Table of Contents

Abstract

There are a range of tools that apply Bayesian statistical techniques to estimate the underlying parameters generating a given set of data, given a statistical model of the generation process and given priors on the parameter values. Such tools typically do not allow inference for parameters representing discrete changes in the underlying statistical model. These "change-points" are important in many situations, notably in epidemiology such change-points represent the emergence of new mutations of a given disease, or the emergence of new treatments. This project will use an existing and well-known tool (the probabilistic programming language Stan) to allow change-point inference for simple datasets and statistical models.

Chapter 1: **Project Specification**

The project uses Bayesian statistical techniques to perform change-point inference, which identifies discrete changes in the underlying parameters of a statistical model. These change points are crucial in scenarios such as epidemiology, where they may represent events like the emergence of new virus mutations or the introduction of treatments. However, traditional Bayesian tools typically lack the capability to handle inference for such discrete changes directly.

This project will use the probabilistic programming language Stan with the R interface to develop methods for inferring change-points in simple datasets and statistical models. Additionally, the project will explore the concept of model switching, which allows for dynamic transitions between different statistical distributions when discrete changes are detected. The aim is to implement this functionality in Stan such that it addresses these needs effectively.

Chapter 2: **Introduction**

The project is worthwhile as it focuses on finding a solution for a problem with Bayesian statistical tools. That problem is handling sudden and discrete changes in the model parameters. This has real world applications like virus mutations. The ability to accurately model virus mutations will make a significant difference.

The change points that will be focused on in this project are of much importance. This is where one distribution starts and the other distribution ends. It could be the case that the two distributions overlap. The distributions might be of different types like normal, exponential, gamma etc. To give real world examples this could be when a virus mutates (like COVID-19) or when stock prices suddenly drop (2008 financial crisis). The aim is to implement functionality allowing for change points as a variable(s) that will be solved by the program. This is because it is not always known where these change points are in a time series dataset.

Building methods for change-point inference and model switching enables more accurate predictions and parameter estimates. Being able to create models that quickly adapt to new data are more useful in fields like epidemiology. If it is possible to detect a virus mutation early adaptations can be made to make informed decisions regarding policies for safety.

This functionality is not restricted to healthcare. The same techniques can be applied whenever something causes an abrupt change that, when recognised, is understandable. So there are applications in finance, climate science and quality control in manufacturing.

This project aims to empower data scientists and statisticians to tackle problems that involve sudden shifts more effectively.

Chapter 3: Related Work and Ideas

This project requires knowing how to use Stan. To learn Stan I went through the tutorials on the official YouTube channel. These tutorials are in a playlist containing eight videos starting at setting up the development environment and all the way to hierarchical modelling and Gaussian processes.

These are the main things I learned from each tutorial is explained in this section. In the Split testing Tutorial I learned how to write a .stan file, including its structure with data, parameters, and model blocks. I fitted a model to the data, which initially suggested that the click-through rate (CTR) on Twitter was higher than on Facebook. I also learned how to plot distributions using Rstan, observing that the small dataset introduced a lot of uncertainty, evident in the wide distribution. However when the dataset was increased, it became clear that Facebook's CTR was actually higher than Twitter's. The initial observation was due to chance, which highlightes the importance of having a sufficient sample size to achieve statistical significance.

In the Linear Regression Tutorial I used the iris (a type of flower) dataset for this. The goal here was to quantify a linear relationship between the petal length and the sepal length for the iris flower to determine how the sepal length varies as the petal length varies. The sepal length is what I wanted to determine the value of. The petal length is known. Prior knowledge of the relationship between the sepal and petal was used to help Stan fit the data. So this involved choosing a distribution mean and standard deviation. I was adjusting the values of the mean and standard deviation to get a more accurate result. The n-eff and r-hat can be used to see if the model fits the data well. An r-hat value of close to one is ideal and indicates that the chains have mixed well. N-eff is the effective sample size and its better to have a higher number for this variable. I also learned that simulated data generation is used to check the models accuracy. This is done in a new block, generated quantities, within the Stan file.

In the Convergence checks in Stan tutorial. The Central limit theorem was explained to an introductory level. It states that the sampling distribution of the mean approaches normality as the sample size grows, regardless of the population's shape. The chance a coin lands on heads is 0.5. The same is true for tails. However if you check the mean of 10 coin tosses using 1 and 0 to refer to heads and tails respectively, you might get 0.7. If you keep increasing the sample size using these values: 50, 100, 500, 1000. You will notice the mean will keep getting closer to 0.5. This is because the extremes get smoothed out. Moving on to the convergence of chains. Markov Chain Monte Carlo also know as MCMC is the sampler used to approximate complex probability distributions, particularly in Bayesian statistics . Chains are constructed to explore the target probability distribution which in our case is posterior distribution in Bayesian analysis.

To check convergence one needs to run multiple chains from different starting points and see if they converge. Generally we don't consider the first few iterations of the sample. The sample path of chains can be visually inspected using trace plots. When we increase the number of iterations the mixing will improve but will also take longer. We can use the the scalar value r-hat to see if the chains have converged. If the R-hat is close to one that's a good sign. There are multiple types of r-hat, the one Stan uses by default is split r-hat statistic.

There are other values we can check as well like bulk ess, tail ess, n eff etc. MCMC is employed to sample from the posterior distribution. If the chain is not geometrically ergodic(chain eventually reaches every part of the probability distribution by running long enough), convergence diagnostics like the effective sample size (ESS) and R-hat statistics might fail to detect convergence issues.

Without geometric ergodicity, chains may have heavy tails or be overly influenced by starting values, leading to unreliable parameter estimates. Poor mixing of the chain and/or slow movement between modes can be problematic. Reparametrisation by changing the priors can be used to fix this. If divergences occur then by increasing adapt delta the sampler becomes more cautious by reducing the step size which in turn improves numerical stability at the cost of longer runtime.

The Modelling heteroscedasticity tutorial mentions heteroscedasticity. Heteroscedasticity refers to non constant variance in a dataset. So the scale of the noise terms vary over time. This has applications in finance as the stock prices' volatility changes. This can be used in risk management. ARCH models are used when the dataset is heteroscedastic. ARCH stands for AutoRegressive Conditional heteroscedasticity. This type of model has different versions. They take into account the previous state(s). This ARCH model is not great at adjusting to small changes in the variance so the fluctuations happen when there are big changes. Now moving onto another model that is better at handling small changes in the variance. This type of model is known as the GARCH model. This stands for Generalised AutoRegressive Conditional heteroscedasticity. This type of model takes into account two things, the previous values and the previous volatility values.

In the Hierarchical modelling tutorial we take a look at discrete heterogeneity. This is a measure of variance of categorical data in a dataset. I used a premierleague.csv file dataset from here. The number of goals here is a discrete probability distribution so I used the Poisson distribution to represent this. I left some of the data points out of the analysis to allow for checking the predictive accuracy. In hierarchical modelling we often use hyperpriors instead. So hyperpriors are used on hyperparameters. Hyperparameters control priors on other parameters. A hyperprior is a prior placed on the hyperparameter.

There are many benefits to using hierarchical modelling. Hierarchical modelling allows for partial pooling which allows information to be shared between groups, balancing groups and global trends. This can reduce over-fitting and improve predictability. Using these techniques I was able to predict the winners of the premier league. The prediction was quite accurate as four out of the five top teams estimated were correct.

In the Gaussian processes tutorial it is mentioned that Gaussian Processes are used to determine which function best fits a given dataset, since there are infinitely many possible functions. A Gaussian Process assigns a probability to each function, effectively treating functions as random variables. The mean of this probability distribution over functions represents the most likely (or best-fit) function. As described in the literature, "A Gaussian Process is a prior over functions P(f)P(f) that can be used for Bayesian regression" [1].

There are many kernels that can be chosen for regression. There are two main types of kernels. Stationary kernels and non-stationary kernels. Stationary kernels like the Radial Bias Function (RBF) kernel focus on relationships between points not taking into consideration their location. In contrast with the non-stationary kernels that consider absolute positions. One should choose an appropriate kernel for the best results. The alpha and rho variables are significant. "Alpha measures the average distance from the mean function. Rho values is the frequency of the function represent by the GP[**definition2**]". In this tutorial I used an astronomical dataset. The goal was to identify exoplanets. Exoplanets are planets that orbit stars outside our solar system. A bit of data cleaning was done on the dataset. There are two types of Gaussian processes covered in this video. The first being Latent Variable Gaussian Process. Latent Variable Gaussian Processes extend Gaussian Processes by introducing latent variables to model complex and high-dimensional data. Since we only care about the parameters of the Gaussian Process we can use another Gaussian Process. That's called the Marginal Likelihood Gaussian Process. I used a non central for efficiency.

Chapter 4: Data Considerations

The datasets were created from scratch to test the solution's ability to figure out where the change point is occurring. The data was designed as a time series dataset. It contains numerical values that simulate specific scenarios. More details about these scenarios are in the outline to approach section. These scenarios have chosen change points allowing for accurate and effective testing.

The data was structured and stored in a static format and saved locally. Since the datasets were generated for this project, they are free from noise and irrelevant columns. This means there was no need for data cleaning. A dataset or multiple datasets were created for each stage of development as described in the outline of approach section. It is better in this project to create synthetic data than to use real world data. This is because in real world data the change points are unknown so it is not possible to use it for testing. The datasets that were used are two hundred and fifty day datasets.

There are no licensing issues or restrictions associated with these datasets as the datasets are synthetic and generated for this project. Similarly, there are no ethical or privacy concerns as the data does not involve any personal or sensitive information. After the project is complete the datasets will be made publicly available. This could help others who are interested in testing similar models or conducting related research.

Chapter 5: Outline of Approach

Currently in Stan model switching at change points relies on the user having to hard code the values for it. This means the user has to already have knowledge of where the change point is. This greatly limits its practicality in real world scenarios. This is due to the fact that usually change-points are unknown and unpredictable. Like the spread of viruses and financial market shifts. To solve this problem I aim to implement the functionality in Stan to identify change points automatically thus being able to switch models making it a better fit of the data and without the user knowing where the change points are. The solution has four different stages of development. Each stage handles another layer of complexity. The synthetic data is used to check the accuracy of the solution at each stage as it has a known change point. Four different test suites/datasets will be created to evaluate each stage.

Stage One is the simplest case. This is where there will be two of the same type of distribution with no overlap. Then a brute-force approach will be used that will try every possible point as a change point and then use the one that is closest to/the same as the actual change point. This brute-force approach ensures all options are explored and a solution is found. The synthetic data is used to check the accuracy of the solution as it has a known change point.

Stage Two introduces overlap between distributions. This stage allows for transitions. This is similar to the real world where distributions often overlap. The two overlapping distribution will still be of the same type (like a normal distribution for example).

Stage Three involves working with two different distributions. This means that at the change point the model will switch to a different distribution. There are many combinations of distributions that can be tested.

Stage Four focuses on improving efficiency by using a search method instead of a brute-force method. This, if successfully done will greatly reduce the time taken for the model to fit the data. This optimisation is necessary if I want to work with larger datasets in the future. If the accuracy of the search method is close to the brute-force approach then the search method's improvement in terms of speed will make it more useful.

The synthetic data is used to check the accuracy of the solution at each stage as it has a known change point. The only limiting factor here is that another synthetic dataset will need to be introduced for each scenario that I test. The accuracy will be calculated by checking the difference between the actual and predicted change point.

Chapter 6: **Project Workplan**

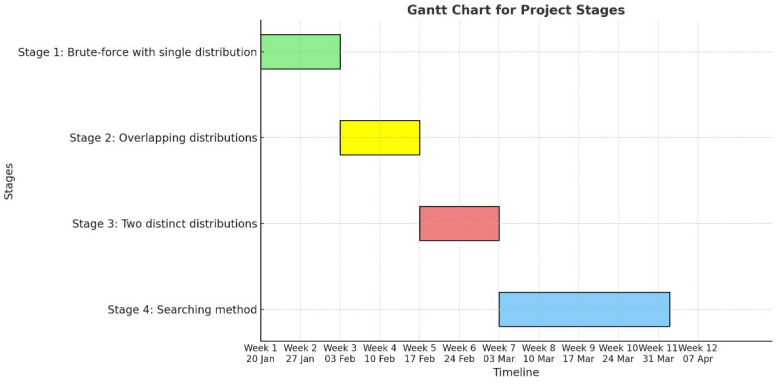

Figure 6.1: 1

Each stage mentioned in the outline of approach and in the gantt chart has two stages. The first is to prepare datasets that will act as a test suites. This is because the accuracy of each stage can be checked. The method in which the success of the project is evaluated is by using the test suites. The closer the implementations are of each stage in determining the change points the better. Each stage will involve one or more datasets. The stages are described in more detail in the outline to approach section.

Chapter 7: Progress

7.1 Dataset Generation

The dataset generation process is an important part of this project. To be able to test the predicated value of the change point tau in a dataset the value of the change point tau must be known. This is so they can be compared. In a real life dataset set the value of tau is unknown so the best way to approach this problem is to use a generated dataset. This is because the value of tau can be chosen and checked to see if the solution returns the same value or a similar value.

The stage one dataset simulates a time-series dataset representing the spread of two outbreaks: an initial variant and a second variant emerging later on. The dataset is generated using the python libraries NumPy, Matplotlib, and Pandas with no overlap between the distributions and a known change point tau. The simulation spans two hundred days with a change point (tau is equal to one hundred) where the new variant starts. The Mean is equal to fifty and standard deviation is equal to ten. It is declared explicitly for both distributions in the python script.

The change point tau is declared explicitly when generating the data as well and is what the solution should figure out. The first variant's cases are computed using a Gaussian function and turn into a flat line before tau. The second variant remains zero until t-switch and then follows a Gaussian curve. Total cases are the sum of both variants.

The formula that is used is the following.

```
cases-variant1 = amp1 * np.exp(-((days - mu1) ** 2) / (2 * sigma1 ** 2))
```

It is the normal distribution formula that is used for modelling (omitting the first fraction). The reason for the first fraction being omitted is because probabilities are not being considered and the purpose of that fraction is to ensure the area under the curve adds up to one. The dataset is structured into a data frame with columns: Day, Total-Cases, Cases-Variant1, and Cases-Variant2, and saved as datasetStageOne.csv to be used to test the solution.

The stage two and stage three datasets are generated in a similar manner. The stage two dataset will involve two overlapping normal distributions. The stage three dataset uses a gamma distribution for the second distribution and a normal distribution for the first distribution.

7.2 Explanation of Results

In each stage there are multiple files. There is a Stan file, an R script and a python script that is used to generate the dataset. A Stan model is written using a .stan file. Each .stan file contains multiple blocks like the data block, parameters block, the model block and potentially a transformed parameters block. The data block is used to declare the input data which in this case is the time-series dataset with the two COVID variants columns, the day column and the total number of cases column.

Th parameters block within the Stan file defines unknown variables that the model will estimate such as the mean and standard deviation of distributions as well as the change point which is the main focus in this project. The model block specifies how the data and parameters are related using probability distributions like normal and log normal distributions. It uses priors to achieve this.

In Bayesian modelling with Stan the priors represent initial beliefs about parameters before observing any data. In the change-point inference models in this project, the parameter of interest is the change point tau. Due to the absence of strong prior information about the value of tau in a dataset with an unknown change point tau, an uninformed prior approach has been used. This enables the Stan models developed in this project to generalise effectively to unseen datasets. The priors are centred around the midpoint of the dataset, reflecting a neutral assumption that the change is most likely to occur near the middle rather than at either extreme. In a fifty day dataset this would be twenty five and in a two hundred day dataset this would be one hundred. A normal distribution centred at the midpoint with a relatively large standard deviation was used, allowing the observed data to guide the final inference.

Using a midpoint-centred prior with a wide spread ensures that the prior remains weakly informative. This setup provides a reasonable default while allowing the data to dominate the posterior distribution. The bigger the variance the more uninformed the prior. This results in the observed data guiding the estimation for tau. When these Stan models are being used by a user with a random dataset, the user should adjust the priors in accordance with what is expected and what is known about the dataset. For example if it is known that the number of cases at the peak will be in the hundreds of thousands then the priors should be changed to reflect this.

When stan samples from a posterior distribution (like the priors) it uses Markov Chain Monte Carlo chains. These chains are run independent of each other and are used to explore the parameter space. To give an example it will start with an initial guess that the mean should be twenty two and then it will check how well it explains the observed data. Then using the hamiltonian monte carlo algorithm it will propose a new value which will be accepted or rejected depending on how much it improves the posterior probability. Convergence of all chains provides confidence in the accuracy of the prior variable's estimation. If the chains diverge this means that each chain is guessing a different value for the best fit data and is usually a sign that the priors are not correct.

The quality of the fit can be evaluated using the effective sample size (n-eff) and the r-hat diagnostic. A high n-eff value suggests that the samples provide useful information whereas a low n-eff suggests that the chains are highly correlated. This indicates that each new sample contributes less information and thus reduces the reliability of the inference. The r-hat values indicate how well the chains have converged. if the r hat is close to 1.0 then the chains have mixed well. If the r hat is greater than 1.1 this suggests that the chains are not converging well.

Stan requires using another language as an interface like python and R. Stan cannot be used by itself. R and the Rstan package were used to compile and run Stan models. The R scripts start by importing the necessary libraries. Two statements are used to optimize the code: one enables parallelism by utilising all CPU cores, and the other saves the compiled model. Then the dataset is read and the model is fit to the data specifying the number of chains and iterations. Then the results are printed out including the change point tau's value.

7.3 Stage One Solution

For stage one a smaller dataset was used for testing as the main dataset (datasetStageOne.csv) takes long to run due it being a bigger dataset. The smaller dataset is fifty days long so a quarter the size of datasetStageOne.csv. This smaller dataset is a lot quicker to run as it takes approximately fifteen minutes for Stan to fit the model whereas on the one hundred day dataset it took Stan around twenty four hours. To estimate the change point tau a Stan model stageOneSolution.stan was implemented. The model takes into account the two Gaussian distributions. In the parameter block there are two amplitudes, means and spreads for the two distributions and the variable that will be determined which is the change point tau.

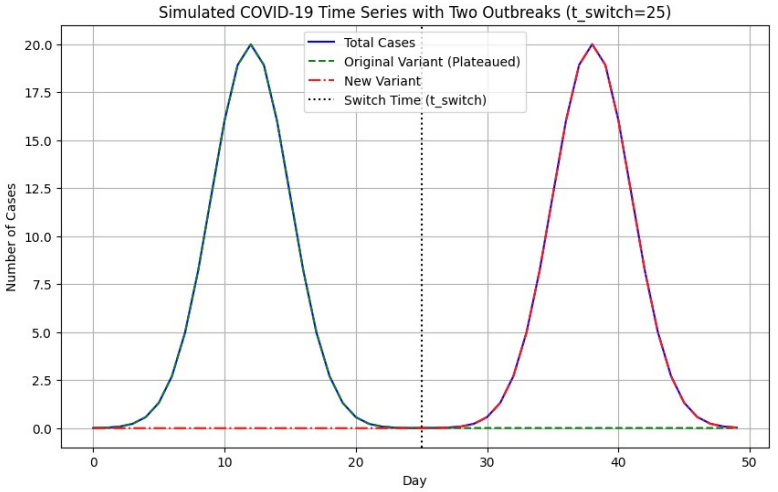

Figure 7.1: 1

There is a transformed parameter block in the Stan model. The transformed block is used to define quantities that depend on parameters and other known values. This acts as a bridge between the model's parameters and the observed data. A sigmoid function (init-logit) was used to ensure a smooth transition from the first distribution to the second. The co-efficient, which in this case is five, is used to control how sharp the transition is. There is a mu-cases array defined in the transformed parameters block that combines the outbreak waves to create a single expected case count for each day. This transformation helps structure the model and makes sampling more efficient.

The model block in the Stan model contains the priors. Since the data is synthetic and has known variables it is not difficult to incorporate this knowledge into the model. The amplitude values for both distributions specified that the distributions are normal and centred at twenty with a standard deviation of five. The mu values are sampled from a normal distribution with specified means and standard deviations as you can see in the following code snippet.

```
model {
```

```
    // Priors
    amp1 ~ normal(20, 5);  // Amplitudes
    amp2 ~ normal(20, 5);
    mu1 ~ normal(N/3, N/6);
    mu2 ~ normal(2*N/3, N/6);
    sigma1 ~ lognormal(0, 0.5);
    sigma2 ~ lognormal(0, 0.5);
    tau ~ normal(N/2, N/6);
    sigma_obs ~ lognormal(0, 0.5);

    // Likelihood
    for (n in 1:N) {
        Total_Cases[n] ~ normal(mu_cases[n], sigma_obs);
    }

}
```

So all the variables in the parameters block specified here are incorporating prior knowledge and are explicitly mentioning the distributions that they are being sampled from as well as the mean and standard deviations of these distributions.

An R script was used to fit the Stan model to the data. The R script read the dataset, imported the relevant libraries, fit the model and printed out the values of tau and other variables. The fit command took approximately fifteen minutes to run which is quick enough for debugging purposes.

```
fit <- stan(
  file = "stageOneSolution.stan",
  data = stan_data,
  iter = 12000, chains = 4, warmup = 4000,
  control = list(adapt_delta = 0.9999, max_treedepth = 25),
  seed = 42
)
```

```
print(fit, pars = c("amp1", "amp2", "mu1", "mu2", "sigma1", "sigma2", "tau", "sigma_obs"))
```

The result of the print command is displayed in the following figure.

```
Inference for Stan model: anon_model.
4 chains, each with iter=12000; warmup=4000; thin=1;
post-warmup draws per chain=8000, total post-warmup draws=32000.

           mean se_mean  sd 2.5%   25%  50%   75% 97.5% n_eff Rhat
amp1      20.00    0.00 0.0 20.0 20.00 20.0 20.00 20.00 29060    1
amp2      20.00    0.00 0.0 20.0 20.00 20.0 20.01 20.01 27494    1
mu1       13.00    0.00 0.0 13.0 13.00 13.0 13.00 13.00 37869    1
mu2       39.00    0.00 0.0 39.0 39.00 39.0 39.00 39.00 39634    1
sigma1     3.00    0.00 0.0  3.0  3.00  3.0  3.00  3.00 27542    1
sigma2     3.00    0.00 0.0  3.0  3.00  3.0  3.00  3.00 27303    1
tau       19.43    0.11 5.2  6.7 16.35 20.4 23.54 26.35  2184    1
sigma_obs  0.00    0.00 0.0  0.0  0.00  0.0  0.00  0.01  3616    1

Samples were drawn using NUTS(diag_e) at Tue Mar 11 20:17:33 2025.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

Figure 7.2: 1

The change point was at day twenty five. From day twenty two to day twenty eight the first distribution has come to an end and the second distribution has not yet started. When the Stan model was fit to this dataset to predict the change point tau the resulting output was 19.43 . This is because the Stan model is eager to find the change point. In the next stages or in future research this can be explored by having an offset value or using a technique when there is a large range where both distributions have zero cases.

7.4 Adjusting the Prior for the Change Point

This section investigates how different prior expectations on the change point day tau influences the inferred dynamics of the two outbreaks in the model. Specifically the mean of the prior distribution for tau is altered while keeping the standard deviation constant (tau normal(x, 8)).

The same dataset as stage one was used with the same associated Stan file and R script. The only difference was the tau prior's mean value in the model section of the Stan file. The change point tau in this dataset is twenty five but from twenty two to twenty eight the line is flat between the two distributions.

```
tau ~ normal(8, 8);
```

```
> fit <- stan(
+   file = "stageOneSolution.stan",
+   data = stan_data,
+   iter = 12000, chains = 4, warmup = 4000,
+   control = list(adapt_delta = 0.9999, max_treedepth = 25),
+   seed = 42
+ )
hash mismatch so recompiling; make sure Stan code ends with a blank line
Warning messages:
1: The largest R-hat is 1.53, indicating chains have not mixed.
Running the chains for more iterations may help. See
https://mc-stan.org/misc/warnings.html#r-hat
2: Bulk Effective Samples Size (ESS) is too low, indicating posterior means and medians may be unreliable.
Running the chains for more iterations may help. See
https://mc-stan.org/misc/warnings.html#bulk-ess
3: Tail Effective Samples Size (ESS) is too low, indicating posterior variances and tail quantiles may be unreliable.
Running the chains for more iterations may help. See
https://mc-stan.org/misc/warnings.html#tail-ess
>
```

Figure 7.3: 1

```
> print(fit, pars = c("amp1", "amp2", "mu1", "mu2", "sigma1", "sigma2", "tau", "sigma_obs"))
Inference for Stan model: anon_model.
4 chains, each with iter=12000; warmup=4000; thin=1;
post-warmup draws per chain=8000, total post-warmup draws=32000.

           mean se_mean    sd  2.5%   25%   50%   75% 97.5% n_eff     Rhat
amp1      20.00    0.00  0.00 20.00 20.00 20.00 20.00 20.01     4     1.38
amp2      20.00    0.00  0.00 20.00 20.00 20.00 20.01 20.01     4     1.37
mu1       19.50    7.96 11.26 13.00 13.00 13.00 19.50 39.00     2 30002.74
mu2       32.50    7.96 11.26 13.00 32.50 39.00 39.00 39.00     2 29964.83
sigma1     3.00    0.00  0.00  3.00  3.00  3.00  3.00  3.00     3     1.90
sigma2     3.00    0.00  0.00  3.00  3.00  3.00  3.00  3.00     3     1.91
tau        8.12    2.92  6.36  1.04  1.69  7.17 12.61 22.40     5     1.31
sigma_obs  0.00    0.00  0.00  0.00  0.00  0.00  0.00  0.01  3800     1.01

Samples were drawn using NUTS(diag_e) at Tue Apr  8 20:41:53 2025.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

Figure 7.4: 1

```
tau ~ normal(16, 8);
```

```
> print(fit, pars = c("amp1", "amp2", "mu1", "mu2", "sigma1", "sigma2", "tau", "sigma_obs"))
Inference for Stan model: anon_model.
4 chains, each with iter=12000; warmup=4000; thin=1;
post-warmup draws per chain=8000, total post-warmup draws=32000.

           mean se_mean  sd 2.5%   25%  50%   75% 97.5% n_eff Rhat
amp1      20.00    0.00 0.0 20.0 20.00 20.0 20.00 20.00 29060    1
amp2      20.00    0.00 0.0 20.0 20.00 20.0 20.01 20.01 27494    1
mu1       13.00    0.00 0.0 13.0 13.00 13.0 13.00 13.00 37869    1
mu2       39.00    0.00 0.0 39.0 39.00 39.0 39.00 39.00 39634    1
sigma1     3.00    0.00 0.0  3.0  3.00  3.0  3.00  3.00 27542    1
sigma2     3.00    0.00 0.0  3.0  3.00  3.0  3.00  3.00 27303    1
tau       19.43    0.11 5.2  6.7 16.35 20.4 23.54 26.35  2184    1
sigma_obs  0.00    0.00 0.0  0.0  0.00  0.0  0.00  0.01  3616    1

Samples were drawn using NUTS(diag_e) at Wed Mar 26 10:09:57 2025.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

Figure 7.5: 1

```
tau ~ normal(25, 8);
```

```
> print(fit, pars = c("amp1", "amp2", "mu1", "mu2", "sigma1", "sigma2", "tau", "sigma_obs"))
Inference for Stan model: anon_model.
4 chains, each with iter=12000; warmup=4000; thin=1;
post-warmup draws per chain=8000, total post-warmup draws=32000.

           mean se_mean  sd 2.5%   25%  50%   75% 97.5% n_eff Rhat
amp1      20.00    0.00 0.0 20.0 20.00 20.0 20.00 20.00 29060    1
amp2      20.00    0.00 0.0 20.0 20.00 20.0 20.01 20.01 27494    1
mu1       13.00    0.00 0.0 13.0 13.00 13.0 13.00 13.00 37869    1
mu2       39.00    0.00 0.0 39.0 39.00 39.0 39.00 39.00 39634    1
sigma1     3.00    0.00 0.0  3.0  3.00  3.0  3.00  3.00 27542    1
sigma2     3.00    0.00 0.0  3.0  3.00  3.0  3.00  3.00 27303    1
tau       19.43    0.11 5.2  6.7 16.35 20.4 23.54 26.35  2184    1
sigma_obs  0.00    0.00 0.0  0.0  0.00  0.0  0.00  0.01  3616    1

Samples were drawn using NUTS(diag_e) at Wed Mar 26 10:09:57 2025.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

Figure 7.6: 1

```
tau ~ normal(32, 8);
```

```
> print(fit, pars = c("amp1", "amp2", "mu1", "mu2", "sigma1", "sigma2", "tau", "sigma_obs"))
Inference for Stan model: anon_model.
4 chains, each with iter=12000; warmup=4000; thin=1;
post-warmup draws per chain=8000, total post-warmup draws=32000.

           mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
amp1      20.00    0.00 0.00 20.00 20.00 20.00 20.00 20.00 29083    1
amp2      20.00    0.00 0.00 20.00 20.00 20.00 20.01 20.01 28530    1
mu1       13.00    0.00 0.00 13.00 13.00 13.00 13.00 13.00 39295    1
mu2       39.00    0.00 0.00 39.00 39.00 39.00 39.00 39.00 38384    1
sigma1     3.00    0.00 0.00  3.00  3.00  3.00  3.00  3.00 27500    1
sigma2     3.00    0.00 0.00  3.00  3.00  3.00  3.00  3.00 26727    1
tau       21.68    0.08 3.96 11.74 19.61 22.66 24.69 26.56  2355    1
sigma_obs  0.00    0.00 0.00  0.00  0.00  0.00  0.00  0.01  4151    1

Samples were drawn using NUTS(diag_e) at Wed Mar 26 10:35:11 2025.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
>
```

Figure 7.7: 1

```
tau ~ normal(40, 8);
```

```
hash mismatch so recompiling; make sure Stan code ends with a blank line
> print(fit, pars = c("amp1", "amp2", "mu1", "mu2", "sigma1", "sigma2", "tau", "sigma_obs"))
Inference for Stan model: anon_model.
4 chains, each with iter=12000; warmup=4000; thin=1;
post-warmup draws per chain=8000, total post-warmup draws=32000.

           mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
amp1      20.00    0.00 0.00 20.00 20.00 20.00 20.00 20.00 28201    1
amp2      20.00    0.00 0.00 20.00 20.00 20.00 20.01 20.01 27400    1
mu1       13.00    0.00 0.00 13.00 13.00 13.00 13.00 13.00 38224    1
mu2       39.00    0.00 0.00 39.00 39.00 39.00 39.00 39.00 40719    1
sigma1     3.00    0.00 0.00  3.00  3.00  3.00  3.00  3.00 27099    1
sigma2     3.00    0.00 0.00  3.00  3.00  3.00  3.00  3.00 26490    1
tau       23.12    0.06 3.07 15.21 21.69 23.92 25.38 26.70  2505    1
sigma_obs  0.00    0.00 0.00  0.00  0.00  0.00  0.00  0.01  3843    1

Samples were drawn using NUTS(diag_e) at Wed Mar 26 10:39:49 2025.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

Figure 7.8: 1

In this experiment it can be seen that the greater the mean value chosen in the prior the more accurate the result is for tau. This is due to the model having more information. The rest of the variables in the output have the same value as their priors have been kept consistent. In the beginning when the mean of tau is eight the R-hat's value is 1.53. The error message occurs as the model does not have access to much information. This leads to a very inaccurate tau estimation of 8.12. Besides this first attempt with a mean of 8 all the following attempts return good results. The most accurate being 23.12 which is the output of the Stan model when the mean of tau which follows a normal distribution is fourty. An interesting thing to investigate here is why is it the case that when the mean of the prior of tau is fourty the result is more accurate than when the mean of the prior of tau is twenty five. Possibly it could be the Stan model is a bit eager so it estimates the value for tau a bit early and the late prior is delaying it a bit resulting in a more accurate estimation.

7.5 Stage One with a Bigger Dataset

For this stage a larger dataset was used to test how the model performs when provided with more data points. This dataset is four times larger than the one used in Stage One (datasetStageOne.csv) and includes two hundred days of data. The dataset simulates two COVID-19 outbreaks with a defined change point at day one hundred. As the dataset is significantly larger the Stan model takes longer to run.

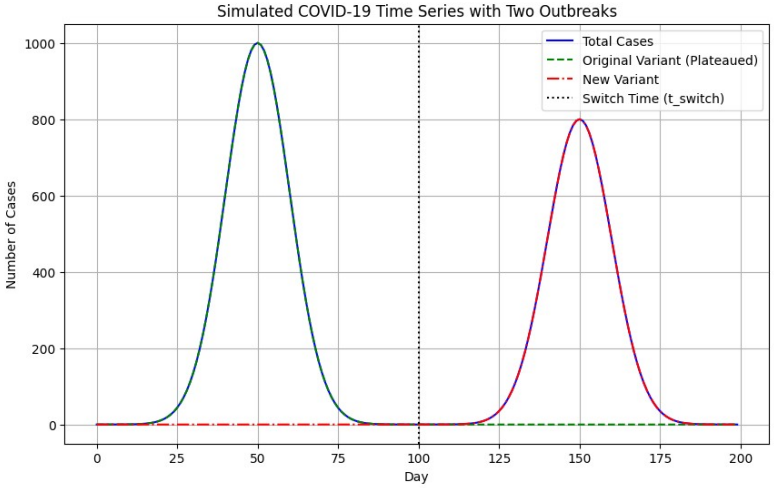

Figure 7.9: 1

To estimate the change point tau, the same Stan model was reused (with an updated dataset and data block) used in Stage One, making no significant changes to the model structure. The priors in the model block were changed to reflect the new dataset. The model incorporates two Gaussian-shaped outbreaks with separate amplitudes, means, and standard deviations, and includes a sigmoid function for a smooth transition between the two distributions. This allows the model to estimate the change point tau accurately based on the observed data.

In the parameter block, the amplitude, mean, and standard deviation priors remain consistent in terms of being present with previous sections. As the dataset is synthetic and fully known, prior knowledge of the dataset was embedded into the priors directly. Below is a snippet of the updated R script that was used to run the model:

```
> print(fit, pars = c("amp1", "amp2", "mu1", "mu2", "sigma1", "sigma2", "tau", "sigma_obs"))
Inference for Stan model: anon_model.
4 chains, each with iter=12000; warmup=4000; thin=1;
post-warmup draws per chain=8000, total post-warmup draws=32000.

             mean se_mean   sd    2.5%     25%     50%     75%   97.5% n_eff Rhat
amp1      1000.00    0.00 0.00 1000.00 1000.00 1000.00 1000.00 1000.00 28490    1
amp2       800.00    0.00 0.00  800.00  800.00  800.00  800.00  800.01 26103    1
mu1         51.00    0.00 0.00   51.00   51.00   51.00   51.00   51.00 35519    1
mu2        151.00    0.00 0.00  151.00  151.00  151.00  151.00  151.00 37358    1
sigma1      10.00    0.00 0.00   10.00   10.00   10.00   10.00   10.00 26023    1
sigma2      10.00    0.00 0.00   10.00   10.00   10.00   10.00   10.00 23478    1
tau         97.04    0.08 3.32   89.05   95.03   97.74   99.72  101.40  1581    1
sigma_obs    0.00    0.00 0.00    0.00    0.00    0.00    0.00    0.00  2477    1

Samples were drawn using NUTS(diag_e) at Wed Mar 19 13:08:57 2025.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
>
```

Figure 7.10: 1

The estimated change point tau returned by the model had a value of 97.04. This is very close to the true change point tau with a value of one hundred. The model was able to use the large amount of data to confidently estimate the change point with a narrow credible interval of [89.05, 101.40]. The standard deviation of tau was about 3.32, indicating low uncertainty. All parameters converged well with R-hat values of one and high effective sample sizes (n-eff), showing that the model was both stable and well-calibrated. This is proof that the Stan models used to detect change-points in other sections using fifty day datasets can also handle one hundred day datasets.

7.6 Stage Two Solution

For Stage Two a smaller dataset was used for testing, similar to Stage One. The main dataset takes longer to process so a scaled down version containing fifty days of data was used to make the computations faster. This smaller dataset allows for quicker debugging and model validation.

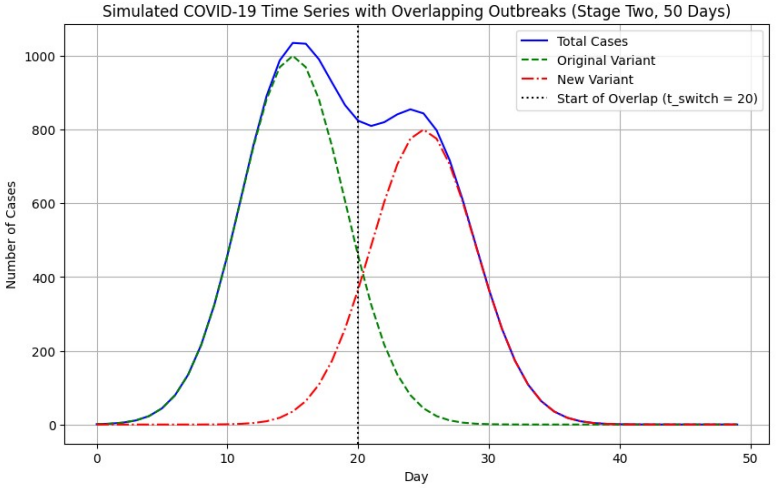

Figure 7.11: 1

The dataset simulates a COVID-19 outbreak with two overlapping variants. The original variant peaks at day fifteen with a spread of four days, while the new variant peaks at day twenty five with a spread of four days. The transition between outbreaks occurs at day twenty which is the change point in the dataset.

To model the outbreaks, two Gaussian distributions were used. The original variant follows a Gaussian curve with an amplitude of one thousand, and the new variant follows a Gaussian curve with an amplitude of eight hundred. The total cases per day are computed as the sum of both curves. The dataset was saved as datasetStageTwo-50days.csv .

To estimate the change point tau, I implemented a Stan model (stageTwoSolution.stan). The model differs to Stage One by accounting for overlapping outbreaks. It includes amplitudes (amp1, amp2) for the two waves, means (mu1, mu2) representing the peak days, spreads (sigma1, sigma2) controlling the distribution width, change point tau to estimate the transition between the two outbreaks, and observation noise (sigma-obs) to account for the data variability. This model looks quite similar to the model for stage one with the difference between them being the priors in the model block.

Since the dataset is synthetic like before, prior knowledge was easily incorporated into the model. The amplitudes follow a normal distribution centred at one hundred and eighty, while the mu values follow normal distributions centred at fifteen and thirty five. The sigma values use lognormal priors for positive constraints, and the tau (change point) follows a normal prior centred at day twenty five. The observation noise also follows a lognormal prior.

The model block specifies:

```
model {
    // Priors
    amp1 ~ normal(100, 20);  // Prior for first wave amplitude
```

```
    amp2 ~ normal(80, 20);   // Prior for second wave amplitude
    mu1 ~ normal(15, 5);     // First wave expected around day 15
    mu2 ~ normal(35, 5);     // Second wave expected around day 35
    sigma1 ~ lognormal(0, 0.5);
    sigma2 ~ lognormal(0, 0.5);
    tau ~ normal(25, 5); // Change-point prior centered around day 25
    sigma_obs ~ lognormal(0, 0.5);

    // Likelihood
    for (n in 1:N) {
        Observed_Cases[n] ~ normal(mu_cases[n], sigma_obs);
    }
}
```

An R script was used to fit the Stan model to the dataset. The dataset is read from the CSV file, and the model is run using stan() with twelve thousand iterations and four chains. The posterior distribution of tau (change-point) is analysed, and the results are visualized using ggplot2.

The fitting process:

```
fit <- stan(
  file = "stageTwoSolution.stan",
  data = stan_data,
  iter = 12000, chains = 4, warmup = 4000,
  control = list(adapt_delta = 0.9999, max_treedepth = 25),
  seed = 42
)
```

```
print(fit, pars = c("amp1", "amp2", "mu1", "mu2", "sigma1", "sigma2", "tau", "sigma_obs"))
```

The result of the print command is displayed in the following figure.

```
Inference for Stan model: anon_model.
4 chains, each with iter=12000; warmup=4000; thin=1;
post-warmup draws per chain=8000, total post-warmup draws=32000.

            mean se_mean   sd  2.5%    25%    50%    75%  97.5% n_eff Rhat
amp1      101.83    0.01 1.99 97.91 100.51 101.82 103.14 105.76 21983    1
amp2       80.62    0.01 1.94 76.81  79.34  80.59  81.89  84.48 22603    1
mu1        14.90    0.00 0.11 14.68  14.82  14.90  14.97  15.12 25431    1
mu2        34.90    0.00 0.14 34.63  34.81  34.90  35.00  35.18 27369    1
sigma1      4.75    0.00 0.11  4.53   4.67   4.75   4.82   4.98 19696    1
sigma2      4.89    0.00 0.14  4.62   4.80   4.89   4.99   5.18 21472    1
tau        20.30    0.03 2.94 13.37  18.53  20.95  22.47  24.63 13194    1
sigma_obs   4.68    0.00 0.47  3.88   4.35   4.64   4.97   5.71 24300    1

Samples were drawn using NUTS(diag_e) at Tue Mar 11 21:42:52 2025.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

Figure 7.12: 1

The model successfully captured the change point tau at 20.3, closely aligning with the expected value of 20. The estimated tau is very close to the expected value. In this model the r-hat value for all parameters was one. This indicates proper convergence. An R-hat value close to one suggests that the Markov Chain Monte Carlo (MCMC) chains have mixed well. The output contains high effective sample sizes (n-eff) which is also a good sign.

7.7 Different Change Point

Another time-series dataset was created with a different change point to make sure the model was not always guessing that tau is near day twenty. Just like the previous dataset it was generated using a python script with specified attributes like change point and gaussian distribution etc. In this dataset the change point tau's value is thirty five. In the dataset there are two Gaussian distribution that are non-overlapping. The dataset is a fifty day dataset.

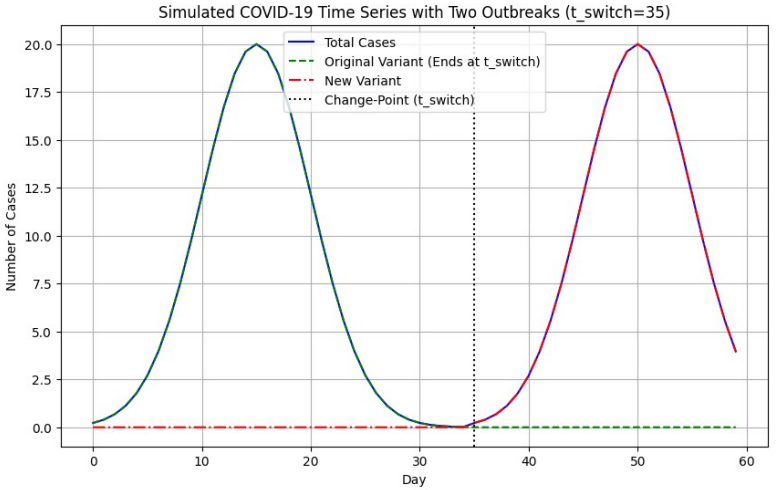

Figure 7.13: 1

The dataset simulates a COVID-19 outbreak with two not overlapping variants. The original variant peaks at day fifteen with a spread of five days, while the new variant peaks at day fifty with a spread of five days. The transition between outbreaks occurs at day thirty five which is the change point.

To model the outbreaks, two Gaussian distributions (normal distributions) were used. The original variant follows a Gaussian curve with an amplitude of twenty, and the new variant follows a Gaussian curve with an amplitude of twenty. The total cases per day are computed as the sum of both curves. The dataset was plotted and saved as t35.csv.

To estimate the change point tau, a Stan model (differentChangePoint.stan) was implemented.

The model includes amplitudes (amp1, amp2) for the two waves, means (mu1, mu2) representing the peak days, spreads (sigma1, sigma2) controlling the distribution width, change point (tau) to estimate the transition between the two outbreaks, and observation noise (sigma-obs) to account for data variability.

Since the dataset is synthetic like before, prior knowledge was easily incorporated into the model. Like the amplitudes which follow a normal distribution centred at twenty, while the mu values follow normal distributions centred at fifteen and fifty. The sigma values use lognormal priors for positive constraints, and the tau (change-point) follows a normal prior centred at day thirty five. The observation noise also follows a lognormal prior.

The model block specifies:

```
model {
    // Priors
    amp1 ~ normal(20, 5);
    amp2 ~ normal(20, 5);
    mu1 ~ normal(15, 5);
    mu2 ~ normal(50, 5);
    sigma1 ~ lognormal(0, 0.5);
    sigma2 ~ lognormal(0, 0.5);
    tau ~ normal(35, 5);
    sigma_obs ~ lognormal(0, 0.5);

    // Likelihood
    for (n in 1:N) {
        Total_Cases[n] ~ normal(mu_cases[n], sigma_obs);
    }
}
```

An R script was used to fit the Stan model to the dataset. The dataset is read from the CSV file, and the model is run using stan() with 12000 iterations and 4 chains. The posterior distribution of tau (change-point) is analysed, and the results are visualized using ggplot2.

The fitting process:

```
fit <- stan(
  file = "differentChangePoint.stan",
  data = stan_data,
  iter = 12000, chains = 4, warmup = 4000,
  control = list(adapt_delta = 0.9999, max_treedepth = 25),
  seed = 42
)
```

```
print(fit, pars = c("amp1", "amp2", "mu1", "mu2", "sigma1", "sigma2", "tau", "sigma_obs"))
```

The result of the print command is displayed in the following figure.

```
> print(fit, pars = c("amp1", "amp2", "mu1", "mu2", "sigma1", "sigma2", "tau", "sigma_obs"))
Inference for Stan model: anon_model.
4 chains, each with iter=12000; warmup=4000; thin=1;
post-warmup draws per chain=8000, total post-warmup draws=32000.

           mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
amp1      20.00       0 0.00 20.00 20.00 20.00 20.00 20.00 27392    1
amp2      20.00       0 0.00 20.00 20.00 20.00 20.00 20.00 27120    1
mu1       16.00       0 0.00 16.00 16.00 16.00 16.00 16.00 42037    1
mu2       51.00       0 0.00 51.00 51.00 51.00 51.00 51.00 36897    1
sigma1     5.00       0 0.00  5.00  5.00  5.00  5.00  5.00 26042    1
sigma2     5.00       0 0.00  5.00  5.00  5.00  5.00  5.00 25791    1
tau       35.46       0 0.02 35.42 35.45 35.47 35.48 35.51  5423    1
sigma_obs  0.00       0 0.00  0.00  0.00  0.00  0.00  0.00  3733    1

Samples were drawn using NUTS(diag_e) at Fri Mar 14 23:17:19 2025.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

Figure 7.14: 1

The model successfully captured the change point tau at 35.46, closely aligning with the expected value of 35. It has been confirmed that the Stan model is correctly estimating the value for the change point tau and not just guessing twenty. In this model the R-hat value for all parameters was one. This indicates proper convergence. An R-hat value close to one suggests that the Markov Chain Monte Carlo (MCMC) chains have mixed well. A high effective sample (n-eff) size which is the case here suggests low autocorrelation between samples which indicates that the posterior estimates are reliable and good chain convergence.

7.8 Stage Three Solution

In stage three there are two different distributions within the dataset unlike the previous two stages. A synthetic dataset of fifty days was used similar to before because its faster than using a two hundred day dataset. This is quite important for debugging. There is a great number of possible permutations when choosing two different distributions. In this stage a Gaussian distribution will be used which will be followed by a gamma distribution. The change point's value is equal to twenty five.

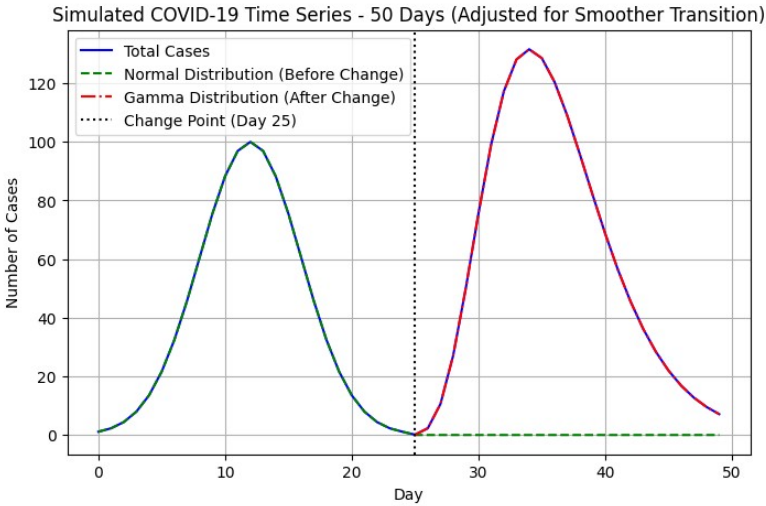

Figure 7.15: 1

A Stan model was implemented in a file called stageThreeSolution.stan. The parameters block includes amplitudes for both distributions, the means and standard deviations for the normal distribution, the shape and rate for the gamma distribution, as well as a noise parameter and the latent variable tau, which represents the day the change occurs. The model structure looks a bit different due to the presence of a gamma distribution.

```
transformed parameters {
    real tau = 1 + (N - 1) * inv_logit(tau_unconstrained); // Maps to [1, N]
    array[N] real mu_cases;
    for (n in 1:N) {
        real w = inv_logit(2 * (n - tau));  // Smoothed transition
        real variant1 = amp1 * exp(-((n - mu1)^2) / (2 * sigma1^2));
        real variant2 = amp2 * gamma_cdf(n | alpha, beta);
        mu_cases[n] = (1 - w) * variant1 + w * variant2;
    }
}

model {
    // Priors
    amp1 ~ normal(100, 20);
    amp2 ~ normal(1500, 5);
    mu1 ~ normal(10, 3);
    mu2 ~ normal(35, 5);
    sigma1 ~ lognormal(0, 0.5);
    sigma2 ~ lognormal(0, 0.5);

    tau_unconstrained ~ normal(-0.08, 1);  // tau = 25
```

```
    sigma_obs ~ lognormal(0, 0.5);
    alpha ~ gamma(6, 1);
    beta ~ gamma(2, 4);

    // Likelihood
    for (n in 1:N) {
        Total_Cases[n] ~ normal(mu_cases[n], sigma_obs);
    }
}
```

Tau is calculated through a transformation of an unconstrained parameter. This technique maps it to the range [1, N] using the inverse logit function, which helps ensure smooth sampling and better model convergence. This approach seemed to be better as the change point is close to sharp curves and this way there were no divergent transitions.

In the transformed parameters block the expected number of cases per day was defined as a weighted combination of the two distributions. A sigmoid function was used to create a smooth transition between the normal and gamma distributions. This is done through the expression inv-logit(2 * (n - tau)), which produces a smooth curve from 0 to 1 across the change point. The expected number of cases on each day mu-cases[n] is calculated by mixing the output from the normal distribution and the cumulative gamma distribution using this sigmoid weight. This approach avoids any abrupt change and helps the model blend the two components naturally.

A reparametrisation technique was used which is a technique in Stan. The tau-unconstrained is sampling from a normal distribution with the mean being -0.08. The -0.08 is related to the day of the change point which is 25. In this Stan model the parameter tau-unconstrained is transformed into a bounded variable tau that lies between 1 and N using a logistic mapping:

```
tau = 1 + (N - 1) * inv-logit(tau-unconstrained).
```

This ensures tau always stays within the valid range of days. The prior tau-unconstrained normal(-0.08, 1) is carefully chosen so that the transformed tau is centred around day 25. So in this case N = 50 so then inv-logit(-0.08) is approximately 0.48 which results in $tau \approx 1 + 49 * 0.48 \approx 24.5$. This shows that a value of -0.08 for tau-unconstrained produces a tau value very close to 25, aligning with the comment in the model. The use of an unconstrained parameter and logistic transform allows for more flexible sampling while still keeping tau within the desired range.

This method of reparameterisation was discovered using ChatGPT and it can be used for choosing the mean of a tau-unconstrained is shown in the following picture:

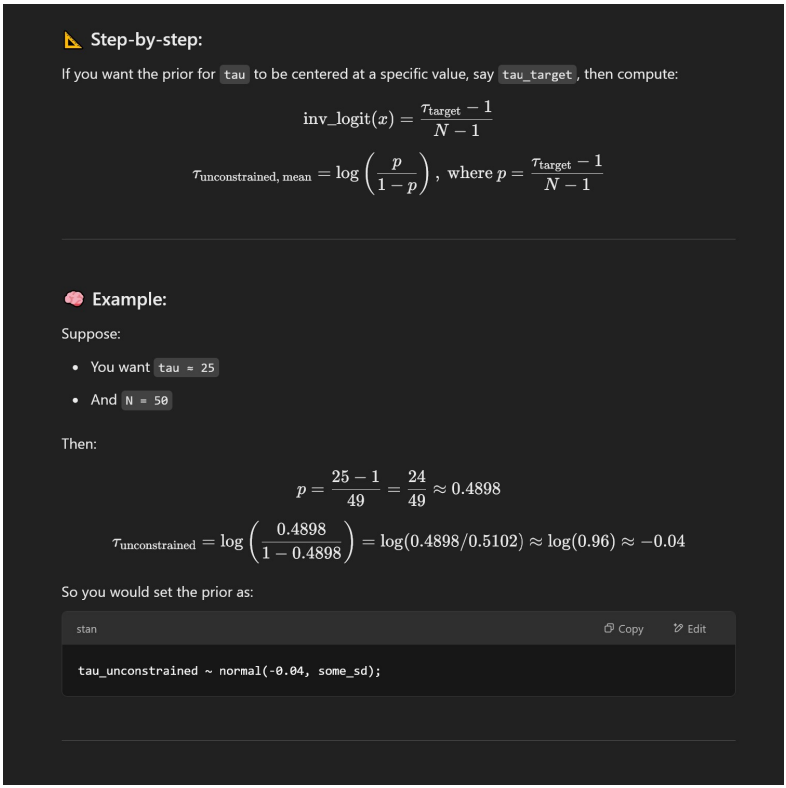

Figure 7.16: 1

This technique is also discussed in the documentation for Stan but the method to do it is not there.

```
> print(fit, pars = c("mu1", "sigma1", "alpha", "beta", "tau"))
Inference for Stan model: anon_model.
4 chains, each with iter=12000; warmup=4000; thin=1;
post-warmup draws per chain=8000, total post-warmup draws=32000.

        mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
mu1    12.84    0.00 0.69 11.46 12.40 12.84 13.29 14.20 25082    1
sigma1  3.45    0.00 0.62  2.28  3.03  3.42  3.83  4.74 20111    1
alpha   1.66    0.01 0.64  0.78  1.22  1.55  1.97  3.23 11107    1
beta    0.01    0.00 0.01  0.00  0.00  0.00  0.01  0.02 12335    1
tau    26.07    0.04 3.61 17.28 24.35 27.29 28.66 30.42  8059    1

Samples were drawn using NUTS(diag_e) at Fri Mar 28 00:58:14 2025.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

Figure 7.17: 1

The model successfully captured the change-point tau at 26.07, closely aligning with the expected value of 25. It has been confirmed the Stan model is correctly estimating the value for the change point tau. In this model the R-hat value for all parameters was one. This indicates proper convergence. An R-hat value close to one suggests that the Markov Chain Monte Carlo (MCMC) chains have mixed well. A high effective sample (n-eff) size which is the case here suggests low autocorrelation between samples which indicates that the posterior estimates are reliable and good chain convergence.

7.9 Flexible Prior with Variable Dataset

Three python scripts were used to create three datasets required for this section. Each dataset has a different change point tau. The first dataset's change point is at day seventy five. The second dataset's change point is at one hundred. The third dataset's change point is at one hundred and twenty five.

The prior here is flexible so it should be able to estimate the change points in all three different datasets. In the model the prior for tau only looks at values between fifty and one hundred and fifty. The assumption here is that the change point is not at the edges of the dataset which is what it usually expected. The Stan file is the same for all the data sets.

```
transformed parameters {
    array[N] real mu_cases;
    for (n in 1:N) {

        real w = inv_logit(0.5 * (n - tau));
        real variant1 = amp1 * exp(-((n - mu1)^2) / (2 * sigma1^2));
        real variant2 = amp2 * exp(-((n - mu2)^2) / (2 * sigma2^2)) * w;
        mu_cases[n] = variant1 + variant2;
    }
}
```

```
model {
    // Priors
    amp1 ~ normal(1000, 100);
    amp2 ~ normal(800, 100);
    mu1 ~ normal(50, 25);
    mu2 ~ normal(150, 25);
    sigma1 ~ lognormal(log(10), 0.5);
    sigma2 ~ lognormal(log(10), 0.5);


    tau ~ normal(100, 20) T[50, 150];

    sigma_obs ~ lognormal(0, 0.5);

    // Likelihood
    for (n in 1:N) {
        Total_Cases[n] ~ normal(mu_cases[n], sigma_obs);
    }
}
```

In the following is each dataset's graph and the predicted change point tau's value.

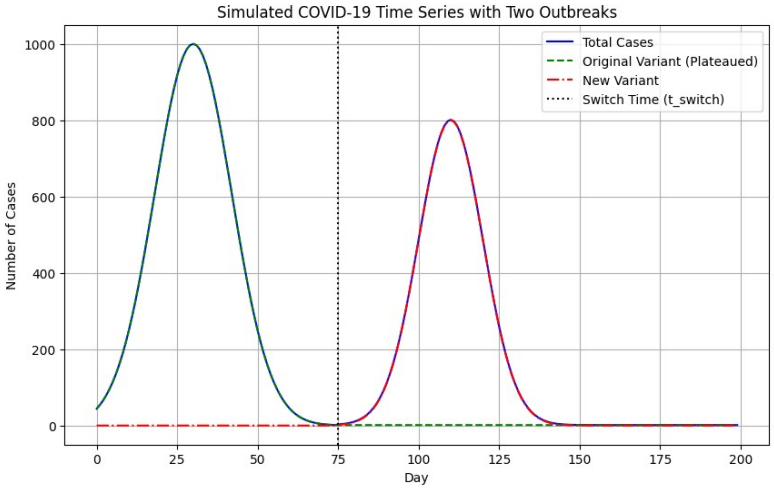

Figure 7.18: 1

```
> library(rstan)
> library(ggplot2)
> rstan_options(auto_write = TRUE)
> options(mc.cores = parallel::detectCores())
> df <- read.csv("dataset generation/multipleDatasets1.csv")  # Change the number here 1 2 3
> stan_data <- list(
+   N = nrow(df),
+   Total_Cases = df$Total_Cases
+ )
> fit <- stan(
+   file = "multipleChangePointsV2.stan",
+   data = stan_data,
+   iter = 12000, chains = 4, warmup = 4000,
+   control = list(adapt_delta = 0.9999, max_treedepth = 25),
+   seed = 42
+ )
Warning messages:
1: There were 1406 divergent transitions after warmup. See
https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
to find out why this is a problem and how to eliminate them.
2: Examine the pairs() plot to diagnose sampling problems

> print(fit, pars = c("amp1", "amp2", "mu1", "mu2", "sigma1", "sigma2", "tau", "sigma_obs"))
Inference for Stan model: anon_model.
4 chains, each with iter=12000; warmup=4000; thin=1;
post-warmup draws per chain=8000, total post-warmup draws=32000.

              mean se_mean   sd   2.5%    25%     50%     75%   97.5% n_eff Rhat
amp1       1000.00     0.0 0.20 999.62 999.87 1000.00 1000.13 1000.39 30497    1
amp2        800.86     0.0 0.21 800.45 800.72  800.86  801.00  801.29 25676    1
mu1          31.00     0.0 0.00  30.99  31.00   31.00   31.00   31.01 39016    1
mu2         111.00     0.0 0.00 110.99 111.00  111.00  111.00  111.01 34361    1
sigma1       12.00     0.0 0.00  11.99  12.00   12.00   12.00   12.01 29326    1
sigma2       10.02     0.0 0.00  10.01  10.02   10.02   10.02   10.03 24466    1
tau          65.06     0.1 5.74  52.03  61.27   66.37   69.70   72.79  3104    1
sigma_obs     0.73     0.0 0.04   0.66   0.71    0.73    0.76    0.81  5380    1

Samples were drawn using NUTS(diag_e) at Wed Apr  9 09:29:20 2025.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

Figure 7.19: 1

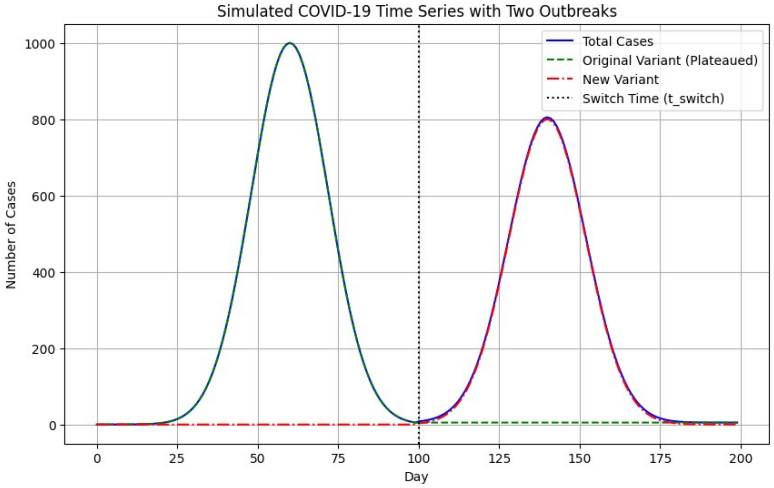
Simulated COVID-19 Time Series with Two Outbreaks
Total Cases
Original Variant (Plateaued)
New Variant
Switch Time (t_switch)
Number of Cases
Day
0
200
400
600
800
1000
0
25
50
75
100
125
150
175
200

Figure 7.20: 1

```
convergence, Rhat=1).
> library(rstan)
> library(ggplot2)
> rstan_options(auto_write = TRUE)
> options(mc.cores = parallel::detectCores())
> df <- read.csv("dataset generation/multipleDatasets2.csv")  # Change the number here 1 2 3
> stan_data <- list(
+   N = nrow(df),
+   Total_Cases = df$Total_Cases
+ )
> fit <- stan(
+   file = "multipleChangePointsV2.stan",
+   data = stan_data,
+   iter = 12000, chains = 4, warmup = 4000,
+   control = list(adapt_delta = 0.9999, max_treedepth = 25),
+   seed = 42
+ )
Warning messages:
1: There were 467 divergent transitions after warmup. See
https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
to find out why this is a problem and how to eliminate them.
2: Examine the pairs() plot to diagnose sampling problems

> print(fit, pars = c("amp1", "amp2", "mu1", "mu2", "sigma1", "sigma2", "tau", "sigma_obs"))
Inference for Stan model: anon_model.
4 chains, each with iter=12000; warmup=4000; thin=1;
post-warmup draws per chain=8000, total post-warmup draws=32000.

              mean se_mean    sd   2.5%    25%     50%     75%   97.5% n_eff Rhat
amp1      1000.02    0.00  0.56 998.93 999.65 1000.02 1000.40 1001.12 27424    1
amp2       803.66    0.00  0.56 802.56 803.28  803.66  804.04  804.77 19532    1
mu1         61.00    0.00  0.01  60.98  60.99   61.00   61.01   61.01 39620    1
mu2        141.00    0.00  0.01 140.98 140.99  141.00  141.01  141.02 40604    1
sigma1      12.00    0.00  0.01  11.98  11.99   12.00   12.00   12.01 23478    1
sigma2      12.11    0.00  0.01  12.09  12.10   12.11   12.11   12.13 18462    1
tau         83.39    0.16 10.85  57.60  76.68   85.63   92.20   97.72  4598    1
sigma_obs    2.10    0.00  0.11   1.90   2.02    2.09    2.16    2.32  7697    1

Samples were drawn using NUTS(diag_e) at Wed Apr  9 09:35:37 2025.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

Figure 7.21: 1

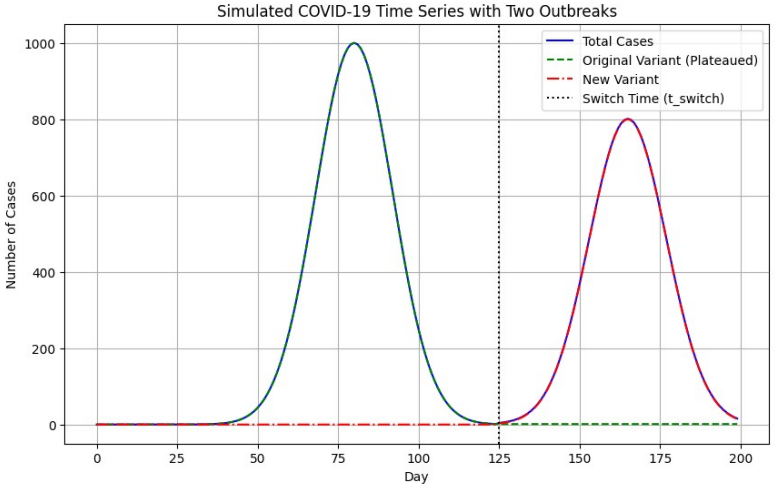

Figure 7.22: 1

```
> library(rstan)
> library(ggplot2)
> rstan_options(auto_write = TRUE)
> options(mc.cores = parallel::detectCores())
> df <- read.csv("dataset generation/multipleDatasets3.csv")  # Change the number here 1 2 3
> stan_data <- list(
+   N = nrow(df),
+   Total_Cases = df$Total_Cases
+ )
> fit <- stan(
+   file = "multipleChangePointsV2.stan",
+   data = stan_data,
+   iter = 12000, chains = 4, warmup = 4000,
+   control = list(adapt_delta = 0.9999, max_treedepth = 25),
+   seed = 42
+ )
> print(fit, pars = c("amp1", "amp2", "mu1", "mu2", "sigma1", "sigma2", "tau", "sigma_obs"))
Inference for Stan model: anon_model.
4 chains, each with iter=12000; warmup=4000; thin=1;
post-warmup draws per chain=8000, total post-warmup draws=32000.

             mean se_mean   sd   2.5%    25%     50%     75%   97.5% n_eff Rhat
amp1      1000.00    0.00 0.09 999.82 999.94 1000.00 1000.06 1000.19 29806    1
amp2       800.89    0.00 0.09 800.71 800.83  800.89  800.95  801.07 26050    1
mu1         81.00    0.00 0.00  81.00  81.00   81.00   81.00   81.00 35610    1
mu2        166.00    0.00 0.00 166.00 166.00  166.00  166.00  166.00 39060    1
sigma1      12.00    0.00 0.00  12.00  12.00   12.00   12.00   12.00 27814    1
sigma2      12.02    0.00 0.00  12.02  12.02   12.02   12.03   12.03 24194    1
tau        122.11    0.01 0.58 120.82 121.76  122.16  122.51  123.11  6072    1
sigma_obs    0.35    0.00 0.02   0.32   0.34    0.35    0.36    0.39  6393    1

Samples were drawn using NUTS(diag_e) at Wed Apr  9 09:55:58 2025.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

Figure 7.23: 1

The same Stan model was used to estimate the change point tau and only swapped out the dataset each time. The true change points for the datasets were day seventy five, one hundred and one hundred and twenty five respectively. In all three cases, the model gave very close estimates of the correct transition day.

For the first dataset where the true change point is seventy, the model returned tau is equal to 65.06. This is slightly earlier than expected, but still close enough to capture the switch between the two outbreaks. The big gap (non-overlap) between the two normal distributions might have contributed to the model predicting the change point a bit earlier. Stan is eager and tends to predict a bit early in situations like this as seen in stage 1.

In the second dataset (where true change-point at 100), the model estimated tau is equal to 83.39. Just like in the first dataset the estimation is a little bit early.

In the third dataset (where the true change point at 125), the model estimated tau is equal to 122.21, which is very accurate. This further confirms the model's flexibility and consistency in estimating tau across different outbreak timings.

Overall this section shows that the same model can be reused for different datasets as long as the structure remains the same and the priors are wide and flexible. The soft transition between the two Gaussians allowed for smoother modelling, and the results show that the model is capable of accurately identifying the point where the dominant variant changes. This is helpful in real-world scenarios where outbreak patterns shift over time, and a model is needed that can adapt without requiring structural changes every time.

7.10 Three Normal Distributions Dataset with Multiple Change Points

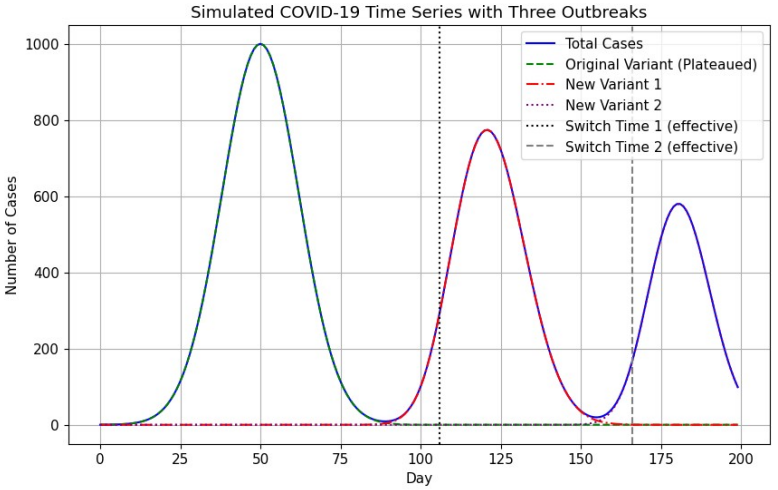

Figure 7.24: 1

To explore multiple distributions and multiple change points, a synthetic dataset was used that simulated a COVID-19 time-series data with three distinct waves of infections. This dataset spans two hundred days and includes an initial plateau (original variant) followed by three successive outbreaks representing new variants. This larger dataset is used to test a more complex model involving two change points.

There are two change points here. The first being at day sixty and the second being at day one hundred and fourty. The prior knowledge of the dataset is then incorporated into the model to assist in estimating the change points tau1 and tau2.

```
  model {
// Priors
tau1 ~ normal(60, 20);           // First change-point prior
tau2 ~ normal(140, 20);          // Second change-point prior
mu1 ~ normal(0, 1000);
mu2 ~ normal(0, 1000);
mu3 ~ normal(0, 1000);
sigma ~ normal(0, 100);

// Likelihood with soft transitions
for (n in 1:N) {
  real w1 = inv_logit((n - tau1) * 2);   // Transition from mu1 to mu2
  real w2 = inv_logit((n - tau2) * 2);   // Transition from mu2 to mu3
```

```
    // Soft combination of three segments
    real mu = (1 - w1) * mu1 + (w1 - w2) * mu2 + w2 * mu3;

    y[n] ~ normal(mu, sigma);
  }
}
```

Smooth transitions have been used as the distributions are continuous and differentiable. This makes making it easier for Stan's Hamiltonian Monte Carlo (HMC) sampler to explore the posterior efficiently. The soft transitions are modelled using logistic functions resulting in smooth blending between the three waves. This avoids sharp, unrealistic breaks in the model.

The mu variables for this model (unlike previous) is used to represent the average number of cases in different parts of the data. So mu1 represents the time before tau1, Mu2 represents the time between tau1 and tau2. Mu3 represents the time after tau2. So the mu variables represent different portions of time relative to the change points tau1 and tau2. This is why there is no amplitude variables in the model.

```
> print(fit, pars = c("mu1", "mu2", "mu3", "sigma", "tau1", "tau2"))
Inference for Stan model: anon_model.
4 chains, each with iter=12000; warmup=4000; thin=1;
post-warmup draws per chain=8000, total post-warmup draws=32000.

        mean se_mean    sd   2.5%    25%    50%    75%  97.5% n_eff Rhat
mu1    42.94    0.29 50.02 -56.73   9.72  43.16  76.64 140.31 29874    1
mu2   453.08    0.15 26.01 401.58 435.77 453.07 470.61 503.92 30431    1
mu3   246.91    0.18 32.77 182.21 225.11 246.99 268.95 310.69 34716    1
sigma 256.81    0.07 12.71 233.16 248.16 256.29 264.87 283.31 31013    1
tau1   31.04    0.01  2.36  26.21  29.51  31.10  32.60  35.50 26844    1
tau2  136.68    0.02  3.09 130.74 134.62 136.63 138.71 142.88 28641    1

Samples were drawn using NUTS(diag_e) at Wed Apr 16 19:05:34 2025.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
>
```

Figure 7.25: 1

This model took only one minute to fit and was significantly faster than the previous stages' models. One of the reasons this model fits significantly faster than the earlier Stage one model is due to its simplicity in structure and computation. In Stage one the outbreak waves were modelled using full Gaussian curves, where each wave had its own amplitude, peak day and standard deviation. The stage one model required evaluating exponential functions for every data point in the time series, as well as estimating a greater number of parameters. In comparison the Stage One model took fifteen minutes to fit.

In contrast, the current model used for the three-outbreak scenario is far more lightweight and only takes one minute to fit. It models the number of cases in each phase as a constant mean, rather than fitting complex curves. The segment transitions are handled using logistic functions (inv-logit), which smoothly blend the means across phases without introducing discontinuities. This approach avoids expensive exponential evaluations for curve shapes and removes the need for separate amplitude and spread parameters.

Additionally this model only estimates six parameters in total: three means (mu1, mu2, mu3), two change-points (tau1, tau2), and one noise parameter (sigma). There are no latent variables or per-day hidden states as all transformations within the loop are deterministic and derived from the parameters. There is no transformed parameters block. This makes the posterior geometry simpler and easier for Stan's Hamiltonian Monte Carlo (HMC) sampler to navigate.

Suprisingly the tau1 variable outputted is equal to 31 even though the first change point is around 90. This occurs because the model is treating the first 15–20 days of the dataset as a separate distribution. The model considers the inital flat part of the dataset to be its own distribution as thus estimated tau1 to be 31. During this early period, the number of cases is very low or nearly flat (representing the original variant plateau), which is statistically distinct from the rising phase of the first outbreak.

By using soft transitions, the model blends the transition from this early flat phase (mu1) into the first peak phase (mu2) gradually. Therefore, it interprets the end of the quiet/plateau period as a meaningful change-point, even if it doesn't yet look like an outbreak visually.

As a result, tau1 marks the start of meaningful change in the dataset, not the change point between the two distributions. This is consistent with how Bayesian change-point models work as they look for changes in the underlying data-generating process. Tau2 then is estimated to be 136.68 which is earlier than its actual value. The estimation is in between the two change points accounted for at the beginning and is not that accurate as it considers the first change point to be between the plateau at the beginning and the first distribution. It turns out there is three change points in the model. The different change points can be focused on by introducing contraints in model block of the Stan file by specifying a range. Like so

```
    parameters {
  real<lower=80, upper=120> tau;
}
model {
  tau ~ normal(100, 50) T[80, 120];
}
```

7.11 Stage Four

In this section the Stage one solution will be used with the two hundred day dataset and improvements will be made. These include making fitting the model quicker, using R to take in the prior for the change point tau and implementing a searching method. The stage one, two hundred day dataset solution currently takes eleven hours and fourty six minutes to fit as indicated by the following screenshot. All the estimates for the other variables can also be seen.

```
> library(rstan)
> library(ggplot2)
> rstan_options(auto_write = TRUE)
> options(mc.cores = parallel::detectCores())
> df <- read.csv("dataset generation/datasetStageOne.csv")  # Change when needed
> stan_data <- list(
+   N = nrow(df),
+   Total_Cases = df$Total_Cases
+ )
> Sys.time()
[1] "2025-04-17 22:33:38 IST"
> fit <- stan(
+   file = "stageOneBiggerDataset.stan",
+   data = stan_data,
+   iter = 12000, chains = 4, warmup = 4000,
+   control = list(adapt_delta = 0.9999, max_treedepth = 25),
+   seed = 42
+ )
> Sys.time()
[1] "2025-04-18 10:19:47 IST"
> print(fit, pars = c("amp1", "amp2", "mu1", "mu2", "sigma1", "sigma2", "tau", "si
gma_obs"))
Inference for Stan model: anon_model.
4 chains, each with iter=12000; warmup=4000; thin=1;
post-warmup draws per chain=8000, total post-warmup draws=32000.

             mean se_mean   sd    2.5%     25%     50%     75%   97.5% n_eff
amp1      1000.00    0.00 0.00 1000.00 1000.00 1000.00 1000.00 1000.00 28490
amp2       800.00    0.00 0.00  800.00  800.00  800.00  800.00  800.01 26103
mu1         51.00    0.00 0.00   51.00   51.00   51.00   51.00   51.00 35519
mu2        151.00    0.00 0.00  151.00  151.00  151.00  151.00  151.00 37358
sigma1      10.00    0.00 0.00   10.00   10.00   10.00   10.00   10.00 26023
sigma2      10.00    0.00 0.00   10.00   10.00   10.00   10.00   10.00 23478
tau         97.04    0.08 3.32   89.05   95.03   97.74   99.72  101.40  1581
sigma_obs    0.00    0.00 0.00    0.00    0.00    0.00    0.00    0.00  2477
          Rhat
amp1         1
amp2         1
mu1          1
mu2          1
sigma1       1
sigma2       1
tau          1
sigma_obs    1

Samples were drawn using NUTS(diag_e) at Fri Apr 18 10:19:36 2025.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
>
```

Figure 7.26: 1

Another Stan file was created as well as an R script that took in the variables for tau's normal distribution's mean and spread. The variables were passed to Stan through R. This cut the time taken to fit the model roughly in half. So the value for the mean and spread are taken like this.

```
  stan_data <- list(
  N = nrow(df),
  Total_Cases = df$Total_Cases,
  tau_prior_mean = 100,  # YOU SET THIS
  tau_prior_sd = 5     # AND THIS TOO
)
```

The result of this change can be seen in the output of the model.

```
> Sys.time()
[1] "2025-04-19 22:18:21 IST"
> fit <- stan(
+   file = "stage4.stan",
+   data = stan_data,
+   iter = 12000, chains = 4, warmup = 4000,
+   control = list(adapt_delta = 0.9999, max_treedepth = 15),
+   seed = 42
+ )
Warning messages:
1: There were 7644 transitions after warmup that exceeded the maximum treedepth. Increase max_treedepth above 15. See
https://mc-stan.org/misc/warnings.html#maximum-treedepth-exceeded
2: Examine the pairs() plot to diagnose sampling problems

> Sys.time()
[1] "2025-04-20 03:30:02 IST"
> cat("\a")  # Terminal bell (depends on system settings)

> print(fit, pars = c("tau", "amp1", "amp2", "mu1", "mu2", "sigma1", "sigma2", "sigma_obs"))
Inference for Stan model: anon_model.
4 chains, each with iter=12000; warmup=4000; thin=1;
post-warmup draws per chain=8000, total post-warmup draws=32000.

             mean se_mean  sd    2.5%     25%     50%     75%   97.5% n_eff Rhat
tau         97.12    0.16 3.3   89.52   95.13   97.82   99.76  101.39   431    1
amp1      1000.00    0.00 0.0 1000.00 1000.00 1000.00 1000.00 1000.00 28247    1
amp2       800.00    0.00 0.0  800.00  800.00  800.00  800.00  800.01 24730    1
mu1         51.00    0.00 0.0   51.00   51.00   51.00   51.00   51.00 34998    1
mu2        151.00    0.00 0.0  151.00  151.00  151.00  151.00  151.00 37005    1
sigma1      10.00    0.00 0.0   10.00   10.00   10.00   10.00   10.00 27156    1
sigma2      10.00    0.00 0.0   10.00   10.00   10.00   10.00   10.00 24025    1
sigma_obs    0.00    0.00 0.0    0.00    0.00    0.00    0.00    0.00   931    1

Samples were drawn using NUTS(diag_e) at Sun Apr 20 03:29:52 2025.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
>
```

Figure 7.27: 1

Looking at the results using R for taking in the input for tau's normal distribution has resulted in a faster execution time. Using R took five hours sixteen minutes to fit the model whereas when using Stan for inputting tau's normal distribution's values it took eleven hours and fourty six minutes. This is more than a fifty percent improvement in execution time. Also by using the R programming language to take in the input it is easier to run the model with different values for tau's normal distributions' values as there is no need to change the Stan file. The downside to using R to take in input is that it there is a warning message that states that there were 7644 transitions that exceeded the maximum tree depth.

Possibly the reason for this happening is because When the prior parameters for tau (such as the mean and standard deviation) are passed from R into Stan as data the sampler treats them as dynamic values. This increases complexity to the posterior geometry. As a result Stan's No-U-Turn Sampler (NUTS) can require deeper exploration to fully capture the posterior. This leads to more transitions that exceed the maximum tree depth. Even though the model may run faster it can cause increased sampling difficulty and more frequent warnings during inference. Stan takes bigger steps estimating tau when it takes in the parameters from R. This could be fixed by increasing the maximum tree depth.

Chapter 8: **Summary and Conclusions**

This project applies Bayesian statistical methods to perform change-point inference on synthetic time-series datasets simulating epidemic outbreaks. It can be applied to any time series data that contains change points. Change points where the underlying statistical model shifts are critical in contexts such as epidemiology, where they may signal new virus variants or treatment interventions. Traditional probabilistic tools often struggle with discrete model transitions. This project addresses that limitation using the Stan probabilistic programming language. The project uses Stan, R and python. R is used as an interface to Stan and python is used to generate the datasets. The R studio IDE was used as well.

Stage One focused on building a working Stan model to estimate the change point, tau, in a synthetic time-series dataset containing two Gaussian distributions that do not overlap. A smaller fifty day dataset was used first to reduce run time, making it easier to debug and adjust the model. Once the model gave good results, it was tested on a larger two hundred day dataset to check if the model could scale and still estimate tau accurately.

Stage Two introduced overlapping distributions, which makes the inference problem more realistic and harder. The dataset simulated two COVID-19 variants where the outbreaks overlapped around the change point. The Stan model had to reflect this overlap, and the parameters were adjusted to account for it. Even with the overlap, the model was able to identify the correct change point. This stage proved that the model is capable of handling transitions that are not clearly separated and still provide accurate and stable results.

Stage Three experimented by working with different types of distributions by combining a Gaussian distribution with a gamma distribution. This introduced more complexity and tested the model's flexibility. The change point tau was re-parametrised using an unconstrained parameter and a logistic transformation to improve sampling efficiency and avoid divergent transitions. The use of a sigmoid function to blend the two distributions allowed for smooth transitions. This stage showed that Stan can be used to model more complex changes in the data and handle different types of distributions.

Stage Four focused on optimising model performance and making the workflow more flexible in terms of changing the prior for the change point tau. The prior for tau was passed in through R instead of being hardcoded in the Stan file, which helped cut down the model's run time by more than half. This approach made it easier to test different prior values without modifying the Stan code and reduced the runtime by more than fifty percent.

The Flexible prior with variable dataset section explored three different datasets with three different change points which was seventy five, one hundred and one hundred and twenty five. The datasets were the variable in this section and all three datasets used the same Stan file and R script. This showed that the same model could adapt to different scenarios without adjusting the priors. It confirmed that the approach can generalise well across a wide range of time-series datasets. This helps the users when specifying the values for tau as they can fewer number of times with a wide flexible prior instead of many times with narrow less flexible priors. This helps the users of the project when specifying values for the mean of the prior of tau. This is because the estimation of the model is influenced more by the observed data and is taking into account many possible values for tau.

The Three Normal Distributions Dataset with Multiple Change Points section explores a more complex scenario. It involves three distinct outbreak phases within a synthetic COVID-19 time-series

dataset spanning two hundred days. The dataset was modelled using three normal distributions, with two main change points. The first one was around day sixty and the second around day one hundred and fourty. A Stan model was implemented using soft transitions between segments through logistic (inv-logit) functions, allowing smooth blending from one distribution to the next without sharp breaks. Unlike previous models that estimated detailed parameters like amplitudes and spreads, this model took a simpler approach by only estimating the average case levels (mu1, mu2, mu3), two change-points (tau1 and tau2), and one noise parameter (sigma). This simplification led to a major reduction in run time as the model was fitted in just one minute compared to over fifteen minutes in earlier stages. Interestingly and unexpectedly the model identified the first change point much earlier than expected (tau1 = 31) due to its interpretation of the early flat period as a distinct distribution. This revealed that the model effectively detects shifts in the dataset. The section concludes by noting that different change points can be targeted by constraining tau's range in the model block, making the method adaptable for scenarios with multiple or hidden transitions.

The models in all four stages consistently estimated change points with high accuracy and stability as indicated by low R-hat values and high effective sample sizes in the output. Notably, the framework demonstrated flexibility in detecting multiple change points and adapting to diverse datasets. These results suggest that Stan is a powerful tool for dynamic model selection and change-point analysis in time-series settings. Further research can be done for building a method to search for the change point automatically. Further research can be done to investigate the Stan models which are eager (guess the change point a bit early) and the best way to deal with this. Further research can be done to observe whether the change point can be seen as a range instead of a singular value. This could be useful in the case where all the distributions in a dataset are equal to zero on a particular day.

Acknowledgements

I would like to sincerely thank my supervisor, Fintan Costello for his invaluable guidance, support and encouragement throughout the course of this project. His insights and expertise have been instrumental in shaping the direction of my work and helping me overcome the challenges I encountered. I am deeply appreciative of the time and effort he dedicated to providing feedback and fostering my development throughout this academic journey.

Bibliography

1. Stan. *Intro to gaussian processes in Stan: Finding exoplanets* https://www.youtube.com/watch?v=132s2B-mzBg&list=PLCrWEzJgSUqwL85xIj1wubGdY15C5Gf7H&index=8 (2024).
2. Stan. *Split testing in Stan: Should I make a Twitter ad or a Facebook ad?* https://www.youtube.com/watch?v=Z1IB4FT_Yog&list=PLCrWEzJgSUqwL85xIj1wubGdY15C5Gf7H&index=3 (2025).
3. Stan. *Linear regression made easy with Stan* https://www.youtube.com/watch?v=UQtFkEOg9SM&list=PLCrWEzJgSUqwL85xIj1wubGdY15C5Gf7H&index=5 (2025).
4. Stan. *Convergence checks in Stan tutorial* https://www.youtube.com/watch?v=0FdMZwIbJ_4&list=PLCrWEzJgSUqwL85xIj1wubGdY15C5Gf7H&index=6 (2025).
5. Stan. *Modelling heteroscedasticity tutorial* https://www.youtube.com/watch?v=nwuU-KEKXhU&list=PLCrWEzJgSUqwL85xIj1wubGdY15C5Gf7H&index=7 (2025).
6. Stan. *Hierarchical modelling tutorial* https://www.youtube.com/watch?v=dNZQrcAjgXQ&list=PLCrWEzJgSUqwL85xIj1wubGdY15C5Gf7H&index=8 (2025).