argmax of this function gives us the following estimates

$$\alpha = 54,\ \hat{b}_0 = -0.452,\ \hat{b}_1 = 3.66 \times 10^{-3},\ \hat{\varepsilon} = 1.74 \times 10^{-3}$$
$$\hat{b}(t) = -0.452 + (3.66 \times 10^{-3})t$$

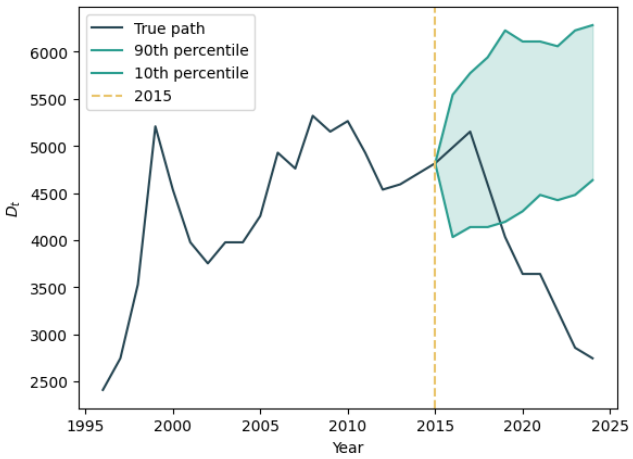


Figure 9: Plot of percentiles for model D where $\hat{b}_0 = -0.452$, $\hat{b}_1 = 3.66 \times 10^{-3}$, and $\hat{\varepsilon} = 1.74 \times 10^{-3}$, scaled by $\alpha = 54$.

Then, simulating 100 predicted paths using these estimates, finding the 10th and 90th percentiles and plotting them, shown in Figure 9. Also finding the cross-entropy of each of the 100 paths, we find the mean is $\bar{h}_D = 0.0401$.
Since this model has the same population size and is predicting the same unseen data as model C, we can compare their cross-entropies. $\bar{h}_C = 0.0399$ and $\bar{h}_D = 0.0401$, this means that the differences in probability distributions are very similar between models. While model C does have the smaller cross-entropy, implying it is the more accurate model, the difference is so insignificant (2×10^{-4}) that we cannot make any clear conclusions. We need another way to compare our models.

4.1.1 Comparing constant and linear bias

There are many methods to compare models, however we are going to use one of the simplest and most common methods, the mean squared error (MSE). The formula is

$$MSE = \frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)^2.$$

Where n is the number of observations (10), y_t is the observed value at time t (from $\tilde{\mathcal{D}}_{test}$) and $\hat{y}$ is the corresponding predicted value. The mean squared error takes the counts of the data, instead of the probabilities like cross-entropy does. This should give us larger differences between models, which will make it easier to analyse, Therefore we can do a paired t-test comparing the difference between the mean squared errors of model C and model D.
If we let $MSE_{C,i}$ be the mean squared error in simulation i for model C and $MSE_{D,i}$ be the mean squared error for simulation i, where $i = 1, 2, \cdots n$, and n is the number of simulations (100). Then $d_i = MSE_{C,i} - MSE_{D,i}$, the difference in mean squared errors in simulation i. Therefore the hypothesis is

$$\begin{aligned} &H_0 : \mu_d = 0 \\ &H_1 : \mu_d \neq 0 \qquad \text{where } d = MSE_C - MSE_D \end{aligned}$$

The mean of the differences is $\bar{d} = -386$ and the standard deviation is $s = 449$, both to three significant figures. Since n is large (100), and this is a two sided test at the 5% level, we can assume the critical value is 1.96 and the test statistic is

$$\begin{aligned} t_0 &= \frac{\bar{d}}{s/\sqrt{n}} \\ &= \frac{-386}{449/\sqrt{100}} \\ &= \frac{-386}{44.9} \\ &= -8.60 \end{aligned}$$

The absolute value of the test statistic is greater than the critical value ($|-8.60| > 1.96$), therefore we can reject H_0 at the 5% level, there is a significant difference between the mean squared errors of models C and D. We can also find the 95% confidence interval:

$$\begin{aligned} 95\%\ CI : \bar{d} &\pm 1.96\left(\frac{s}{\sqrt{n}}\right) \\ &= -386 \pm 1.96\left(\frac{449}{\sqrt{100}}\right) \\ &= -386 \pm 88.00 \\ &= (-474 \text{ to } -298) \end{aligned}$$

As the confidence interval does not include 0, our conclusion above is proven to be correct, there is a significant difference in the mean squared errors. Since the confidence interval is negative, we can say that model D, the linear bias, has a higher mean squared error, which means that model C, with a constant bias, is better than model D. This conclusion is what we see in the graphs as Figure 8's percentile range covers more of the true data.

4.2 Implementing Quadratic Bias

The next model we can make, model E, is a quadratic bias, where

$$b(t) = b_0 + b_1 t + b_2 t^2.$$

Again, this is made in the same way as model D, just with an extra parameter to estimate, b_2. Estimating this in python gives the following estimates:

$$\alpha = 52,\ \hat{b}_0 = -0.557,\ \hat{b}_1 = 0.0181,\ \hat{b}_2 = -7.45 \times 10^{-4},\ \hat{\varepsilon} = 7.31 \times 10^{-3}$$
$$\hat{b}(t) = -0.547 + 0.0181t - (7.45 \times 10^{-4})t^2$$

In Figure 10 we simulated 100 paths with these estimates and plotted their 10th and 90th percentiles. This graph appears to show a much better fit than models C and D (in Figures 8 and 9 respectively),

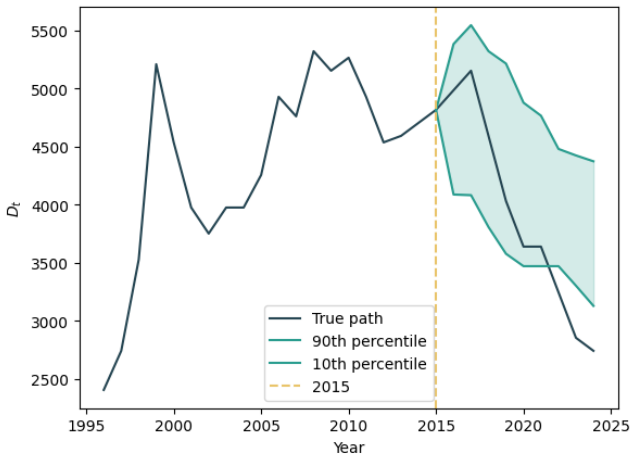


Figure 10: Plot of percentiles for model E where $\hat{b}_0 = -0.557$, $\hat{b}_1 = 0.0181$, $\hat{b}_2 = -7.45 \times 10^{-4}$, and $\hat{\varepsilon} = 7.31 \times 10^{-3}$, scaled with $\alpha = 52$

however the average cross-entropy of model E is $\bar{h}_E = 0.0398$. This cross-entropy is not significantly different from models C and D, meaning that fit of the probability distributions are all very similar. We can instead find the mean squared error, which is 189, however to interpret it we need to square root, then multiply by α so as to match the values in Figure 10, which gives us

$$\begin{aligned} &RMSE: \sqrt{189} = 13.7 \\ &\text{scale back up by } \alpha \\ &RMSE: 13.7 \times 52 = 712 \end{aligned}$$

This value tells that on average model E's predictions differ from the observed number of births by 712 names. Although this appears large, the true range of the data is from around 3,000 to 5,000, therefore this is quite a low error, especially considering the majority of these large errors come from the later five years of predictions.

4.2.1 Comparing constant and quadratic bias models

In section 4.1.1 we found model C to be more accurate than model D, so we can just test the differences of model E and model C to see which is the most accurate. We can mathematically test this using the same method in section 4.1.1, a paired t-test of the mean squared errors. The hypothesis is

$$\begin{aligned} &H_0 : \mu_d = 0 \\ &H_1 : \mu_d \neq 0 \qquad \text{where } d = MSE_C - MSE_E \end{aligned}$$

The mean of the differences is $\bar{d} = 332$ and the standard deviation is $s = 271$, both to three significant figures. The critical value is still 1.96, and the test statistic is

$$\begin{aligned} t_0 &= \frac{331}{271/\sqrt{100}} \\ &= \frac{331}{27.1} \\ &= 12.2 \end{aligned}$$

The absolute value of the test statistic is greater than the critical value ($|12.2| > 1.96$), therefore we can reject H_0 at the 5% level, there is a significant difference between the mean squared errors of models C and E. The confidence interval is

$$\begin{aligned} 95\%\ CI : 331 &\pm 1.96\left(\frac{271}{\sqrt{100}}\right) \\ &= 331 \pm 53.1 \\ &= (277 \text{ to } 384) \end{aligned}$$

This confidence interval does not include 0, therefore we can reject H_0, the same as for our test statistic. As the confidence interval is positive, we can say that model E has a lower mean squared error than model C and can assume that it is a more accurate model.

4.3 Implementing Cubic Bias

The final model we are going to create, model F, is with a cubic bias, where

$$b(t) = b_0 + b_1 t + b_2 t^2 + b_3 t^3.$$

This is the same as the other polynomial bias models, D and E, where we find the parameter estimates to be:

$$\alpha = 52,\ \hat{b}_0 = -0.457,\ \hat{b}_1 = -0.0173,\ \hat{b}_2 = 3.79 \times 10^{-3},\ \hat{b}_3 = -1.63 \times 10^{-4},\ \hat{\varepsilon} = 6.78 \times 10^{-3}$$
$$\hat{b}(t) = -0.457 - 0.017t + 0.004t^2 - 0.0001t^3$$

Simulating 100 paths and plotting in the same way as Figures 8, 9 and 10, we get the graph shown in Figure 11. This plot does not appear to be an accurate model at predicting unseen data, as at no point does the percentile range cross the true data.

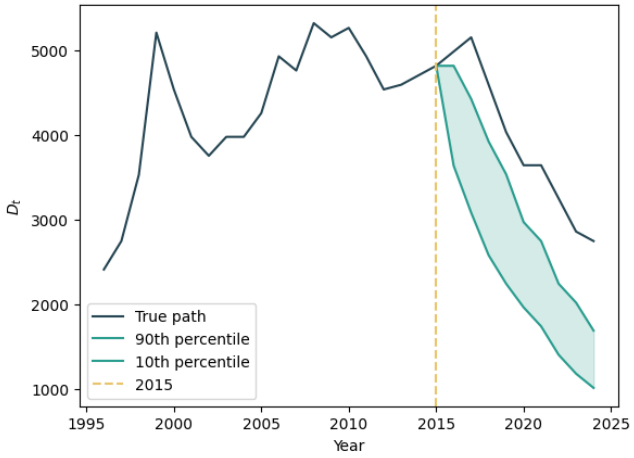


Figure 11: Plot of percentiles for model F where $\hat{b}_0 = -0.457$, $\hat{b}_1 = -0.0173$, $\hat{b}_2 = 3.79 \times 10^{-3}$, $\hat{b}_3 = -1.63 \times 10^{-4}$, and $\hat{\varepsilon} = 6.78 \times 10^{-3}$, scaled with $\alpha = 52$.

The average cross-entropy for this model is $\bar{h} = 0.0402$, which implies this is an accurate model in terms of probabilities, even though we can see from Figure 11 that the predictions do not match the true data. This is a limitation of cross-entropy [12] as it is a measure of the difference in probability distributions not the points of the predicted data, meaning that while $\hat{\mathcal{D}}_t$ is not close to $\mathcal{D}_t$ numerically, the distributions are very similar. We can see this in Figure 11 as the shape of the percentile range is very similar to the shape of the true data.
The bias-variance trade off [13] describes the balance between model flexibility and predictive stability. Bias is known as the systematic error caused by a model being too simple, and variance is how much the model varies depending on the training data. A good model requires a balance of both, however model F has low bias, shown in the low cross-entropy, and high variance because small changes in the training data has led to the sharp downtrend the model predicts.
We could compare this model, like we did for others in this section, however there is no need as it is clear this is not a good model. Therefore model E is still our most accurate model. What we can do now is test model E against completely new data.

5 Testing the model with different names

Throughout this project we have only used one name, 'Olivia', in creating an accurate model. However, there is not just one name that parents can name their child, there are many to choose from,

so before claiming our model is accurate in predicting name dynamics, we need to test its accuracy using other names.
As the full dataset has around 1,400 names, we are not going to look at all of them, we can just take three names with varying popularity, one that starts very popular and decreases significantly ('Sarah'), one that rises and falls a lot ('Millie'), and one that significantly increases in popularity in recent years ('Mabel'). For this whole chapter, we are only going to use model E, as that is our most accurate model.
Using different data, we still fit the model in the same way, first splitting into training and test data, then finding an optimal α, and using the log likelihood function (Equation 4), to estimate $\hat{b}_0, \hat{b}_1, \hat{b}_2$ and $\hat{\varepsilon}$.

5.1 Modelling for 'Sarah'

First we can fit the name 'Sarah', Figure 12, which gives us parameters of $\alpha = 98$, $\hat{b}_0 = -0.331, \hat{b}_1 = -0.0154, \hat{b}_2 = -6.62 \times 10^{-4}$ and $\hat{\varepsilon} = 1.41 \times 10^{-3}$. The mean cross-entropy is $h_S = 5.85 \times 10^{-3}$ and the mean $MSE_S = 4.09$.

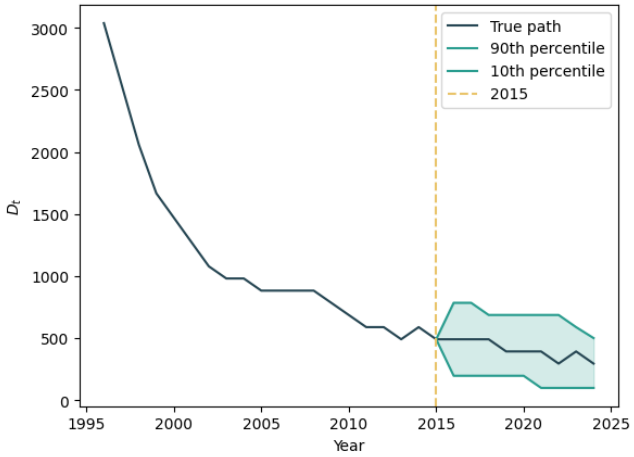


Figure 12: Percentile graph for the name 'Sarah', predicted with parameters $\alpha = 98$, $\hat{b}_0 = -0.331, \hat{b}_1 = -0.0154, \hat{b}_2 = -6.62 \times 10^{-4}$ and $\hat{\varepsilon} = 1.41 \times 10^{-3}$.

From the cross-entropy, mean squared error and Figure 12, model E appears to be good at predicting the name 'Sarah'. However, from the year 2000 onwards the popularity of 'Sarah' is steadily decreasing, which gives a very simple trend for predictions to follow. The cross-entropy being so small (5.85×10^{-3}), means that the probability distribution of the predicted data is very similar to the that of the true data. The mean squared error is also extremely small, which is clear from the plot in Figure 12, as the true path is completely within the percentile range, and generally staying in the centre of that range. Overall, we can say that model E is accurate in predicting the name 'Sarah', but that isn't necessarily an indicator of the quality of the model as this data follows a very simple and predictable downward trend.

5.2 Modelling for 'Millie'

As 'Sarah' had a very simple trend, the next name we can test needs lot more variance, 'Millie'. Fitting this to model E, we get parameters of $\alpha = 24, \hat{b}_0 = -0.0459, \hat{b}_1 = -6.44 \times 10^{-6}, \hat{b}_2 = -5.92 \times 10^{-4}$ and $\hat{\varepsilon} = 8.74 \times 10^{-4}$, this is plotted in Figure 13.
The cross-entropy is $h_M = 0.0166$, which is very small, telling us that the probability distribution for the predicted data is close to that of the true data. While the probabilities are similar, looking at Figure 13 we can see that from 2015 to 2020 the true data is within the percentile range, but after that true data starts to increase, spiking in 2022, while our model continues to predict a decreasing trend. As a consequence, our MSE is quite large, $MSE_M = 460.85$, which we can square root to make it easier to interpret, $RMSE_M = 20.9$. Since the mean squared error was found on our scaled data, to match the RMSE to Figure 13, we need to multiply by $\alpha = 24$. Therefore the root mean squared error



Figure 13: Percentile graph for 'Millie', predicted with parameters $\alpha = 24, \hat{b}_0 = -0.0459, \hat{b}_1 = -6.44 \times 10^{-6}, \hat{b}_2 = -5.92 \times 10^{-4}$ and $\hat{\varepsilon} = 8.74 \times 10^{-4}$.

for the unscaled data is 502 to three significant figures, so on average our prediction was off by 502 counts. Most of this error comes from after 2020, where the model fails to capture a slight increase in popularity.

5.3 Modelling for 'Mabel'

From the names we've looked at so far, model E has generally performed well, particularly in the first five years of the prediction period. However all of these names have started with quite high popularity in 1996 and either decreased or had a lot of variance. Therefore, the last name we are going to look at is 'Mabel', a name that was rare in 1996, and has since risen significantly in popularity. Fitting model E to the name 'Mabel' gives us the following parameters $\alpha = 84, \hat{b}_0 = -9.91, \hat{b}_1 = 1.21, \hat{b}_2 = -0.0376$ and $\hat{\varepsilon} = 1.53 \times 10^{-4}$, where the percentiles graph for 100 simulations is shown in Figure 14.

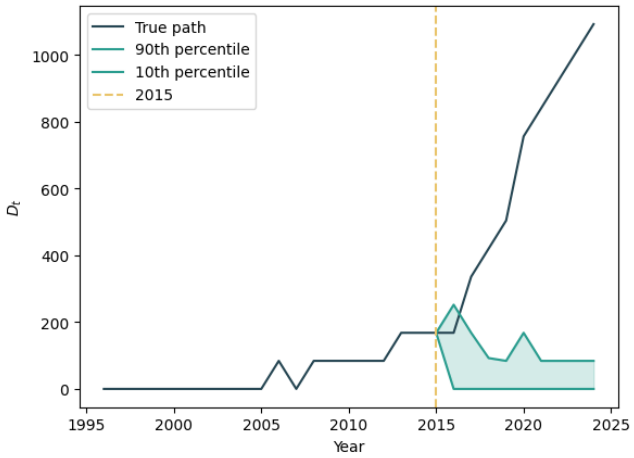


Figure 14: Percentile graph of 'Mabel', predicted with parameters $\alpha = 84, \hat{b}_0 = -9.91, \hat{b}_1 = 1.21, \hat{b}_2 = -0.0376$ and $\hat{\varepsilon} = 1.53 \times 10^{-4}$.

Before finding the cross-entropy or mean squared error, we can see from Figure 14 that our model did not give us accurate predictions for this data. While the true path starts to steadily rise from 2015, the percentiles of the predicted data decreases down to in the range of 0 to 100. The root mean squared error is $RMSE_{Ma} = 8.01$ for the scaled data, so scaling back by α, it is 673 to three significant figures, therefore on average our predictions were off by 673 counts. In contrast, the cross-entropy is $h_{Ma} = 0.0185$, which tells us there is not much difference in the probability distributions of the predicted and true data, despite the model failing to capture the strong upward trend in popularity. These three names have given us a general idea of what happens to our model when we introduce names with different dynamics. Model E performs well when names have consistent trajectories, like

'Sarah', or when there is a lot of variance within the name, like 'Millie'. However the model is not good in predicting significant jumps in popularity, like with 'Mabel'.

6 Conclusion

In conclusion, a stochastic model for name dynamics can predict future frequencies of a specific name with reasonable accuracy, as long as we introduce extra parameters of bias and mutation. Of the models we considered, the most accurate one was model E, with quadratic, time-dependent bias. With our most accurate model, we can now predict the future counts of babies born with the name 'Olivia', as shown in Figure 15. Using the full dataset of counts for 'Olivia', or $\mathcal{D}$, to fit our model, the parameters are

$$\alpha = 42,\ \hat{b}_0 = -0.526,\ \hat{b}_1 = 0.0276,\ \hat{b}_2 = -1.23 \times 10^{-3},\ \hat{\varepsilon} = 6.47 \times 10^{-3}$$
$$\hat{b}(t) = -0.526 + 0.0276t - (1.23 \times 10^{-3})t^2$$

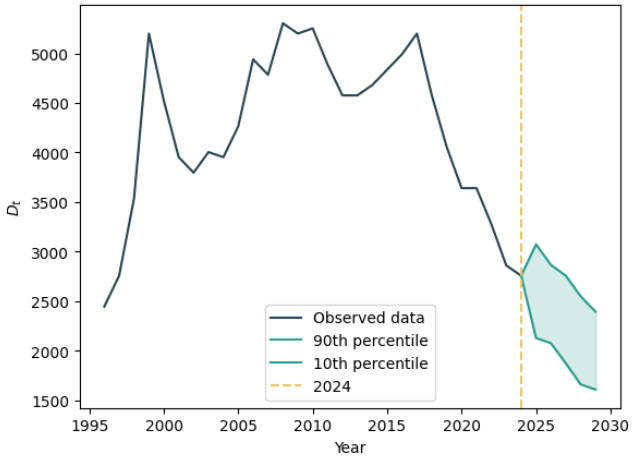


Figure 15: Plot of percentile range for the next five years of the popularity of 'Olivia'

The specific percentile ranges are in Table 1, from these, we can see that 'Olivia' is expected to continue to decline in popularity over the next five years. However in comparison to other names, it will still remain one of the most popular names in England and Wales.

Year	10th percentile	90th percentile
2025	2,184	2,986
2026	1,974	2,856
2027	1,927	2,696
2028	1,788	2,562
2029	1,718	2,398

Table 1: Predicted 10th and 90th percentile ranges for the number of babies named 'Olivia' from 2025 to 2029.

We have found that a quadratic bias model is the most accurate model for the name 'Olivia', and while it performs reasonably well for some names, it also performs poorly for others, particularly names that grow significantly in popularity as time goes on. It appears that our model has the highest chance of making accurate predictions when the data it has been trained on has a lot of variation over time.

If we were to develop this model further, we could test it on all the names in the ONS dataset, and see how often our model accurately predicts the dynamics of a name. From there we can continue to develop the model, possibly by introducing more parameters, with the aim of improving accuracy of predictions across a wider range of names. We could also develop a non-stochastic model, possibly following the Hardy-Weinberg equilibrium or a deterministic selection model, then compare them to our stochastic model to determine whether they provide more accurate predictions of name dynamics. This would further demonstrate how mathematical models from population genetics can be adapted to provide predictions of cultural and linguistic change, including in the dynamics of name popularity.

Acknowledgements

I would like to express my sincere appreciation to my supervisor, Dr. James Burridge for his guidance, feedback, and support throughout this project.

References

[1] C. Hough. *The Oxford Handbook of Names and Naming*. Oxford University Press, 2016.

[2] J. K. Chambers and Peter Trudgill. *Dialectology*. Cambridge University Press, Cambridge, 2 edition, 1998.

[3] Lesley Milroy. *Language and Social Networks*. Wiley-Blackwell, 2 edition, 1987.

[4] D. Deffner, A. Kandler, and L. Fogerty. Effective population size for culturally evolving traits. *PLOS Computational Biology*, 18(4), 2022.

[5] M. G. Newberry, C. A. Ahern, R. Clark, and J. B. Plotkin. Detecting evolutionary forces in language change. *Nature*, 551(7678):223–226, 2017.

[6] G. H. Hardy. Mendelian proportions in a mixed population. *Science*, 28(706):49–50, 1908.

[7] R.A. Fisher. *The Genetical Theory of Natural Selection*. Clarendon Press, Oxford, 1930.

[8] S. Wright. Evolution in mendelian populations. *Genetics*, 16(2):97–159, 03 1931.

[9] A. Karjus, R. A. Blythe, S. Kirby, and K. Smith. Challenges in detecting evolutionary forces in language change using diachronic corpora. *Glossa*, 2020.

[10] S. P. Otto and T. Day. *A Biologist's Guide to Mathematical Modelling in Ecology and Evolution*. Princeton University Press, New Jersey, 2007.

[11] Office for National Statistics. Baby names in england and wales: from 1996 (1996 to 2024 edition), 2025. Accessed: 24 February 2026.

[12] E. Gordon-Rodriguez, G. Loaiza-Ganem, G. Pleiss, and J. P. Cunningham. Uses and abuses of the cross-entropy loss: Case studies in modern deep learning. PMLR, 2020.

[13] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An introduction to Statistical Learning*. Springer, 2013.