STA365: Assignment 1

Eric Zhu

12/03/2021

Contents

Quick Introduction

To keep in mind:

Our STD is Chlamydia, which starts out as an EB, and turns into an RB. We need a lot of it to measure the effects, so the units are $10^4 = 10000$ EBs. For example $C(t) = 1 \Rightarrow 10^4$ EBs at time t.

Data was collected in two waves:

1. Female Guinea pigs were infected with known number of EBs (the original form of the STD).
2. Male Guinea pigs were infected with STD to see how disease spreads due to intercourse.

Number of EBs are an **average** across Guinea pigs with the same dose of EBs.

Head of our data from wave 1:

t	C	dose	number
3	0.000	10^1	2
6	46.829	10^1	2
9	9.106	10^1	2
12	18.862	10^1	2
15	25.366	10^1	2
18	21.463	10^1	2

Head of data from wave 2:

t	C	dose	number
3	35.122	sexual	11
6	11.057	sexual	11
9	11.707	sexual	11
12	0.650	sexual	11
15	0.000	sexual	11
18	0.000	sexual	11

We will use an ODE model to model the count of Chlamydia EBs. We will use the following 3 differential equations:

$$\begin{cases} \frac{\partial E}{\partial t} = P_e - \delta_E E(t) - k_1 C(t) E(t) \\ \frac{\partial C}{\partial t} = P k_2 I(t) - \mu C(t) - k_1 C(t) E(t) \\ \frac{\partial I}{\partial t} = k_1 C(t) E(t) - \gamma I(t) - k_2 I(t) \end{cases}$$

A note about e

Finally, it is also worth nothing that e will mostly be referred to as scientific notation, i.e., $e2 = 100$ and $e - 2 = 0.01$. The exponent base e will be explicitly referred to as **exp**.

Task 1: Exploring the model and setting priors

Suppose y_i is the cell count of Chlamydia EBs at time i for a particular Guinea Pig, where $y_i \sim N(\mu, \sigma)$ by the CLT. We have that μ is the solution to our set of differential equations for the rates of change over time in healthy cells, Chlamydia EBs, and infected cells.

We also know some **key pieces of information**, namely:

1. Chlamydia EBs should be mostly cleared by $t = 30$, i.e., $C(t) < 0.1, t \geq 30$.
2. Distribution of maximum values of $C(t)$ should cover the value $M, M \in (100, 250)$ when $C(0) \in (0.001, 1)$.
3. The lysis event produces hundreds of cells, i.e., $k_2 I(t) \geq 100 \cdot 10^{-4}$.
4. If $C(0) = 0.1$ then the parameters $P = 1000, k_1 = 0.1/1e-4, k_2 = 0.8, Pe = 40 \cdot 1e-4, \delta_E = 2, \gamma = 1.2, \mu = 1.2$ produce a range of approximately [0.1,177] for $C(t)$, which consistent with bullet point 2 above.
5. We take $E(0) = 9600 \times 10^{-4}$ and $I(0) = 0$.

Since we have that P_E, δ_E are both fixed to $40 \times 10^{-4}, 2$ respectively, we need to place priors on only P, k_1, k_2, γ, μ. Note that the prior on μ is not the same μ as the location parameter for our y_i. We will also need a prior on σ, which is to account for the variance of y_i.

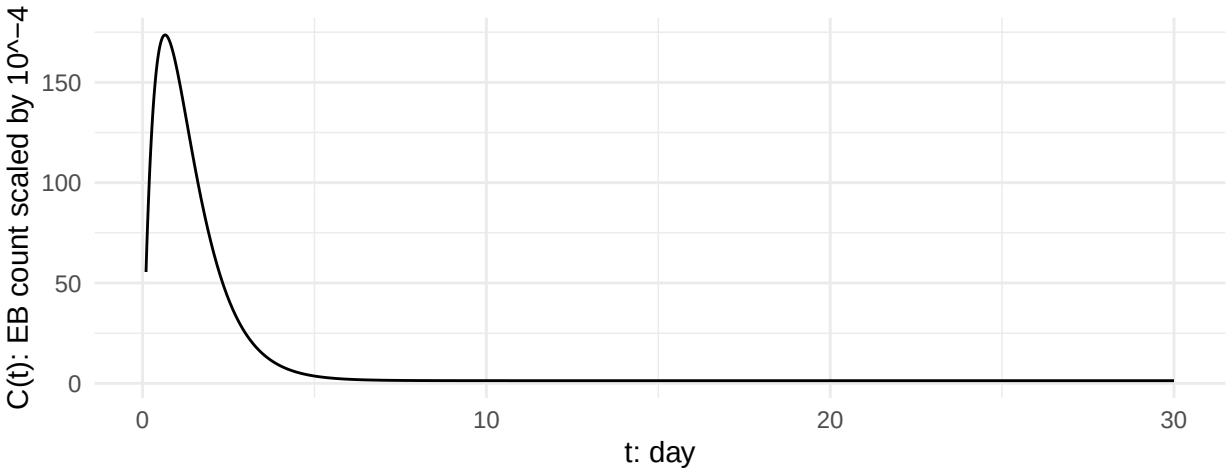
These priors will also be normally distributed due to the CLT.

We begin by considering bullet point 4 from our key info to give us a nice starting point for centering our parameters. Also note that it's not sensible for us to have negative counts of cells.

From bullet point 2, we know that $C(0)$ should range on the interval $[0.001, 1]$. As such, we can take 0.001 to be a sort of lower bound and 1 to be a sort of upper bound. It makes sense that for a fixed set of parameters (and time scale), we could expect to see a higher peak value given a higher initial dose. We can begin with considering what would happen at $C(0) = 0.001$. We will use `Stan` and start with the parameters from bullet point 4. The goal of this step is to figure out which combination of parameters nearly violate our key information.

Note that this should be deterministic, i.e., we make no probabilistic statements here.

Here's the resulting time course plot:



So clearly, we get that the values for $C(t)$ are well within our stated counts for EBs. But, it doesn't quite model the situation correctly, since we still have $C(t)$ values around 1 at day 30. So we need to find some parameter values such that the infection is about clear around day 30. Additionally, we want to keep in mind that over various $C(0)$ values from $[0.001, 1]$, we get maximum values easily range over $[100, 250]$. So ideally, we get two sets of parameters that where one has a peak somewhat under 100 and one has a peak somewhat

over 250 because these will be encoded as distributions, i.e., we cannot expect the maximum values to be exactly bounded below by 100 and above by 250.

Interestingly the behaviour of $C(t)$ appears to be a monotonically decreasing function, so if we apply monotonic transformations to the parameters we currently have, we should still obverse the same behaviour we see here.

Let's first find an upper bound for the parameter values given $C(0) = 0.001$, while keeping in mind the behiavour of $C(t)$ given other $C(0)$ values. Note that this will serve as our 2τ bound as that roughly corresponds to the 97.5th percentile of $C(t)$ values given our assumed normal distribution.

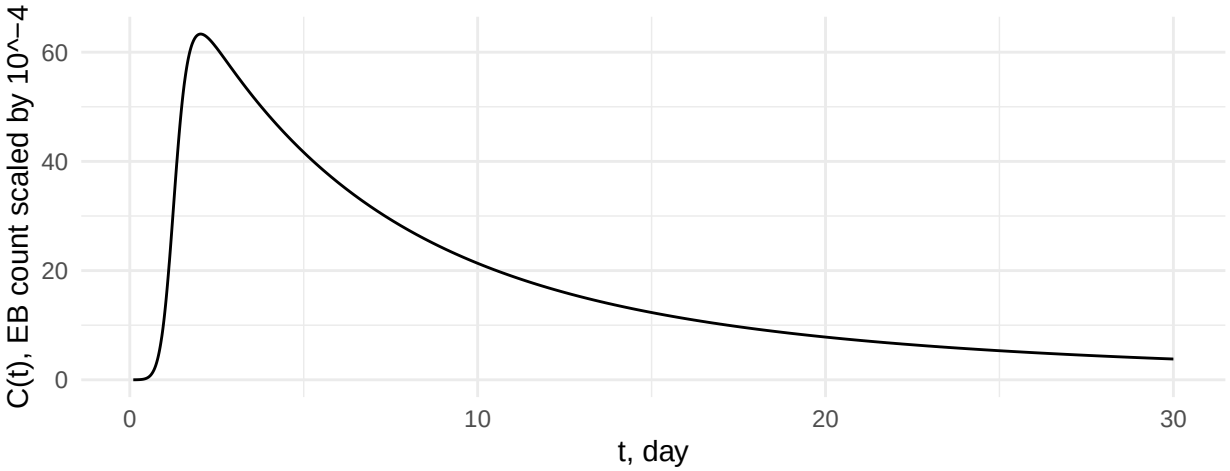
Intuitively, looking at our differential equations, we see that the change in EB counts are directly related to the μ parameter from the "death" component of $\frac{\partial C}{\partial t}$ and the k_1 parameter. So in order to find these high values for $C(t)$, we'd want to minimize parameters like μ because they contribute to a greater negative change in C per unit of time. Conversely, we'd like to increase parameters such as P, k_2 since they contribute to a more positive C value. The two k parameters k_1, k_2 are shared between at least one equation. Specifically k_1 is shared between all 3 as it appears to be a scaling factor for the rate of infection, and from preliminary exploration, we find that a decrease in k_1 is associated with a decrease in C.

From using a grid-search approach (done by hand but could be done computationally), we were able to narrow down some parameter values that achieve our goal of finding the upper 5% of C values given $C(0) = 0.001$. But we also note that the "default" parameters provided for $C(0) = 0.1$ are a **very** good starting points, and the "defaults" acted as centres for our search. Then, we multiplied the "defaults" by certain multipliers to get our new parameter values. Specifically, we took:

1. $P = 1000 \cdot 1.125$
2. $k_1 = \frac{0.1}{1e-4} \cdot 1e-4$
3. $k_2 = 0.8 \cdot 4$
4. $\gamma = 1.2 \cdot 1.1$
5. $\mu = 1.2 \cdot 0.165$
6. $P_E = 40 \cdot 1e-4$
7. $\delta_e = 2$

Note that parameters 6,7 (P_e, δ_e) are fixed.

Here's the resulting plot (the corresponding plot for $C(0) = 1$ is in the appendix under "task 1 plots"):



Conversely, we want to find the set of parameters that give us low values for C. These will act as the 5th percentile, i.e., those at -2τ.

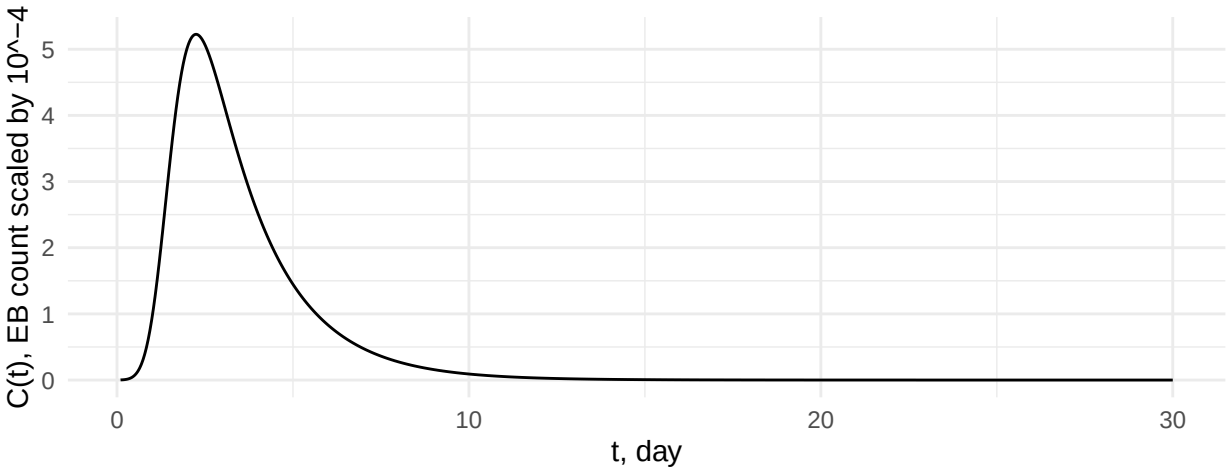
To obtain this set of parameters, we want to do the opposite of what we did before. For example, we'd want to increase the value for μ because, we want the death of EBs to increase and for the infection rate to increase. We'd also want the rate of lysis bursts to increase too.

Again, we used a "grid search" approach done by hand. We used multipliers of the "defaults", while keeping in mind the other inital values for C. Specifically, we took:

1. $P = 1000 \cdot 0.875$
2. $k_1 = 0.00000875/1e-4 = 0.875$
3. $k_2 = 0.8 \cdot 3.85$
4. $\gamma = 1.2 \cdot 1.45$
5. $\mu = 1.2 \cdot 0.55$
6. $P_E = 40 \cdot 1e-4$
7. $\delta_e = 2$

Note again that parameters 6,7 (P_e, δ_e) are fixed.

Here's the resulting plot (the corresponding plot for $C(0) = 1$ is in the appendix under "task 1 plots"):



Now we've got two sets of parameters that correspond to high values of C for varying initial states and low values of C for varying initial states:

1. $P = (875, 1125)$
2. $k_1 = (0.875, 1)$
3. $k_2 = (3.08, 3.2)$
4. $\gamma = (1.32, 1.68)$
5. $\mu = (0.198, 0.6)$
6. $P_E = 40 \cdot 1e-4$
7. $\delta_e = 2$

Note that the first element of the tuples correspond to "low values".

These are sensible bounds for our distribution of parameters because they are consistent with all of the key pieces of information we know about this situation. In particular, they all represent parameters where the infection clears up at day 30 while the maximum values of $C(t)$ cover the values of M, i.e., $[100, 250]$.

Since we set these to be $\pm 2\tau$, we know that the average is the centre of the distribution for these parameters. Additionally, we can find τ for each parameter. After some calculations, we get distributions (μ, τ) for the first 5 parameters (the last two are fixed):

1. $P \sim N(1000, 125)$
2. $k_1 \sim N(0.9375, 0.0625)$
3. $k_2 \sim N(3.14, 0.06)$
4. $\gamma \sim N(1.5, 0.18)$
5. $\mu \sim N(0.399, 0.201)$

Finally, we find a distribution for σ, the standard deviation of y_i. The most important consideration here is the idea that we really shouldn't have negative C values because negative cell counts are illogical. It follows then that σ must be constrained by low values of y_i, i.e., the counts of EBs, which we also know as C. In constructing our priors, we know that it is possible for us to get C values on the order of 10^{-6}. We didn't see any values on the order of 10^{-7} or less while setting priors. Thus, our τ_σ must also be at most on the order of 10^{-6}, as we don't want to get negative values for C. The distributions of our priors already encode all of our key information about C, and we expect to be centered around the centre of y_i, i.e., μ. So μ_σ should be 0. So considering that our other prior distributions should produce quantities of μ that meet our key pieces of information, what should we do with τ_σ?

Here's where we believe we should consider our priors to be weakly informative, i.e., we only know some information about our problem. Clearly, we don't know much about biology or Chlamydia specifically beyond the paragraph at the start of the assignment. As such, we should have a fairly relaxed value for τ_σ. Again, we'll set a value for $2\tau_\sigma$. Recall that our lowest values for C we observe with our 2τ values for the other parameters lead us to values on the order of 10^{-6}. For example:

mu	time	chain_iter	cell_type
1.5e-06	30.00	chain 1, iteration 1	c
1.5e-06	29.99	chain 1, iteration 1	c
1.5e-06	29.98	chain 1, iteration 1	c
1.6e-06	29.97	chain 1, iteration 1	c
1.6e-06	29.96	chain 1, iteration 1	c
1.6e-06	29.95	chain 1, iteration 1	c

Thus, we set $2\tau_\sigma$ to be $9e-7$, and which means $\tau_\sigma = 4.5e-7$. This is about as big as τ_σ can get while still allowing for values of y_i to be greater than 0.

However, in actually running the model, having such a small τ_σ runs into numerical stability issues and with the MCMC sampler sampling values for σ that are essentially 0. Taking this issue of practicality into consideration, we increased τ_σ to $4.5e-1$. This will give us negative values for y_i, but we just note that negative values are nonsensical and an artifact of numerical stability concerns. Additionally, most sampled y_i values from the posterior predictive distribution should still be positive and we should still see behaviour consistent with a smaller τ_σ value, i.e., the infection clears up around day 30.

To recap, our list of distributions for parameters is:

1. $P \sim N(1000, 125)$
2. $k_1 \sim N(0.9375, 0.0625)$
3. $k_2 \sim N(3.14, 0.06)$
4. $\gamma \sim N(1.56, 0.12)$
5. $\mu \sim N(0.399, 0.201)$
6. $\sigma \sim N_+(0, 4.5e-1)$

Task 2: Fit the data (experimental conditions).

Quick Overview

In fitting the model to all 4 "participants" (Guinea Pigs), we found that σ showed the most evidence of prior-data conflict, i.e., there was considerable distance between the distribution of σ from the posterior and that from the prior. We will over this most in detail in the section regarding "participant 1" with other sections being less detailed due to avoid verbosity and redundancy. Additionally, all other parameters (μ, γ, P, k_1, k_2) do not show much evidence, if any, of prior-data conflict across all "participants". We call them "participants" because while it is technically an average over different sized groups, "participant" is much more concise.

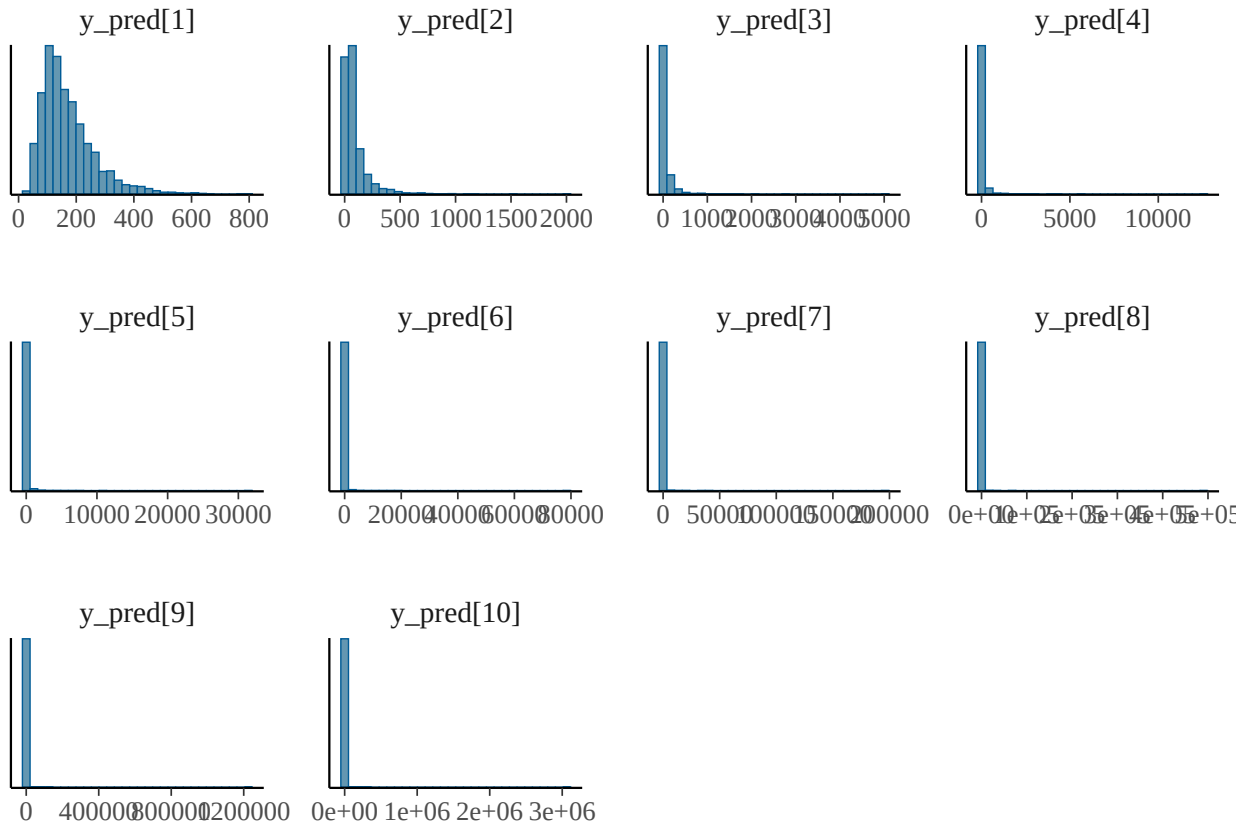
This model indeed doesn't seem to be a very good model of a Chlamydia infection, particularly with $C(0) = 0.001$ as made obvious with the time course plot of predicted values of participant 1. As such, we didn't try to really adjust τ_σ to that the distributions really overlap; as we will see, the posterior values of sigma are quite far from 0, and adjusting τ_σ would easily allow for extremely negative values for y_i, which doesn't seem sensical given that y_i is a cell count. This will be explained further below with graphs.

Finally, for every graph that overlays the prior distribution over the posterior distribution, the distribution coloured in darker blue is the posterior distribution (and the lighter is the prior distribution). The legend was elimated for clarity sake and to avoid redundancy in a grid.

Participant 1: Guinea Pig data with number = 2

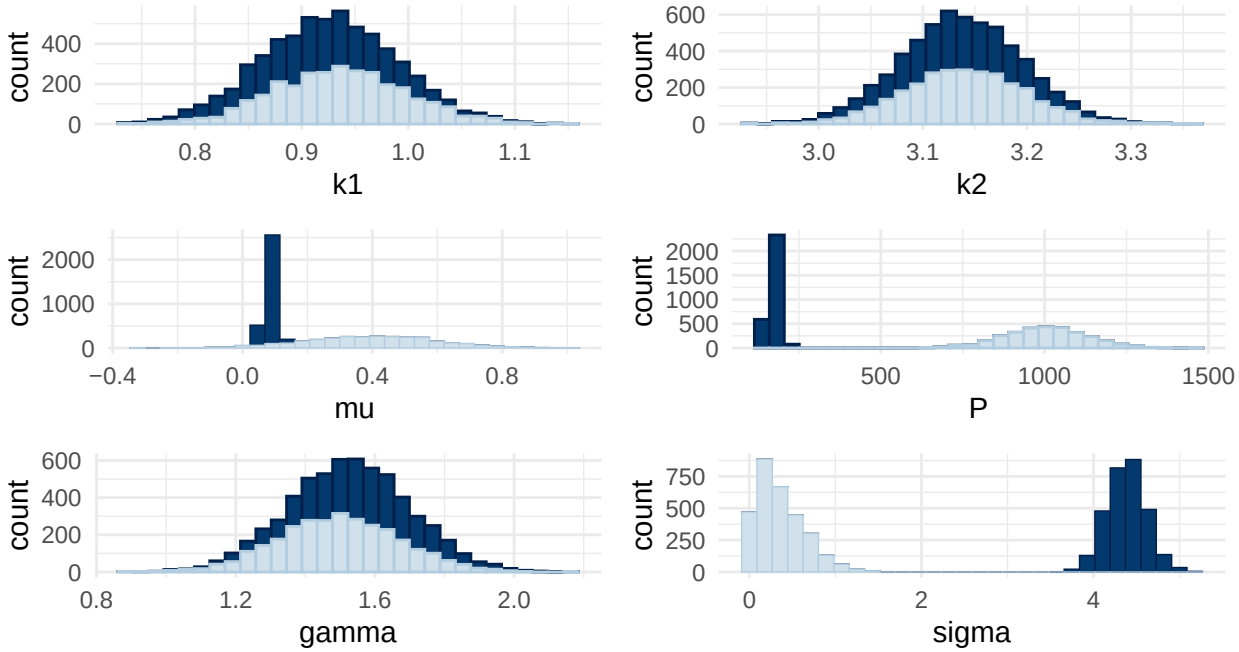
Prior predictive checks/Comparison of prior and posterior distributions

First, we'll take a look at the prior predictive distribution:



We see from the histograms across our range of time points (days 3 to 30) that the distribution of EB counts

are reasonable with the exception of consistently extremely tails. Note that these extremely long tails are less than 5% of the total density and the summary of the prior predictive distribution shows 95th percentile quantities that are consistent with what we'd expect, e.g., 353.4148500 for the 95%th percentile of `y_pred[1]`. Additionally, they look good as the centre of the prior posterior distribution moves from being centered around 200 to around 0, which is behaviour we know from the key information as it goes from having an infection to having it mostly cleared up.



Next, looking at the plots of the prior distributions versus the posterior distribution for each parameter, we see that the posterior distributions for all parameters contract within the prior distribution, with the exceptions of σ, P. σ as stated in the quick introduction shows substantial evidence of prior-data conflict due to the large distance in between the centres of the prior and posterior distribution; however, the posterior distribution is in a location where there is still density under the prior distribution. Since the location of the posterior distribution of σ is far from 0, we could increase the centre of the prior on σ, but that would be actively making it so that we would get negative y_i values, which are EB counts; thus it doesn't seem sensible to increase the centre of σ. The difference in centres could be due to the model not being a good model for the count of EBs, similar to how using model 2 on homework 1 resulted in prior-data conflict for σ. In homework 1 the data in homework 1 was generated using model 1. The plot for P also shows some signs of prior data conflict but there is clearly some density from the prior distribution covering that of the posterior. We also decided to not change the prior on P because we weren't able to find a set that worked better and was "more justified" as weakly informative priors, which is what we went for in task 1. We will further justify why we kept these priors later on with a time course plot, ancillary test statistics, and a density overlay.

We did originally set τ_σ to $4.5e-7$, but as mentioned above, but we increased it, as mentioned above, due to numerical stability issues. We settled on the current value of τ_σ because it gives us posterior predicitive distributions that conform to counts of EBs not being really negative as is evident in the time course plot while also allowing for the model to complete.

Evaluation of ancillary test statitics

Recall that since we are using normal distributions, reasonable ancillary test statistics are `min`, `max`, and `skewness`.

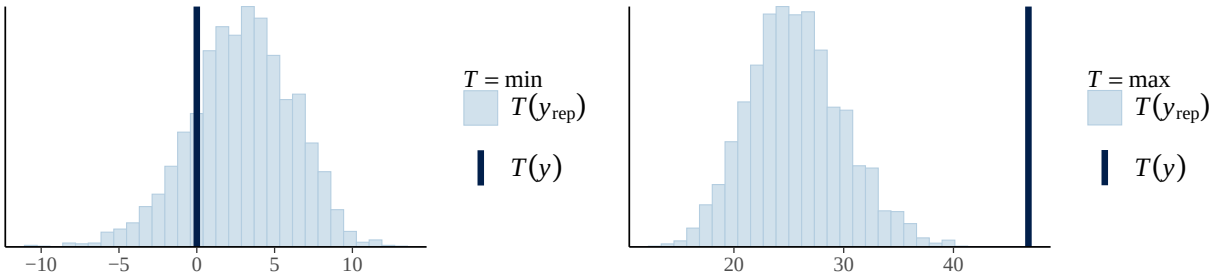


Figure 1: Test statistics plots of min, max - participant 1

From the two ancillary test statistics above, we see that the model was able to capture the `min` test statistic quite well while the model was unable to capture the `max` test statistic. The model actually captured the `min` test statistic quite well; the data `min` is almost right at the centre of the $T(y_{rep})$ distribution. We believe that the model was unable to capture the `max` test statistic because when fitting the model, we notice that the data for participant 1 mostly includes 0s, and so the max of the observations for participant 1 seems like a sort of "influential point". We can see this further down below in the time course plot, where the y_pred draws show that the model is unable to predict for the peak measurement of participant 1. So again, this may just be that the model we choose to use, i.e., the differential equation model is a poor model for this data.

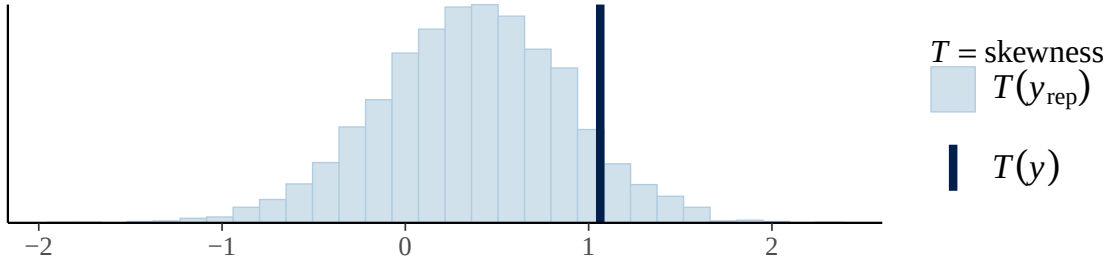


Figure 2: Test statistics plots of sknewness - participant 1

We see again that the model was fairly well able to capture the skewness test statistic as it's relatively close to the centre of the $T(y_{rep})$ distribution.

Evaluation of the time course plot

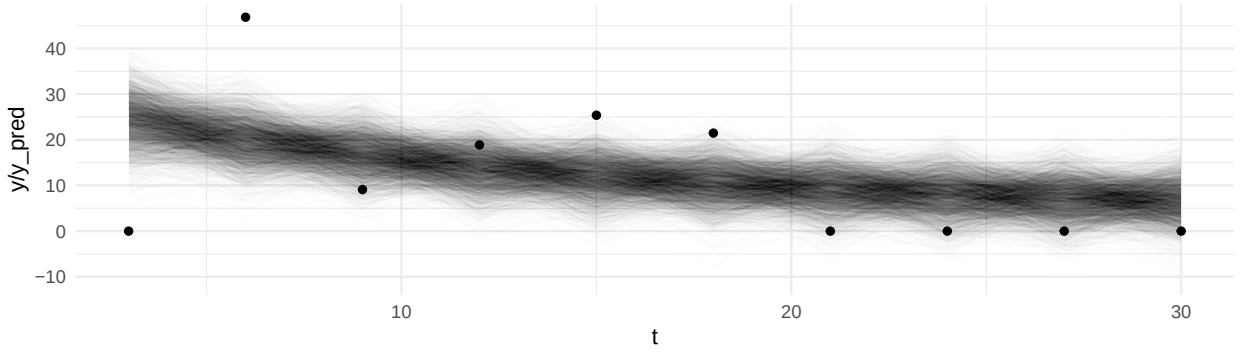


Figure 3: Time course plot - participant 1

We see that the model was unable to capture the actual behaviour of the data, i.e., the shape we saw when choosing the priors, where it was steeply upwards sloping and steeply downwards sloping with eventually asymptotic behaviour. Instead, this model fit a weakly downwards sloping trend that went through the "centre" of the data points, sort of similar to model 1 of homework 1. It was unable to capture points like a count of 0 at day 3 and the peak measurement, but was actually able to capture the majority of the points. This could very well be the reason of the prior-data conflict we see above; the model is just a really bad model at capturing the true behaviour of the EB counts. Additionally, other various combinations of prior values didn't change the behaviour of this model, which contributed to us keeping these priors.

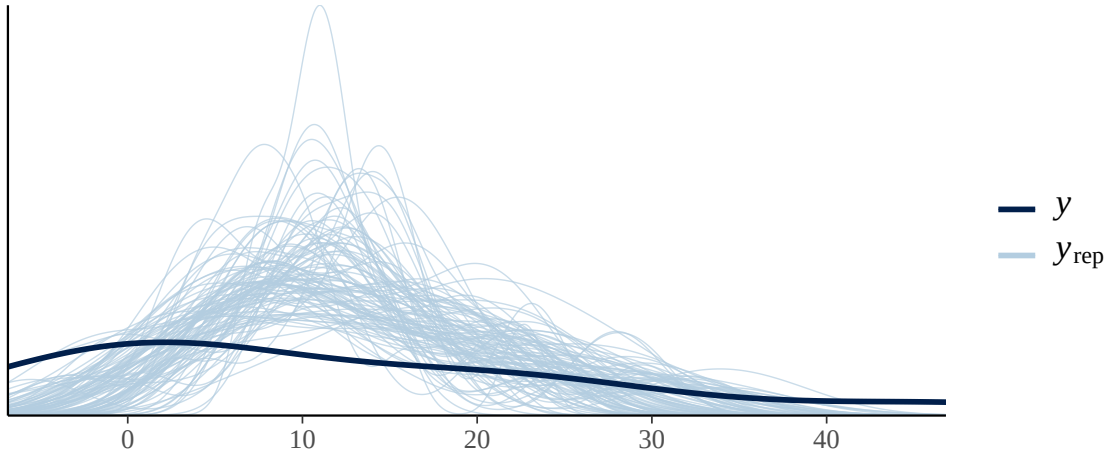


Figure 4: Estimated density of y vs replicated dataset density - participant 1

From the plot we see that the model performed poorly in replicating the density of y values around 5-20, i.e., the replicated data sets had a higher density of values around 5-20 than the observed one. The model's densities of y_{rep} also vary wildly depending on the sample from the posterior predictive distribution as the teal lines vary greatly. This is extremely indicative of a bad fit on the data. So again, it seems the the model is a bad model for the data.

Conclusion for participant 1

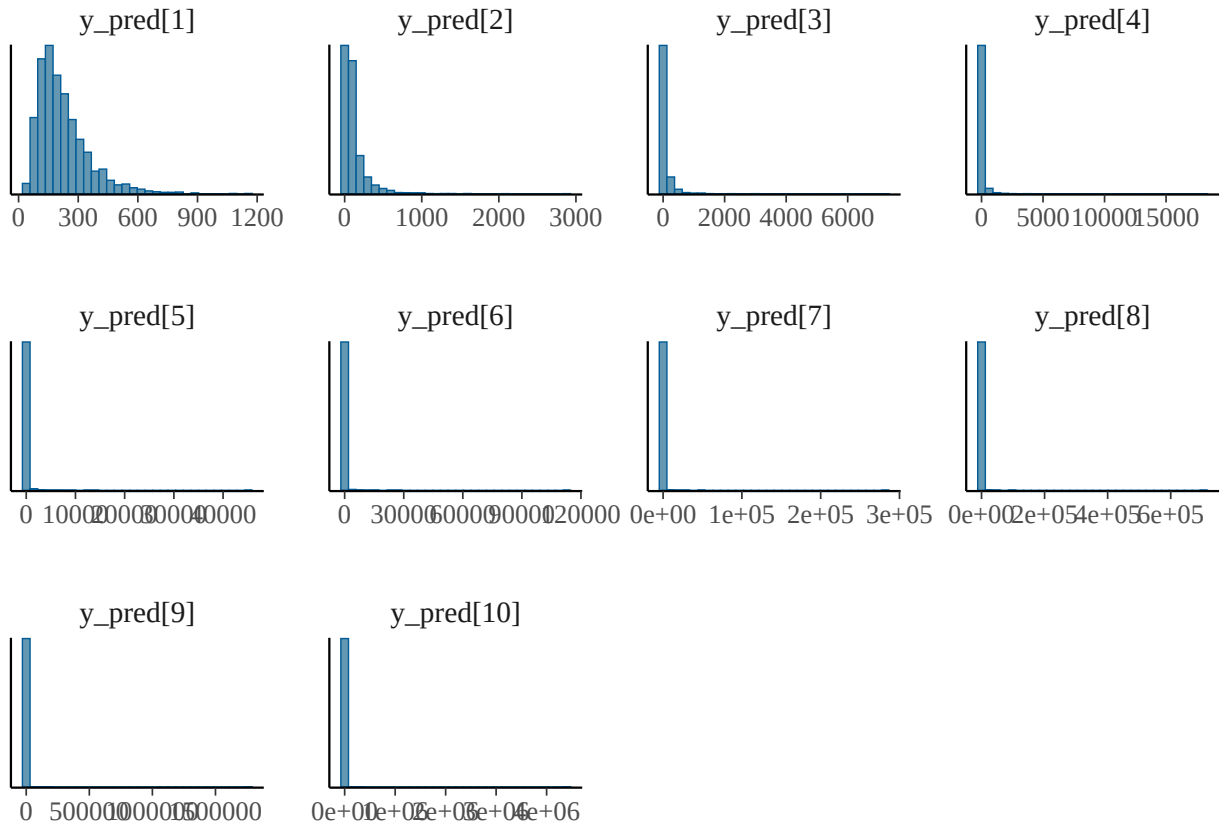
Overall, we believe we have justified priors because it seems that much of the issues regarding prior-data conflicts are due to the model just being a poor model for this data, which is most evident from the time course plot, which managed to cover most of the data points but not the behaviour of the EB counts over

time. The density overlay also very clearly shows that this model is a poor fit for the data. Ignoring potential issues with the model, such as the max ancillary test statistic (which may be due to the model not being able to capture the data's true behaviour), we much our evaluation of the priors such as the prior vs posterior plots and other test statistics indicate behaviours expected from weakly informative priors. We feel comfortable keeping these priors.

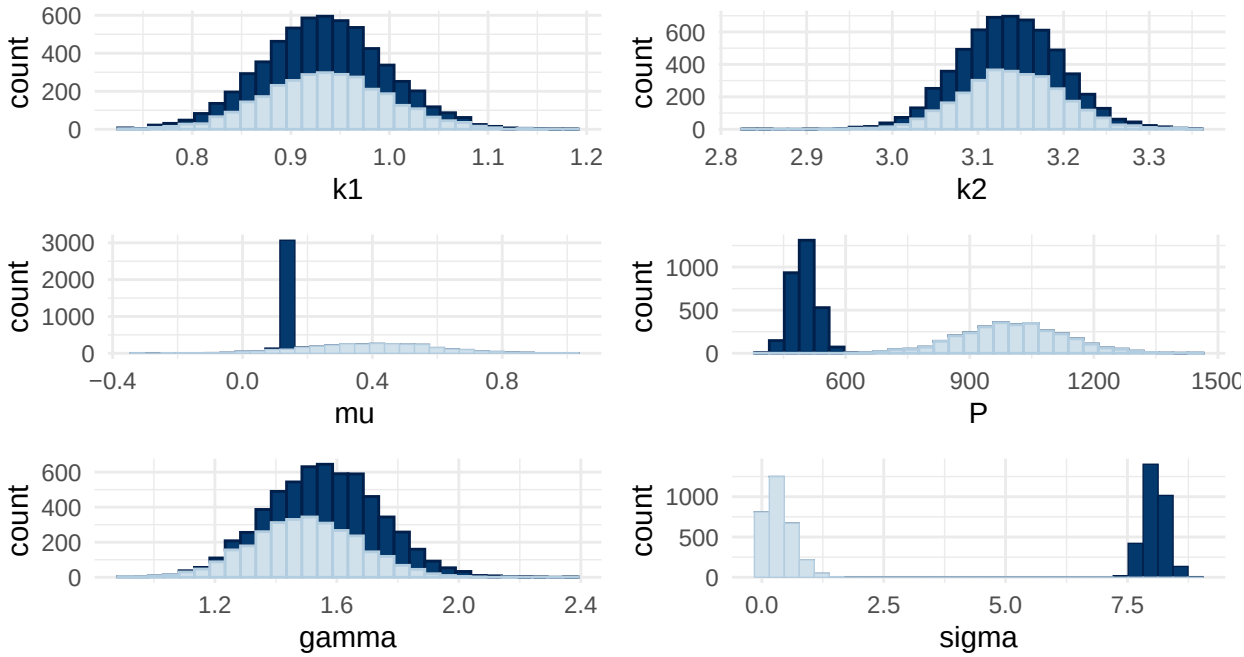
Participant 2: Guinea Pig data with number = 5

Prior predictive checks/Comparison of prior and posterior distributions

We'll again first take a look at the prior predictive distribution:



We see from the histograms across our range of time points (days 3 to 30) that the distribution of EB counts are reasonable with the exception of consistently extremely tails similar to participant 1. Again, the super long tails consist of density that is less than 5% of the total density. The posterior predictive distributions also start out from values we'd expect to see and show that the infection clears up by day around day 30, i.e., from around 250 to around 0.



Next, looking at the plots of the prior distributions versus the posterior distribution for each parameter, we see that the posterior distributions for all parameters contract within the prior distribution, with the exceptions of σ, P. σ as stated in the quick introduction shows substantial evidence of prior-data conflict due to the large distance in between the centres of the prior and posterior distribution; however, the posterior distribution is in a location where there is still density under the prior distribution. Since the location of the posterior distribution of σ is far from 0, we could increase the centre of the prior on σ, but that would be actively making it so that we would get negative y_i values, which are EB counts; thus it doesn't seem sensible to increase the centre of σ. The difference in centres could be due to the model not being a good model for the count of EBs, similar to how using model 2 on homework 1 resulted in prior-data conflict for σ. In homework 1 the data in homework 1 was generated using model 1. The plot for P also shows some signs of prior data conflict but there is clearly some density from the prior distribution's long left tail covering that of the posterior. This is somewhat in contrast to partcipant 1 as there is much greater overlap between the two distributions of P here. We also decided to not change the prior on P because we weren't able to find a set that worked better and was "more justified" as weakly informative priors, which is what we went for in task 1. We will further justify why we kept these priors later on with a time course plot, ancillary test statistics, and a density overlay.

Evaluation of ancillary test statitics

Recall that since we are using normal distributions, reasonable ancillary test statistics are `min`, `max`, and `skewness`.

From the two ancillary test statistics above, we see that the model was able to capture the `min` test statistic quite well while the model barely able to capture the `max` test statistic. The model actually captured the `min` test statistic quite well; the data `min` is almost right at the centre of the $T(y_{rep})$ distribution. We believe that the model was unable to capture the `max` test statistic for reasons similar to participant 1, i.e., most of the data was quite far from the rest of the other data points. So again, this may just be that the model we choose to use, i.e., the differential equation model is a poor model for this data.

We see again that the model was able to capture the skewness test statistic as it's extremely close to the centre of the $T(y_{rep})$ distribution.

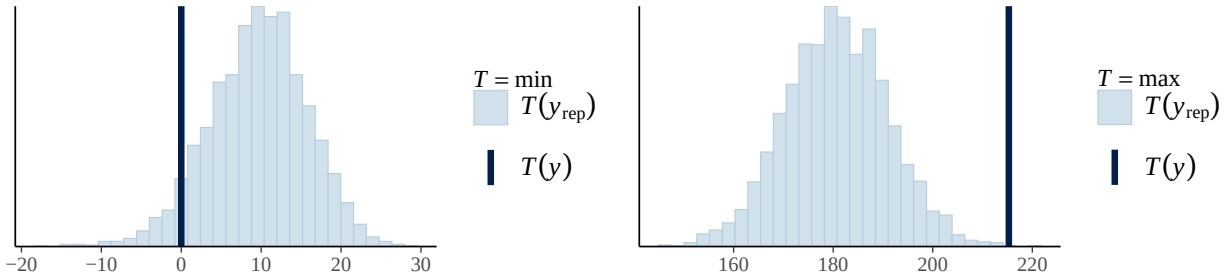


Figure 5: Test statistics plots of min, max - partcipant 2

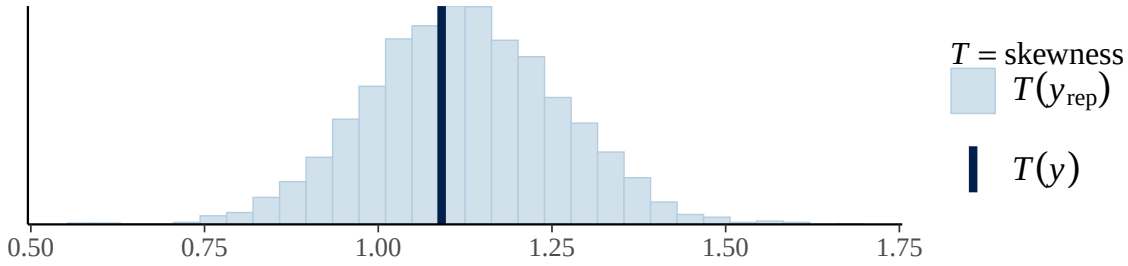


Figure 6: Test statistics plots of sknewness - participant 2

Evaluation of the time course plot

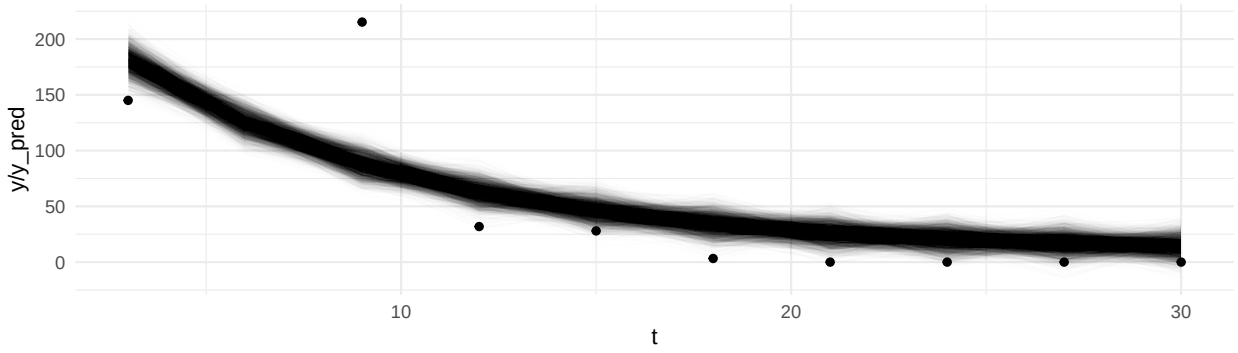


Figure 7: Time course plot - participant 2

Like participant 1, the model was unable to capture the actual behaviour of the data. Instead, this model again fit a downwards sloping trend that went through the "centre" of the data points, sort of similar to model 1 of homework 1. It was unable to capture the peak measurement, but was actually able to capture the majority of the points and have predictions close to the majority of the points. This again could very well be the reason of the prior-data conflict we see above; the model is just a really bad model at capturing the true behaviour of the EB counts. We also, like for participant 1, tried various other combinations of prior values, which didn't change the behaviour of this model that contributed to us keeping these priors.

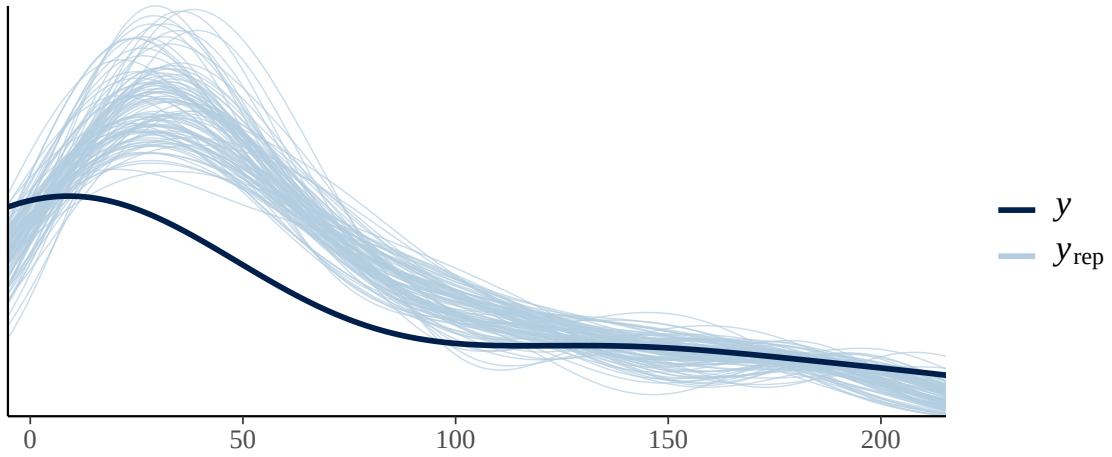


Figure 8: Estimated density of y vs replicated dataset density - participant 2

From the plot we see that the model performed poorly in replicating the density of y values around 0-100, i.e., the replicated data sets had a consistently higher density of values around 0-100 than the observed one. The model's densities of y_{rep} also vary wildly depending on the sample from the posterior predictive distribution as the teal lines vary greatly. This is extremely indicative of a bad fit on the data. So again, it seems the the model is a bad model for the data.

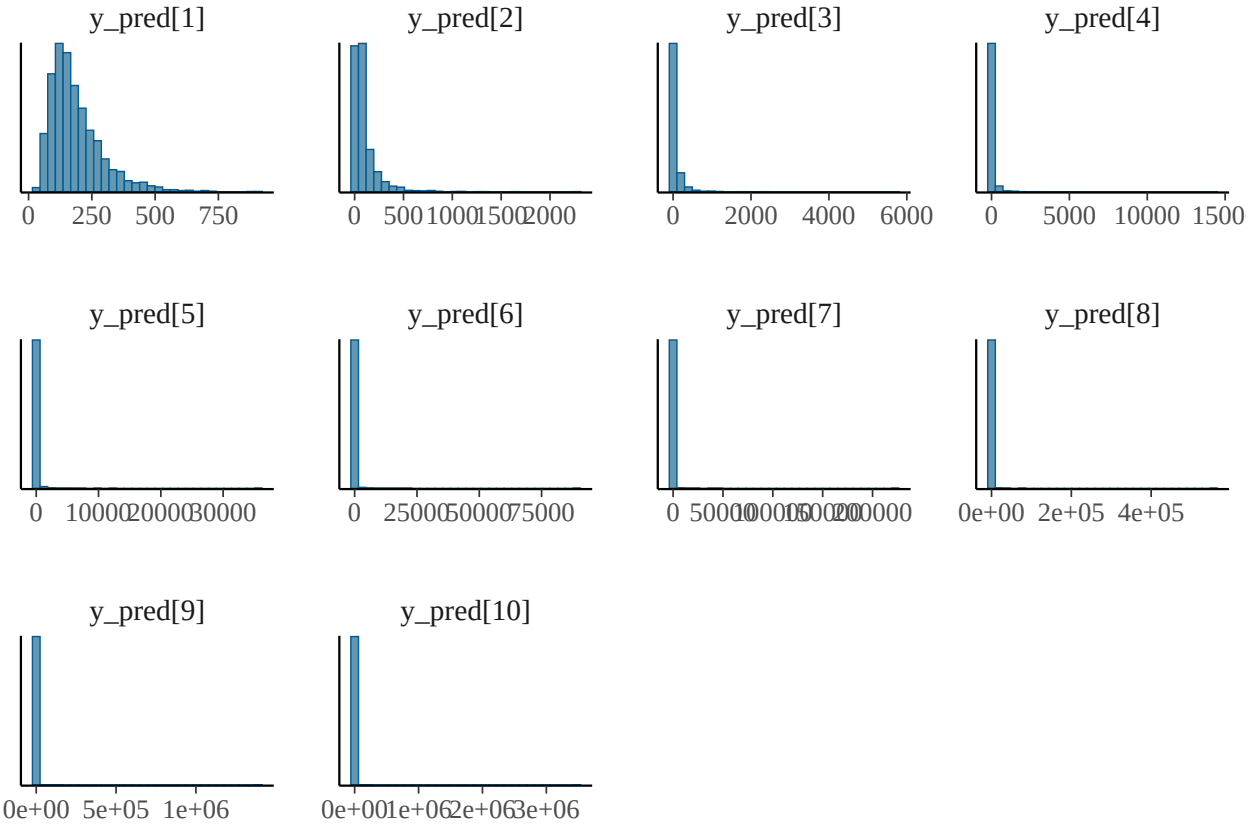
Conclusion for participant 2

Examining participant 2, makes us once again believe we have justified priors because it seems that much of the issues regarding prior-data conflicts are due to the model just being a poor model for this data, which again is most evident from the time course plot, which managed to cover most of the data points but not the behaviour of the EB counts over time. The density overlay also very clearly shows that this model is a poor fit for the data. Ignoring potential issues with the model, such as the max ancillary test statistic (which may be due to the model not being able to capture the data's true behaviour), we much our evaluation of the priors such as the prior vs posterior plots and other test statistics indicate behaviours expected from weakly informative priors. We still feel comfortable keeping these priors.

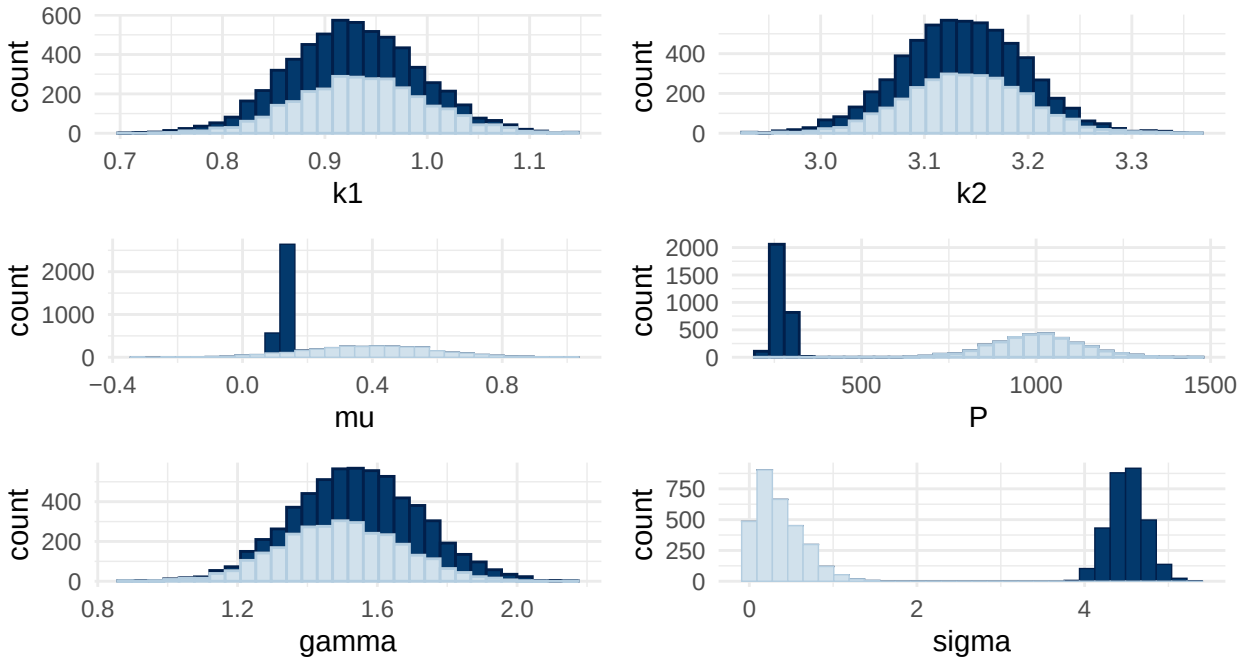
Participant 3: Guinea Pig with number = 7

Prior predictive checks/Comparison of prior and posterior distributions

We'll again first take a look at the prior predictive distribution:



We see from the histograms across our range of time points (days 3 to 30) that the distribution of EB counts are reasonable with the exception of consistently extremely tails similar to all other participants. Again, the super long tails consist of density that is less than 5% of the total density. The posterior predictive distributions also start out from values we'd expect to see and show that the infection clears up by day around day 30, i.e., from around 150 to around 0.



Next, looking at the plots of the prior distributions versus the posterior distribution for each parameter, we see that the posterior distributions for all parameters contract within the prior distribution, with the exceptions of σ, P. σ as stated in the quick introduction shows substantial evidence of prior-data conflict due to the large distance in between the centres of the prior and posterior distribution; however, the posterior distribution is in a location where there is still density under the prior distribution. Since the location of the posterior distribution of σ is far from 0, we could increase the centre of the prior on σ, but that would be actively making it so that we would get negative y_i values, which are EB counts; thus it doesn't seem sensible to increase the centre of σ. The difference in centres could be due to the model not being a good model for the count of EBs, similar to how using model 2 on homework 1 resulted in prior-data conflict for σ. In homework 1 the data in homework 1 was generated using model 1. The plot for P also shows some signs of prior data conflict but there is clearly some density from the prior distribution's long left tail covering that of the posterior, which seems very similar to participant 1. We also decided to not change the prior on P because we weren't able to find a set that worked better and was "more justified" as weakly informative priors, which is what we went for in task 1. We will further justify why we kept these priors later on with a time course plot, ancillary test statistics, and a density overlay.

Evaluation of ancillary test statitics

Recall that since we are using normal distributions, reasonable ancillary test statistics are `min`, `max`, and `skewness`.

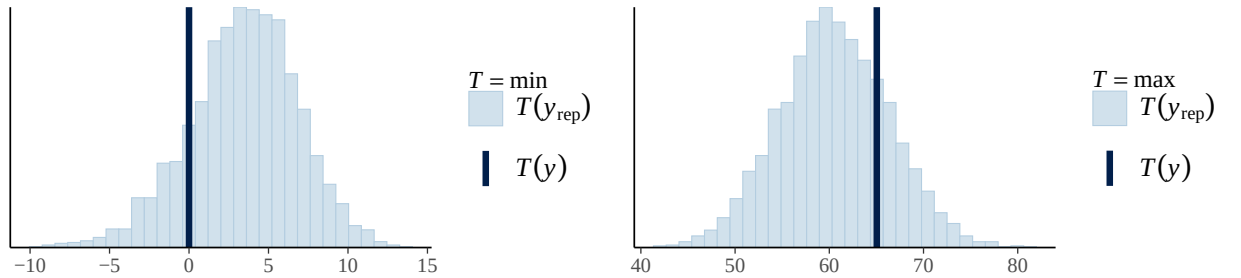


Figure 9: Test statistics plots of min, max - participant 3

From the two ancillary test statistics above, we see that the model was able to capture both test statistics quite well, unlike the previous two participants as we see that the test statistic in both distributions are close to the centre of the two distributions. This is likely due to the data for this participant having low variance.

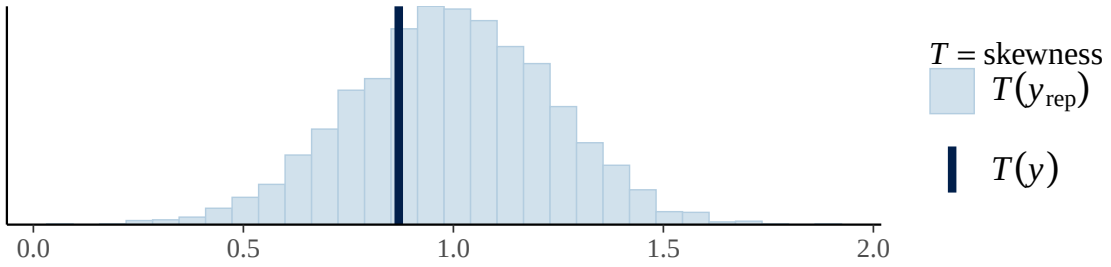


Figure 10: Test statistics plots of sknewness - participant 3

We see again that the model was fairly well able to capture the skewness test statistic as it's extremely close to the centre of the $T(y_{rep})$ distribution. So all 3 test statistics were well captured by this model.

Evaluation of the time course plot

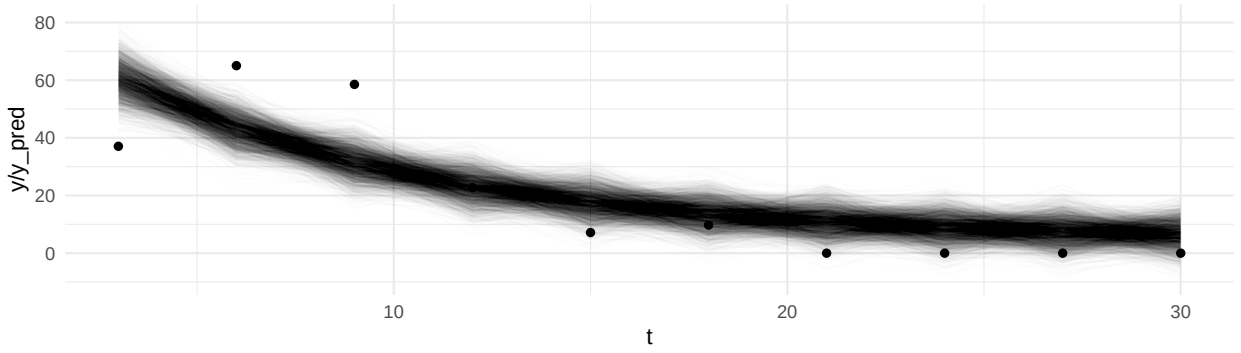


Figure 11: Time course plot - participant 3

Like the previous two participants, the model was unable to capture the actual behaviour of the data. Instead, this model again fit a downwards sloping trend that went through the "centre" of the data points, sort of similar to model 1 of homework 1. It was unable to capture the peak measurement, but was actually able to capture the majority of the points and have predictions close to the majority of the points. This again could very well be the reason of the prior-data conflict we see above; the model is just a really bad model at capturing the true behaviour of the EB counts. We also,like for the other participants, tried various other combinations of prior values, which didn't change the behaviour of this model that contributed to us keeping these priors.

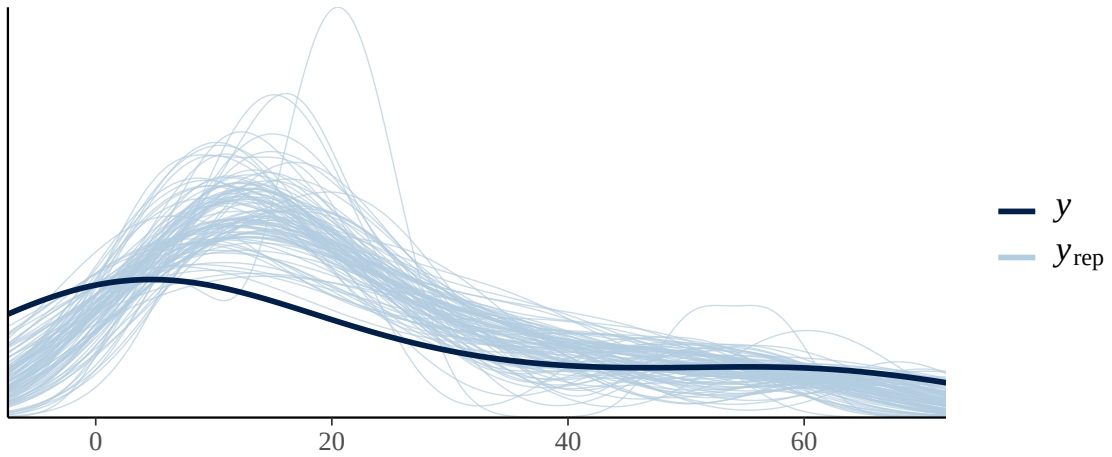


Figure 12: Estimated density of y vs replicated dataset density - participant 3

From the plot we see that the model performed poorly in replicating the density of y values around 0-30, i.e., the replicated data sets had a consistently higher density of values around 0-30 than the observed one. The model's densities of y_{rep} also vary wildly depending on the sample from the posterior predictive distribution as the teal lines vary greatly. This is extremely indicative of a bad fit on the data. So again, it seems the the model is a bad model for the data.

Conclusion for participant 3

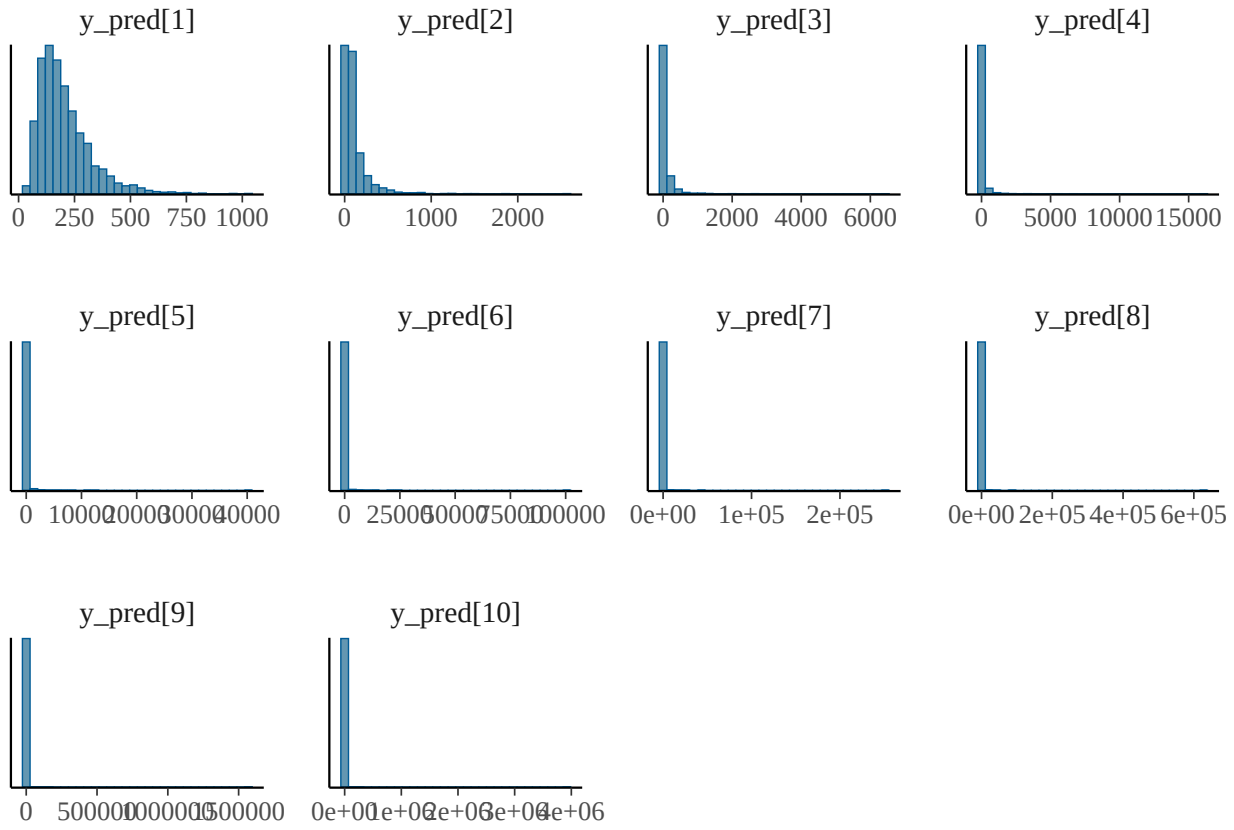
Examining participant 3 still makes us believe we have justified priors. It seems that much of the issues regarding prior-data conflicts are due to the model just being a poor model for this data, which again is most evident from the time course plot that managed to cover most of the data points but not the behaviour of

the EB counts over time, similar to our analysis for the participants 1 and 2. The density overlay also very clearly shows that this model is a poor fit for the data. With participant 3, much of our evaluation of the priors such as the prior vs posterior plots and other test statistics indicate behaviours expected from weakly informative priors. We still feel comfortable keeping these priors; especially because with so many of the posterior predictive checks checking out and because **the behaviour of the model seems to always fail to capture the true behaviour**, we believe this model is just simply a bad model.

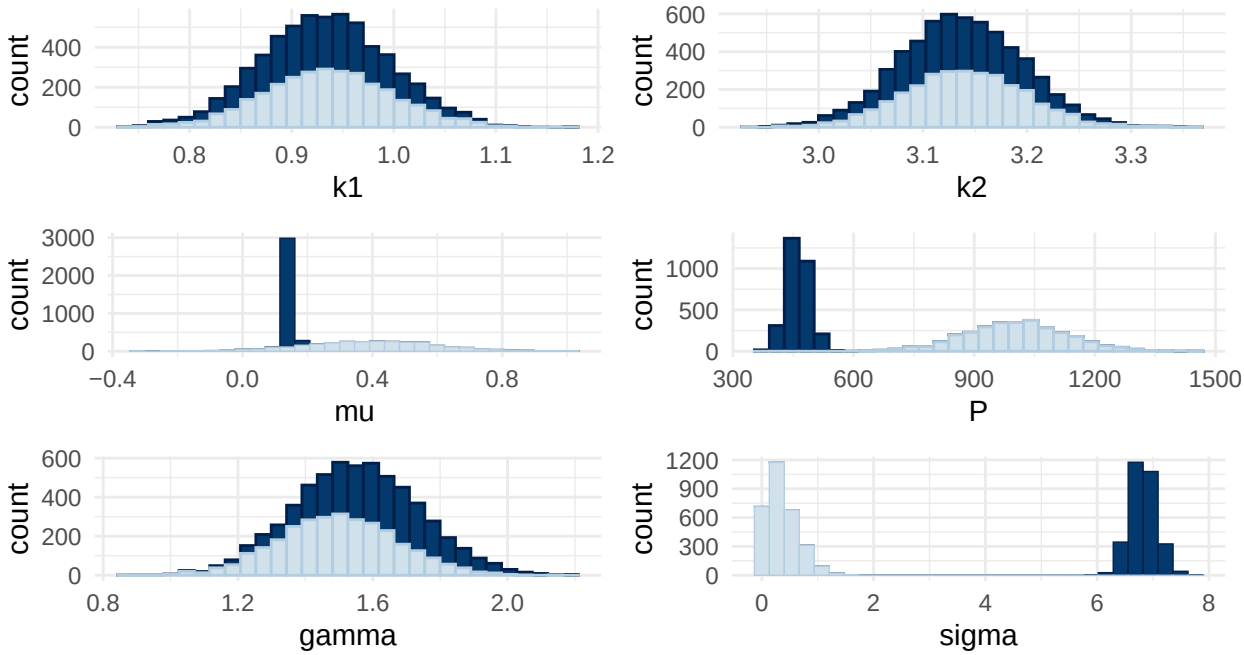
Participant 4: Guinea Pig with number = 11

Prior predictive checks/Comparison of prior and posterior distributions

We'll again first take a look at the prior predictive distribution:



We see from the histograms across our range of time points (days 3 to 30) that the distribution of EB counts are reasonable with the exception of consistently extremely tails similar to participant 1. Again, the super long tails consist of density that is less than 5% of the total density. The posterior predictive distributions also start out from values we'd expect to see and show that the infection clears up by day around day 30, i.e., from around 175 to around 0.



Next, looking at the plots of the prior distributions versus the posterior distribution for each parameter, we see that the posterior distributions for all parameters contract within the prior distribution, with the exceptions of σ, P. σ as stated in the quick introduction shows substantial evidence of prior-data conflict due to the large distance in between the centres of the prior and posterior distribution; however, the posterior distribution is in a location where there is still density under the prior distribution. Since the location of the posterior distribution of σ is far from 0, we could increase the centre of the prior on σ, but that would be actively making it so that we would get negative y_i values, which are EB counts; thus it doesn't seem sensible to increase the centre of σ. The difference in centres could be due to the model not being a good model for the count of EBs, similar to how using model 2 on homework 1 resulted in prior-data conflict for σ. In homework 1 the data in homework 1 was generated using model 1. The plot for P also shows some signs of prior data conflict but there is clearly some density from the prior distribution's long left tail covering that of the posterior. This is somewhat in contrast to partcipant 1 as there is much greater overlap between the two distributions of P here. We also decided to not change the prior on P because we weren't able to find a set that worked better and was "more justified" as weakly informative priors, which is what we went for in task 1. We will further justify why we kept these priors later on with a time course plot, ancillary test statistics, and a density overlay.

Evaluation of ancillary test statitics

Recall that since we are using normal distributions, reasonable ancillary test statistics are `min`, `max`, and `skewness`.

From the two ancillary test statistics above, we see that the model was able to capture the `min` test statistic quite well while the model was somewhat able to capture the `max` test statistic. The model actually captured the `min` test statistic quite well; the data `min` is almost right at the centre of the $T(y_{rep})$ distribution. We believe that the model was unable to capture the `max` test statistic for reasons similar to participants 1 and 2, i.e., most of the data was quite far from the rest of the other data points. So again, this may just be that the model we choose to use, i.e., the differential equation model is a poor model for this data.

We see again that the model was fairly well able to capture the skewness test statistic as it's close to the centre of the $T(y_{rep})$ distribution.

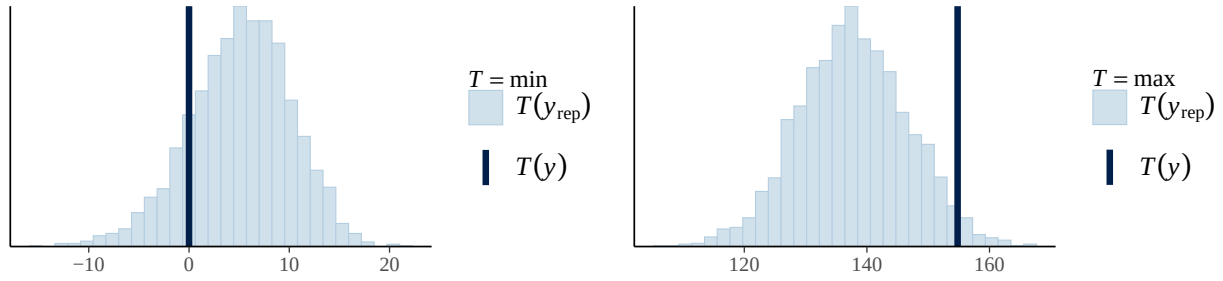


Figure 13: Test statistics plots of min, max - participant 4

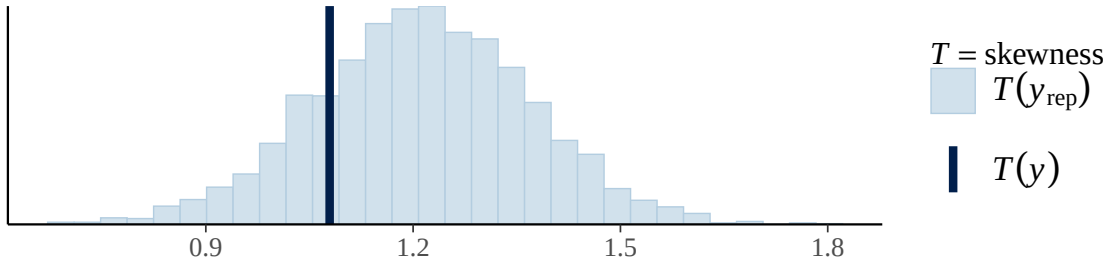


Figure 14: Test statistics plots of sknewness - participant 4

Evaluation of the time course plot

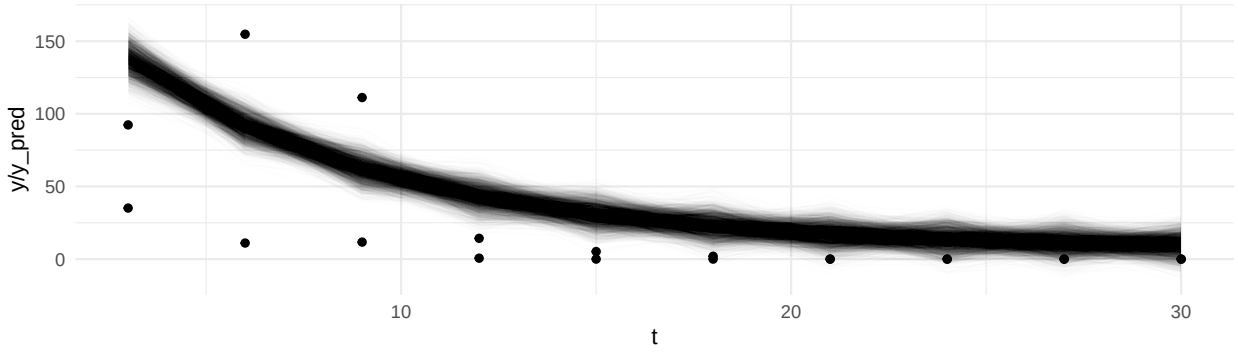


Figure 15: Time course plot - participant 4

Like every other participant, the model was unable to capture the actual behaviour of the data. Instead, this model again fit a downwards sloping trend that went through the "centre" of the data points, sort of similar to model 1 of homework 1. It was unable to capture the peak measurement, but was actually able to capture the majority of the points and have predictions close to the majority of the points. This again could very well be the reason of the prior-data conflict we see above; the model is just a really bad model at capturing the true behaviour of the EB counts. We also tried various other combinations of prior values, which didn't change the behaviour of this model that contributed to us keeping these priors.

From the plot we see that the model performed poorly in replicating the density of y values around 0-60, i.e., the replicated data sets had a consistently higher density of values around 0-60 than the observed one. The model's densities of y_{rep} also vary wildly depending on the sample from the posterior predictive distribution as the teal lines vary greatly. This is extremely indicative of a bad fit on the data. So again, it seems the the model is a bad model for the data.

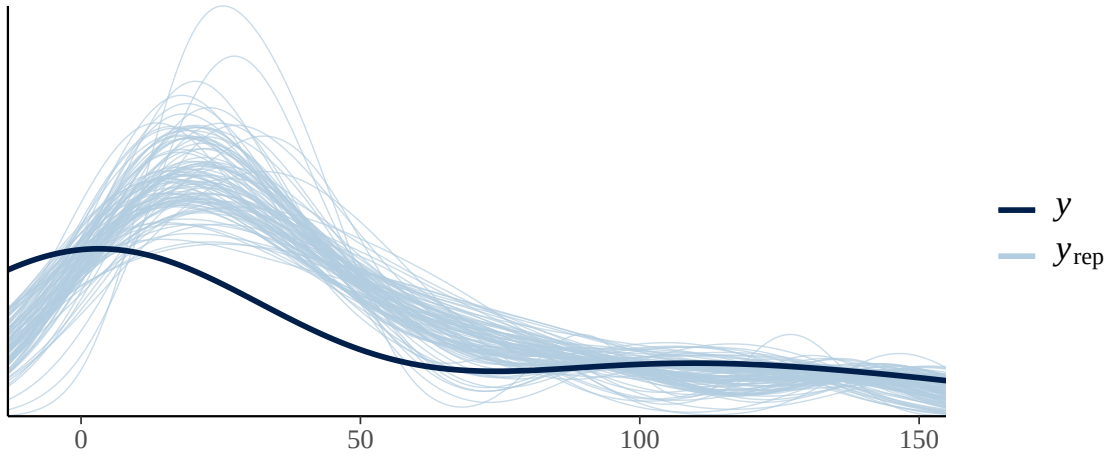


Figure 16: Estimated density of y vs replicated dataset density - participant 4

Conclusion for participant 4

Analysing participant 4 still leads us to believe we have justified priors because it seems that much of the issues regarding prior-data conflicts are due to the model just being a poor model for this data, which again is most evident from the time course plot, which managed to cover most of the data points but not the behaviour of the EB counts over time. The density overlay also very clearly shows that this model is a poor fit for the data. Ignoring potential issues with the model, such as the max ancillary test statistic (which may be due to the model not being able to capture the data's true behaviour), we much our evaluation of the priors such as the prior vs posterior plots and other test statistics indicate behaviours expected from weakly informative priors. We still feel comfortable keeping these priors.

Task 3: How many EBs are passed sexually.

We need to average the conditional posterior $p(C(0)|y, P, k_1, k_2, \gamma, \mu, \sigma)$ over the joint posterior $p(P, k_1, k_2, \gamma, \mu, \sigma|y)$ through multiple imputation.

First, we'll specify a prior distribution on $C(0)$; we'll assume a normal distribution by the CLT. Since we want to evaluate the statement in the task description, we'll set μ for this prior to be $10^2 \cdot 10^{-4}$. It is also the only information we know about $C(0)$ with respect to sexual infection. Since we know that $C(0)$ should be non-negative, we set τ to be $4e2 * 1e - 5$ because $\mu - 2 \cdot (4e2 * 1e - 5)$ should still be greater than 0.

```
draws_df <- as_draws_df(posterior_p4)
random_row <- sample(1:nrow(draws_df), 1)
days_data_raw <- sexual %>% arrange(t) %>% distinct(t)
days_data_conv <- days_data_raw %>% mutate(t = ((t - seq_start)/seq_timestep + 1)) %>% pull(t)
participant_data_sexual <- sexual %>% select(c(t, C, number))
%>% arrange(t, number) %>%pivot_wider(names_from = number, values_from = C)
%>% select(-c(t))%>%pull("11")
pred_times_t3<- sexual %>% select(t) %>% pull(t)
pred_times_seq_t3 <- seq(0.1, max(pred_times_t3), by = 0.01)
```

```
C0_list = list()
for(i in 1:100){
  # draw a row of parameters from the posterior
  sampled_row <- draws_df[random_row, ]
  conditional_params <- list(
      participants = 1,
      n_days = length(pred_times_seq),
      pred_times = pred_times_seq,
      initial_cell_counts = c(9600*1e-4, 0.0),
      n_measurements = sexual %>% select(t) %>% n_distinct(),
      days_data = days_data_conv,
      y = participant_data_sexual,
      Pe = 40 * 1e-4,
      delta_e = 2,
      k1 = sampled_row %>% pull(k1),
      k2 = sampled_row %>% pull(k2),
      P = sampled_row %>% pull(P),
      mu_param = sampled_row %>% pull(mu_param),
      gamma = sampled_row %>% pull(gamma),
      sigma = sampled_row %>% pull(sigma),
      c0_mu = 1e2*1e-4,
      c0_tau = 4e2*1e-5,
      only_prior = 0
    )
  multi_imputation_fit <- imputation_model$
    sample(conditional_params,seed = 365,
           refresh = 1000, parallel_chains = 4 ,
           iter_warmup = 1000, iter_sampling=200)
  posterior_df <- as_draws_df(multi_imputation_fit$draws())
  # draw a value of C(0)
  random_row <- sample(1:nrow(posterior_df), 1)
  append(C0_list, posterior_df[random_row, ])
  # get a new random row to prep for drawing from posterior
  random_row <- sample(1:nrow(draws_df), 1)
}
```

We were unable to run this code in a reasonable amount of time (2+ hours), and it didn't seem like there were any coding errors, so the rest of this task will be based on my background knowledge. The stan code is in the appendix.

Critical assessment of multiple imputation

Drawbacks

1. This methods requires putting a prior on the quantity you wish to generate distribution for. Then, you need a sensible prior, which may require significant domain knowledge, experience in modelling that kind of data such as epidemiological data. So this is different than black box methods like neural networks or boosted trees, where it's easy to tune a model despite having little domain knowledge. In particular, putting a completely uninformative prior on $C(0)$ would be ill advised, e.g., you may get many negative values for $C(0)$.
2. This takes an extremely long amount of time. On a modern CPU this algorithm was unable to finish. MCMC sampling is still extremely slow compared to other methods and you need to finish running the chain in order to get an acceptable `ess_bulk` size. On the contrary, other time consuming ML/Statistical learning methods such as neural networks, which can take a notoriously long time to fit, are still able to early stop because you are simply shift around weights/coefficients. This isn't a perfect comparsion as neural networks are often frequentist, but this algorithm takes a frankly crazy long amount of time.

Pros

1. I can't tell empirically what happens with multiple imputation from the algorithm I wrote, but from CSC311 and personal readings, I know that imputation allows for us to fill in missing values using some algorithm. This is powerful because a lot of the time data is dirty or missing, so imputation would allow for us to get a cleaner data set by imputing missing or bad values with plausible values.
2. Bayesian multiple imputation may be uniquely powerful because it allows us to construct a posterior distribution for various given parameters. In comparison this could lead to more sensible imputed values than say imputation using kNN. Over the course, we've seen how adding a prior, such as a weakly informative prior can have regularizing properties, which is can be a drawback to MLE based methods.

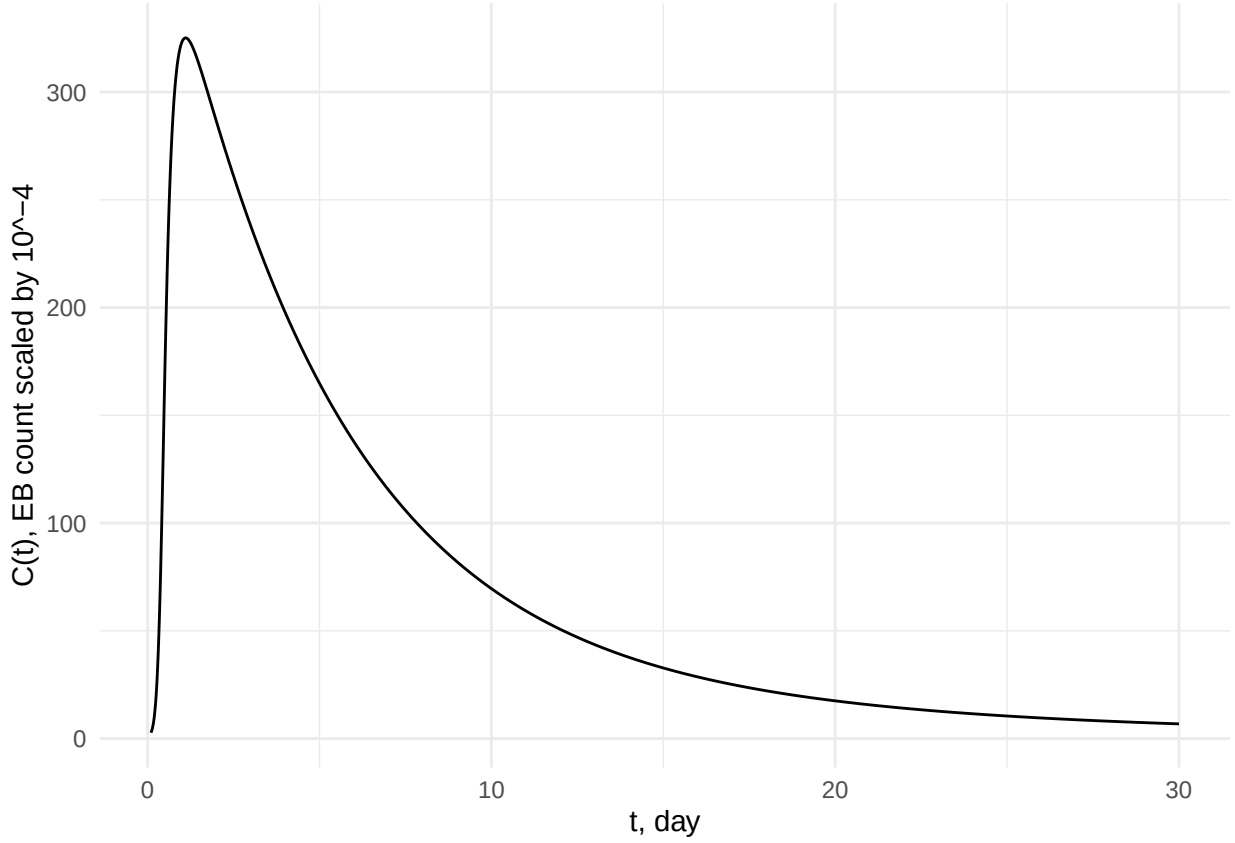
Discussion of Rank et al.

I can't say for certain what the distribution would look like for $C(0)$, but I would expect it to be somewhat similar to a distribution constructed using MAP estimates of parameters of the $C(0)$ posterior distribution. I'm honestly pretty bummed out that this wasn't able to finish in a reasonable amount of time :(

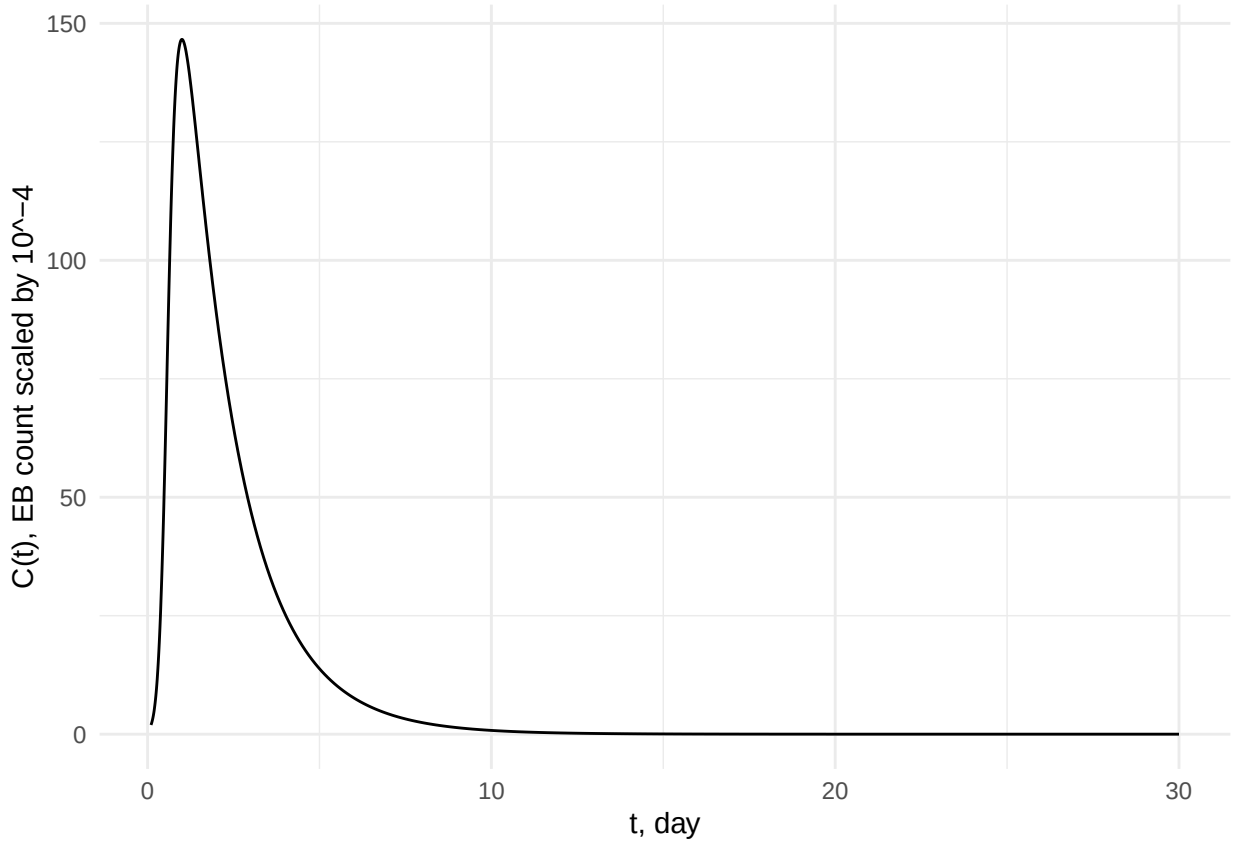
Appendix

Task 1 plots

High values of y given C(0) = 1



Low values of y given C(0) = 1



R code

Code used in rmd

```
kable(head(data %>% filter(dose != "sexual")))

kable(head(data %>% filter(dose == "sexual")))


opencl_options <- list(
  stan_opencl = TRUE,
  opencl_platform_id = 0, # replace the ID based on step 3
  opencl_device_id = 0,
  ldflags_opencl = "-L'C:\\Program Files\\NVIDIA GPU Computing Toolkit\\CUDA\\v11.2\\lib\\x64'"
)
param_model <- cmdstan_model("a1_prior_exploration.stan", compile = TRUE)

ini_cell_counts <- c(E = 9600 * 1e-4, C = 1e1 * 1e-4, I = 0)
pred_times <- data %>%
  filter(number == 2) %>%
  select(t) %>%
  pull(t)
# pred_times <- pred_times %>% insert(ats = 1, values = c(0.0))
pred_times_seq <- seq(0.1, max(pred_times), by = 0.01)
# length(pred_times_seq)
```

```r
# ini_cell_counts['E']

test_params_default <- list(
  P = 1000,
  k1 = 0.1 / 1e-4,
  k2 = 0.8,
  Pe = 40 * 1e-4,
  delta_e = 2.0,
  gamma = 1.2,
  mu_param = 1.2,
  initial_cell_counts = ini_cell_counts,
  pred_times = pred_times_seq,
  n_days = length(pred_times_seq)
)
# test_params_default$pred_times
test_params_default_ct <- param_model$sample(test_params_default,
  seed = 365,
  refresh = 0, fixed_param = T, chains = 1, iter_warmup = 10, iter_sampling = 1
)

cs <- test_params_default_ct$draws() %>%
  reshape2::melt() %>%
  dplyr::filter(str_detect(variable, "C")) %>%
  tidyr::extract(
    col = variable, into = "ind",
    regex = "C\\[([0-9]*)\\]",
    convert = TRUE
  ) %>%
  dplyr::mutate(
    time = test_params_default$pred_times[ind],
    chain_iter = glue::glue("chain {chain}, iteration {iteration}"),
    .keep = "unused"
  ) %>%
  rename(mu = value) %>%
  mutate(cell_type = rep("c", test_params_default$n_days))


cs %>% ggplot(aes(x = time, y = mu)) +
  geom_line(alpha = 1) +
  theme_minimal() +
  labs(x = "t: day", y = "C(t): EB count scaled by 10^-4")

test_params_hi <- list(
  P = 1000 * 1.125,
  k1 = 0.00001 / 1e-4,
  k2 = 0.8 * 4,
  Pe = 40 * 1e-4,
  delta_e = 2,
  gamma = 1.2 * 1.1,
  mu_param = 1.2 * 0.165,
  initial_cell_counts = ini_cell_counts,
  pred_times = pred_times_seq,
  n_days = length(pred_times_seq)
```

```
)
test_params_hi_ct <- param_model$sample(test_params_hi, seed = 365, refresh = 0, fixed_param = T, chains

cs <- test_params_hi_ct$draws() %>%
  reshape2::melt() %>%
  dplyr::filter(str_detect(variable, "C")) %>%
  tidyr::extract(
    col = variable, into = "ind",
    regex = "C\\[([0-9]*)\\]",
    convert = TRUE
  ) %>%
  dplyr::mutate(
    time = test_params_hi$pred_times[ind],
    chain_iter = glue::glue("chain {chain}, iteration {iteration}"),
    .keep = "unused"
  ) %>%
  rename(mu = value) %>%
  mutate(cell_type = rep("c", test_params_hi$n_days))




cs %>% ggplot(aes(x = time, y = mu)) +
  geom_line(alpha = 1) +
  theme_minimal() +
  labs(x = "t, day", y = "C(t), EB count scaled by 10^-4")

test_params_lo <- list(
  P = 1000 * 0.875,
  k1 = 0.00000875 / 1e-4,
  k2 = 0.8 * 3.85,
  Pe = 40 * 1e-4,
  delta_e = 2,
  gamma = 1.2 * 1.45,
  mu_param = 1.2 * 0.55,
  initial_cell_counts = ini_cell_counts,
  pred_times = pred_times_seq,
  n_days = length(pred_times_seq)
)
test_params_lo_ct <- param_model$sample(test_params_lo, seed = 365, refresh = 0, fixed_param = T, chains


cs <- test_params_lo_ct$draws() %>%
  reshape2::melt() %>%
  dplyr::filter(str_detect(variable, "C")) %>%
  tidyr::extract(
    col = variable, into = "ind",
    regex = "C\\[([0-9]*)\\]",
    convert = TRUE
  ) %>%
  dplyr::mutate(
    time = test_params_lo$pred_times[ind],
    chain_iter = glue::glue("chain {chain}, iteration {iteration}"),
```

```
    .keep = "unused"
  ) %>%
  rename(mu = value) %>%
  mutate(cell_type = rep("c", test_params_lo$n_days))




cs %>% ggplot(aes(x = time, y = mu)) +
  geom_line(alpha = 1) +
  theme_minimal() +
  labs(x = "t, day", y = "C(t), EB count scaled by 10^-4")

kable(head(cs %>% arrange(desc(time))))

opencl_options <- list(
  stan_opencl = TRUE,
  opencl_platform_id = 0,
  opencl_device_id = 0,
  ldflags_opencl = "-L'C:\\Program Files\\NVIDIA GPU Computing Toolkit\\CUDA\\v11.2\\lib\\x64'"
)
ode_model <- cmdstan_model("a1_single.stan", compile = TRUE, threads = 12)

participant_data <- non_sexual %>%
  select(c(t, C, number)) %>%
  arrange(t, number) %>%
  pivot_wider(names_from = number, values_from = C) %>%
  select(-c(t))
EB_counts_vec <- c(1e1 * 1e-4, 1e4 * 1e-4, 1e2 * 1e-4, 1e3 * 1e-4)
cell_counts <- matrix(NA, nrow = 4, ncol = 3)
cell_counts[1, ] <- c(9600 * 1e-4, 1e1 * 1e-4, 0.0)
cell_counts[2, ] <- c(9600 * 1e-4, 1e4 * 1e-4, 0.0)
cell_counts[3, ] <- c(9600 * 1e-4, 1e2 * 1e-4, 0.0)
cell_counts[4, ] <- c(9600 * 1e-4, 1e3 * 1e-4, 0.0)
seq_start <- 0.1
seq_timestep <- 0.05
pred_times_seq <- seq(seq_start, max(pred_times), by = seq_timestep)
num_measurements <- non_sexual %>%
  select(t) %>%
  n_distinct()
days_data_raw <- non_sexual %>%
  arrange(t) %>%
  distinct(t)
days_data_conv <- days_data_raw %>%
  mutate(t = ((t - seq_start) / seq_timestep + 1)) %>%
  pull(t)

data_list_single <- list(
  participants = 1,
  n_days = length(pred_times_seq),
  pred_times = pred_times_seq,
  initial_cell_counts = c(9600 * 1e-4, 1e1 * 1e-4, 0.0),
  n_measurements = non_sexual %>% select(t) %>% n_distinct(),
```

```
  days_data = days_data_conv,
  y = participant_data[, 1] %>% pull("2"),
  Pe = 40 * 1e-4,
  delta_e = 2,
  k1_mu = 0.9375,
  k1_tau = 0.0625,
  k2_mu = 3.14,
  k2_tau = 0.06,
  P_mu = 1000,
  P_tau = 125,
  mu_param_mu = 0.399,
  mu_param_tau = 0.201,
  gamma_mu = 1.5,
  gamma_tau = 0.18,
  sigma_tau = 4.5e-1,
  only_prior = 1
)

model_prior_fit_p1 <- ode_model$sample(data_list_single, seed = 365, refresh = 0, chains = 3, parallel_

model_prior_fit_p1$summary()

data_list_single$only_prior <- 0
model_fit_p1 <- ode_model$sample(data_list_single, seed = 365, refresh = 1500, chains = 3, parallel_cha

prior_p1 <- model_prior_fit_p1$draws()
posterior_p1 <- model_fit_p1$draws()

model_fit_p1$summary()

ypreds_p1 <- posterior_p1 %>%
  reshape2::melt() %>%
  filter(str_detect(variable, "y_pred")) %>%
  tidyr::extract(
    col = variable, into = "ind",
    regex = "y_pred\\[([0-9]*)\\]",
    convert = TRUE
  )
ypreds_p1 <- ypreds_p1 %>%
  mutate(
    time = data_list_single$days_data[ind],
    chain_iter = glue::glue("chain {chain}, iteration {iteration}"),
    .keep = "unused"
  ) %>%
  rename(y_pred = value)
ypreds_p1 <- ypreds_p1 %>% mutate(time = (time + 1) * seq_timestep)


k1_posterior_p1 <- melt(as_draws_matrix(subset_draws(model_fit_p1$draws(), regex = TRUE, variable = "k1
k1_prior_p1 <- melt(as_draws_matrix(subset_draws(model_prior_fit_p1$draws(), regex = TRUE, variable = "

k1_p1_comparison_df <- rbind(k1_prior_p1, k1_posterior_p1)
```

```r
k1_p1_plt <- ggplot(k1_p1_comparison_df, aes(x = value, fill = variable, color = variable)) +
  geom_histogram(alpha = 1) +
  scale_fill_manual(values = c(color_scheme_get()$dark, color_scheme_get()$light)) +
  theme_minimal() +
  theme(legend.position = "none") +
  labs(x = "k1, Guinea Pig number 2") +
  scale_color_manual(values = c(color_scheme_get()$dark_highlight, color_scheme_get()$light_highlight))

k2_posterior_p1 <- melt(as_draws_matrix(subset_draws(model_fit_p1$draws(), regex = TRUE, variable = "k2
k2_prior_p1 <- melt(as_draws_matrix(subset_draws(model_prior_fit_p1$draws(), regex = TRUE, variable = "

k2_p1_comparison_df <- rbind(k2_prior_p1, k2_posterior_p1)

k2_p1_plt <- ggplot(k2_p1_comparison_df, aes(x = value, fill = variable, color = variable)) +
  geom_histogram(alpha = 1) +
  scale_fill_manual(values = c(color_scheme_get()$dark, color_scheme_get()$light)) +
  scale_color_manual(values = c(color_scheme_get()$dark_highlight, color_scheme_get()$light_highlight))
  theme_minimal() +
  theme(legend.position = "none") +
  labs(x = "k2, Guinea Pig number 2")
# comparison_prior_df <- comparison_prior_df  %>% mutate(variable = str_replace_all(comparison_prior_df

k2_posterior_p1 <- melt(as_draws_matrix(subset_draws(model_fit_p1$draws(), regex = TRUE, variable = "k2
k2_prior_p1 <- melt(as_draws_matrix(subset_draws(model_prior_fit_p1$draws(), regex = TRUE, variable = "

k2_p1_comparison_df <- rbind(k2_prior_p1, k2_posterior_p1)

k2_p1_plt <- ggplot(k2_p1_comparison_df, aes(x = value, fill = variable, color = variable)) +
  geom_histogram(alpha = 1) +
  scale_fill_manual(values = c(color_scheme_get()$dark, color_scheme_get()$light)) +
  theme_minimal() +
  theme(legend.position = "none") +
  labs(x = "k2, Guinea Pig number 2") +
  scale_color_manual(values = c(color_scheme_get()$dark_highlight, color_scheme_get()$light_highlight))



gamma_p1_posterior <- melt(as_draws_matrix(subset_draws(model_fit_p1$draws(), regex = TRUE, variable =
gamma_p1_prior <- melt(as_draws_matrix(subset_draws(model_prior_fit_p1$draws(), regex = TRUE, variable

gamma_p1_comparison_df <- rbind(gamma_p1_posterior, gamma_p1_prior)
# comparison_prior_df <- comparison_prior_df  %>% mutate(variable = str_replace_all(comparison_prior_df

gamma_p1_plt <- ggplot(gamma_p1_comparison_df, aes(x = value, fill = variable, color = variable)) +
  geom_histogram(alpha = 1) +
  scale_fill_manual(values = c(color_scheme_get()$dark, color_scheme_get()$light)) +
  theme_minimal() +
  theme(legend.position = "none") +
  labs(x = "gamma") +
  scale_color_manual(values = c(color_scheme_get()$dark_highlight, color_scheme_get()$light_highlight))


P_posterior_p1 <- melt(as_draws_matrix(subset_draws(model_fit_p1$draws(), regex = TRUE, variable = "P"))
```

```r
P_prior_p1 <- melt(as_draws_matrix(subset_draws(model_prior_fit_p1$draws(), regex = TRUE, variable = "P

P_p1_comparison_df <- rbind(P_posterior_p1, P_prior_p1)
# comparison_prior_df <- comparison_prior_df  %>% mutate(variable = str_replace_all(comparison_prior_df

P_p1_plt <- ggplot(P_p1_comparison_df, aes(x = value, fill = variable, color = variable)) +
  geom_histogram(alpha = 1) +
  scale_fill_manual(values = c(color_scheme_get()$dark, color_scheme_get()$light)) +
  theme_minimal() +
  theme(legend.position = "none") +
  labs(x = "P") +
  scale_color_manual(values = c(color_scheme_get()$dark_highlight, color_scheme_get()$light_highlight))


mu_posterior_p1 <- melt(as_draws_matrix(subset_draws(model_fit_p1$draws(), regex = TRUE, variable = "mu
mu_prior_p1 <- melt(as_draws_matrix(subset_draws(model_prior_fit_p1$draws(), regex = TRUE, variable = "

mu_p1_comparison_df <- rbind(mu_posterior_p1, mu_prior_p1)
# comparison_prior_df <- comparison_prior_df  %>% mutate(variable = str_replace_all(comparison_prior_df

mu_p1_plt <- ggplot(mu_p1_comparison_df, aes(x = value, fill = variable, color = variable)) +
  geom_histogram(size = 0.25) +
  scale_fill_manual(values = c(color_scheme_get()$dark, color_scheme_get()$light)) +
  theme_minimal() +
  theme(legend.position = "none") +
  labs(x = "mu") +
  scale_color_manual(values = c(color_scheme_get()$dark_highlight, color_scheme_get()$light_highlight))

sigma_posterior_p1 <- melt(as_draws_matrix(subset_draws(model_fit_p1$draws(), regex = TRUE, variable =
sigma_prior_p1 <- melt(as_draws_matrix(subset_draws(model_prior_fit_p1$draws(), regex = TRUE, variable

sigma_p1_comparison_df <- rbind(sigma_posterior_p1, sigma_prior_p1)
# comparison_prior_df <- comparison_prior_df  %>% mutate(variable = str_replace_all(comparison_prior_df

sigma_p1_plt <- ggplot(sigma_p1_comparison_df, aes(x = value, fill = variable, color = variable)) +
  geom_histogram(alpha = 1, size = 0.25) +
  scale_fill_manual(values = c(color_scheme_get()$dark, color_scheme_get()$light)) +
  theme_minimal() +
  theme(legend.position = "none") +
  labs(x = "sigma") +
  scale_color_manual(values = c(color_scheme_get()$dark_highlight, color_scheme_get()$light_highlight))

p1_prior_pred_draws <- model_prior_fit_p1$draws(c("y_pred[1]", "y_pred[2]", "y_pred[3]", "y_pred[4]", "
mcmc_hist(p1_prior_pred_draws) + scale_fill_manual(values = c(color_scheme_get()$light))

grid.arrange(k1_p1_plt, k2_p1_plt, mu_p1_plt, P_p1_plt, gamma_p1_plt, sigma_p1_plt, ncol = 2, nrow = 3)


min_p1 <- ppc_stat(y = data_list_single$y, yrep = as_draws_matrix(subset_draws(posterior_p1, regex = TRU
max_p1 <- ppc_stat(y = data_list_single$y, yrep = as_draws_matrix(subset_draws(posterior_p1, regex = TRU
grid.arrange(min_p1, max_p1, ncol = 2)

ppc_stat(y = data_list_single$y, yrep = as_draws_matrix(subset_draws(posterior_p1, regex = TRUE, variabl
```

```r
ypreds_p1[sample(nrow(ypreds_p1), 30000), ] %>% ggplot(aes(time, y_pred, group = chain_iter)) +
  geom_line(alpha = 0.007) +
  geom_point(data = data %>% filter(number == 2), mapping = aes(t, C), inherit.aes = FALSE) +
  theme_minimal() +
  labs(x = "t", y = "y/y_pred")

ppc_dens_overlay(
  y = data_list_single$y,
  yrep = head(as_draws_matrix(subset_draws(posterior_p1, regex = TRUE, variable = "y_pred")), 100)
)

data_list_single$only_prior <- 1
data_list_single$initial_cell_counts <- c(9600 * 1e-4, 1e4 * 1e-4, 0.0)
data_list_single$y <- participant_data[, 2] %>% pull("5")

model_prior_fit_p2 <- ode_model$sample(data_list_single, seed = 365, refresh = 0, chains = 3, parallel_

data_list_single$only_prior <- 0
model_fit_p2 <- ode_model$sample(data_list_single, seed = 365, refresh = 0, chains = 3, parallel_chains

prior_p2 <- model_prior_fit_p2$draws()
posterior_p2 <- model_fit_p2$draws()

ypreds_p2 <- posterior_p2 %>%
  reshape2::melt() %>%
  filter(str_detect(variable, "y_pred")) %>%
  tidyr::extract(
    col = variable, into = "ind",
    regex = "y_pred\\[([0-9]*)\\]",
    convert = TRUE
  )
ypreds_p2 <- ypreds_p2 %>%
  mutate(
    time = data_list_single$days_data[ind],
    chain_iter = glue::glue("chain {chain}, iteration {iteration}"),
    .keep = "unused"
  ) %>%
  rename(y_pred = value)
ypreds_p2 <- ypreds_p2 %>% mutate(time = (time + 1) * seq_timestep)

k1_posterior_p2 <- melt(as_draws_matrix(subset_draws(model_fit_p2$draws(), regex = TRUE, variable = "k1
k1_prior_p2 <- melt(as_draws_matrix(subset_draws(model_prior_fit_p2$draws(), regex = TRUE, variable = "

k1_p2_comparison_df <- rbind(k1_prior_p2, k1_posterior_p2)

k1_p2_plt <- ggplot(k1_p2_comparison_df, aes(x = value, fill = variable, color = variable)) +
  geom_histogram(alpha = 1) +
  scale_fill_manual(values = c(color_scheme_get()$dark, color_scheme_get()$light)) +
  theme_minimal() +
  theme(legend.position = "none") +
  labs(x = "k1, Guinea Pig number 2") +
  scale_color_manual(values = c(color_scheme_get()$dark_highlight, color_scheme_get()$light_highlight))
```

```r
k2_posterior_p2 <- melt(as_draws_matrix(subset_draws(model_fit_p2$draws(), regex = TRUE, variable = "k2
k2_prior_p2 <- melt(as_draws_matrix(subset_draws(model_prior_fit_p2$draws(), regex = TRUE, variable = "

k2_p2_comparison_df <- rbind(k2_prior_p2, k2_posterior_p2)

k2_p2_plt <- ggplot(k2_p2_comparison_df, aes(x = value, fill = variable, color = variable)) +
  geom_histogram(alpha = 1) +
  scale_fill_manual(values = c(color_scheme_get()$dark, color_scheme_get()$light)) +
  scale_color_manual(values = c(color_scheme_get()$dark_highlight, color_scheme_get()$light_highlight))
  theme_minimal() +
  theme(legend.position = "none") +
  labs(x = "k2, Guinea Pig number 2")
# comparison_prior_df <- comparison_prior_df  %>% mutate(variable = str_replace_all(comparison_prior_df




gamma_p2_posterior <- melt(as_draws_matrix(subset_draws(model_fit_p2$draws(), regex = TRUE, variable =
gamma_p2_prior <- melt(as_draws_matrix(subset_draws(model_prior_fit_p2$draws(), regex = TRUE, variable =

gamma_p2_comparison_df <- rbind(gamma_p2_posterior, gamma_p2_prior)
# comparison_prior_df <- comparison_prior_df  %>% mutate(variable = str_replace_all(comparison_prior_df

gamma_p2_plt <- ggplot(gamma_p2_comparison_df, aes(x = value, fill = variable, color = variable)) +
  geom_histogram(alpha = 1) +
  scale_fill_manual(values = c(color_scheme_get()$dark, color_scheme_get()$light)) +
  theme_minimal() +
  theme(legend.position = "none") +
  labs(x = "gamma") +
  scale_color_manual(values = c(color_scheme_get()$dark_highlight, color_scheme_get()$light_highlight))


P_posterior_p2 <- melt(as_draws_matrix(subset_draws(model_fit_p2$draws(), regex = TRUE, variable = "P"))
P_prior_p2 <- melt(as_draws_matrix(subset_draws(model_prior_fit_p2$draws(), regex = TRUE, variable = "P

P_p2_comparison_df <- rbind(P_posterior_p2, P_prior_p2)
# comparison_prior_df <- comparison_prior_df  %>% mutate(variable = str_replace_all(comparison_prior_df

P_p2_plt <- ggplot(P_p2_comparison_df, aes(x = value, fill = variable, color = variable)) +
  geom_histogram(alpha = 1) +
  scale_fill_manual(values = c(color_scheme_get()$dark, color_scheme_get()$light)) +
  theme_minimal() +
  theme(legend.position = "none") +
  labs(x = "P") +
  scale_color_manual(values = c(color_scheme_get()$dark_highlight, color_scheme_get()$light_highlight))


mu_posterior_p2 <- melt(as_draws_matrix(subset_draws(model_fit_p2$draws(), regex = TRUE, variable = "mu
mu_prior_p2 <- melt(as_draws_matrix(subset_draws(model_prior_fit_p2$draws(), regex = TRUE, variable = "

mu_p2_comparison_df <- rbind(mu_posterior_p2, mu_prior_p2)
# comparison_prior_df <- comparison_prior_df  %>% mutate(variable = str_replace_all(comparison_prior_df
```

```
mu_p2_plt <- ggplot(mu_p2_comparison_df, aes(x = value, fill = variable, color = variable)) +
  geom_histogram(size = 0.25) +
  scale_fill_manual(values = c(color_scheme_get()$dark, color_scheme_get()$light)) +
  theme_minimal() +
  theme(legend.position = "none") +
  labs(x = "mu") +
  scale_color_manual(values = c(color_scheme_get()$dark_highlight, color_scheme_get()$light_highlight))

sigma_posterior_p2 <- melt(as_draws_matrix(subset_draws(model_fit_p2$draws(), regex = TRUE, variable =
sigma_prior_p2 <- melt(as_draws_matrix(subset_draws(model_prior_fit_p2$draws(), regex = TRUE, variable =

sigma_p2_comparison_df <- rbind(sigma_posterior_p2, sigma_prior_p2)
# comparison_prior_df <- comparison_prior_df  %>% mutate(variable = str_replace_all(comparison_prior_df

sigma_p2_plt <- ggplot(sigma_p2_comparison_df, aes(x = value, fill = variable, color = variable)) +
  geom_histogram(alpha = 1, size = 0.25) +
  scale_fill_manual(values = c(color_scheme_get()$dark, color_scheme_get()$light)) +
  theme_minimal() +
  theme(legend.position = "none") +
  labs(x = "sigma") +
  scale_color_manual(values = c(color_scheme_get()$dark_highlight, color_scheme_get()$light_highlight))

p2_prior_pred_draws <- model_prior_fit_p2$draws(c("y_pred[1]", "y_pred[2]", "y_pred[3]", "y_pred[4]", "
mcmc_hist(p2_prior_pred_draws) + scale_fill_manual(values = c(color_scheme_get()$light))

grid.arrange(k1_p2_plt, k2_p2_plt, mu_p2_plt, P_p2_plt, gamma_p2_plt, sigma_p2_plt, ncol = 2, nrow = 3)

min_p2 <- ppc_stat(y = data_list_single$y, yrep = as_draws_matrix(subset_draws(posterior_p2, regex = TRU
max_p2 <- ppc_stat(y = data_list_single$y, yrep = as_draws_matrix(subset_draws(posterior_p2, regex = TRU
grid.arrange(min_p2, max_p2, ncol = 2)

ppc_stat(y = data_list_single$y, yrep = as_draws_matrix(subset_draws(posterior_p2, regex = TRUE, variabl

ypreds_p2[sample(nrow(ypreds_p2), 30000), ] %>% ggplot(aes(time, y_pred, group = chain_iter)) +
  geom_line(alpha = 0.007) +
  geom_point(data = data %>% filter(number == 5), mapping = aes(t, C), inherit.aes = FALSE) +
  theme_minimal() +
  labs(x = "t", y = "y/y_pred")

ppc_dens_overlay(
  y = data_list_single$y,
  yrep = head(as_draws_matrix(subset_draws(posterior_p2, regex = TRUE, variable = "y_pred")), 100)
)

data_list_single$only_prior <- 1
data_list_single$initial_cell_counts <- c(9600 * 1e-4, 1e2 * 1e-4, 0.0)
data_list_single$y <- participant_data[, 3] %>% pull("7")

model_prior_fit_p3 <- ode_model$sample(data_list_single, seed = 365, refresh = 1000, chains = 3, paralle

data_list_single$only_prior <- 0
model_fit_p3 <- ode_model$sample(data_list_single, seed = 365, refresh = 1000, chains = 3, parallel_cha
```

```
prior_p3 <- model_prior_fit_p3$draws()
posterior_p3 <- model_fit_p3$draws()

ypreds_p3 <- posterior_p3 %>%
  reshape2::melt() %>%
  filter(str_detect(variable, "y_pred")) %>%
  tidyr::extract(
    col = variable, into = "ind",
    regex = "y_pred\\[([0-9]*)\\]",
    convert = TRUE
  )
ypreds_p3 <- ypreds_p3 %>%
  mutate(
    time = data_list_single$days_data[ind],
    chain_iter = glue::glue("chain {chain}, iteration {iteration}"),
    .keep = "unused"
  ) %>%
  rename(y_pred = value)
ypreds_p3 <- ypreds_p3 %>% mutate(time = (time + 1) * seq_timestep)


k1_posterior_p3 <- melt(as_draws_matrix(subset_draws(model_fit_p3$draws(), regex = TRUE, variable = "k1
k1_prior_p3 <- melt(as_draws_matrix(subset_draws(model_prior_fit_p3$draws(), regex = TRUE, variable = "

k1_p3_comparison_df <- rbind(k1_prior_p3, k1_posterior_p3)

k1_p3_plt <- ggplot(k1_p3_comparison_df, aes(x = value, fill = variable, color = variable)) +
  geom_histogram(alpha = 1) +
  scale_fill_manual(values = c(color_scheme_get()$dark, color_scheme_get()$light)) +
  theme_minimal() +
  theme(legend.position = "none") +
  labs(x = "k1, Guinea Pig number 2") +
  scale_color_manual(values = c(color_scheme_get()$dark_highlight, color_scheme_get()$light_highlight))


k2_posterior_p3 <- melt(as_draws_matrix(subset_draws(model_fit_p3$draws(), regex = TRUE, variable = "k2
k2_prior_p3 <- melt(as_draws_matrix(subset_draws(model_prior_fit_p3$draws(), regex = TRUE, variable = "

k2_p3_comparison_df <- rbind(k2_prior_p3, k2_posterior_p3)

k2_p3_plt <- ggplot(k2_p3_comparison_df, aes(x = value, fill = variable, color = variable)) +
  geom_histogram(alpha = 1) +
  scale_fill_manual(values = c(color_scheme_get()$dark, color_scheme_get()$light)) +
  theme_minimal() +
  theme(legend.position = "none") +
  labs(x = "k2, Guinea Pig number 2") +
  scale_color_manual(values = c(color_scheme_get()$dark_highlight, color_scheme_get()$light_highlight))



gamma_p3_posterior <- melt(as_draws_matrix(subset_draws(model_fit_p3$draws(), regex = TRUE, variable =
gamma_p3_prior <- melt(as_draws_matrix(subset_draws(model_prior_fit_p3$draws(), regex = TRUE, variable
```

```
gamma_p3_comparison_df <- rbind(gamma_p3_posterior, gamma_p3_prior)
# comparison_prior_df <- comparison_prior_df  %>% mutate(variable = str_replace_all(comparison_prior_df

gamma_p3_plt <- ggplot(gamma_p3_comparison_df, aes(x = value, fill = variable, color = variable)) +
  geom_histogram(alpha = 1) +
  scale_fill_manual(values = c(color_scheme_get()$dark, color_scheme_get()$light)) +
  theme_minimal() +
  theme(legend.position = "none") +
  labs(x = "gamma") +
  scale_color_manual(values = c(color_scheme_get()$dark_highlight, color_scheme_get()$light_highlight))


P_posterior_p3 <- melt(as_draws_matrix(subset_draws(model_fit_p3$draws(), regex = TRUE, variable = "P"))
P_prior_p3 <- melt(as_draws_matrix(subset_draws(model_prior_fit_p3$draws(), regex = TRUE, variable = "P

P_p3_comparison_df <- rbind(P_posterior_p3, P_prior_p3)
# comparison_prior_df <- comparison_prior_df  %>% mutate(variable = str_replace_all(comparison_prior_df

P_p3_plt <- ggplot(P_p3_comparison_df, aes(x = value, fill = variable, color = variable)) +
  geom_histogram(alpha = 1) +
  scale_fill_manual(values = c(color_scheme_get()$dark, color_scheme_get()$light)) +
  theme_minimal() +
  theme(legend.position = "none") +
  labs(x = "P") +
  scale_color_manual(values = c(color_scheme_get()$dark_highlight, color_scheme_get()$light_highlight))


mu_posterior_p3 <- melt(as_draws_matrix(subset_draws(model_fit_p3$draws(),
                                                     regex = TRUE,
                                                     variable = "mu"))) %>%
  mutate(variable = str_replace_all(variable, pattern = "mu*", replacement = "posterior"))
mu_prior_p3 <- melt(as_draws_matrix(subset_draws(model_prior_fit_p3$draws(),
                                                 regex = TRUE, variable = "mu"))) %>%
  mutate(variable = str_replace_all(variable, pattern = "mu*",
                                    replacement = "prior"))

mu_p3_comparison_df <- rbind(mu_posterior_p3, mu_prior_p3)
# comparison_prior_df <- comparison_prior_df  %>% mutate(variable = str_replace_all(comparison_prior_df

mu_p3_plt <- ggplot(mu_p3_comparison_df, aes(x = value, fill = variable, color = variable)) +
  geom_histogram(size = 0.25) +
  scale_fill_manual(values = c(color_scheme_get()$dark, color_scheme_get()$light)) +
  theme_minimal() +
  theme(legend.position = "none") +
  labs(x = "mu") +
  scale_color_manual(values = c(color_scheme_get()$dark_highlight, color_scheme_get()$light_highlight))

sigma_posterior_p3 <- melt(as_draws_matrix(subset_draws(model_fit_p3$draws(), regex = TRUE, variable =
  mutate(variable = str_replace_all(variable, pattern = "sigma*", replacement = "posterior"))
sigma_prior_p3 <- melt(as_draws_matrix(subset_draws(model_prior_fit_p3$draws(), regex = TRUE, variable
  mutate(variable = str_replace_all(variable, pattern = "sigma*", replacement = "prior"))

sigma_p3_comparison_df <- rbind(sigma_posterior_p3, sigma_prior_p3)
```

```r
# comparison_prior_df <- comparison_prior_df  %>% mutate(variable = str_replace_all(comparison_prior_df

sigma_p3_plt <- ggplot(sigma_p3_comparison_df, aes(x = value, fill = variable, color = variable)) +
  geom_histogram(alpha = 1, size = 0.25) +
  scale_fill_manual(values = c(color_scheme_get()$dark, color_scheme_get()$light)) +
  theme_minimal() +
  theme(legend.position = "none") +
  labs(x = "sigma") +
  scale_color_manual(values = c(color_scheme_get()$dark_highlight, color_scheme_get()$light_highlight))

p3_prior_pred_draws <- model_prior_fit_p3$draws(c("y_pred[1]", "y_pred[2]", "y_pred[3]", "y_pred[4]", "
mcmc_hist(p3_prior_pred_draws) + scale_fill_manual(values = c(color_scheme_get()$light))

grid.arrange(k1_p3_plt, k2_p3_plt, mu_p3_plt, P_p3_plt, gamma_p3_plt, sigma_p3_plt, ncol = 2, nrow = 3)

min_p3 <- ppc_stat(y = data_list_single$y, yrep = as_draws_matrix(subset_draws(posterior_p3, regex = TRU
max_p3 <- ppc_stat(y = data_list_single$y, yrep = as_draws_matrix(subset_draws(posterior_p3, regex = TRU
grid.arrange(min_p3, max_p3, ncol = 2)

ppc_stat(y = data_list_single$y, yrep = as_draws_matrix(subset_draws(posterior_p3, regex = TRUE, variabl

ypreds_p3[sample(nrow(ypreds_p3), 30000), ] %>% ggplot(aes(time, y_pred, group = chain_iter)) +
  geom_line(alpha = 0.007) +
  geom_point(data = data %>% filter(number == 5), mapping = aes(t, C), inherit.aes = FALSE) +
  theme_minimal() +
  labs(x = "t", y = "y/y_pred")

ppc_dens_overlay(
  y = data_list_single$y,
  yrep = head(as_draws_matrix(subset_draws(posterior_p3, regex = TRUE, variable = "y_pred")), 100)
)

data_list_single$initial_cell_counts <- c(9600 * 1e-4, 1e3 * 1e-4, 0.0)
data_list_single$y <- participant_data[, 4] %>% pull("11")
data_list_single$only_prior <- 1

model_prior_fit_p4 <- ode_model$sample(data_list_single, seed = 365, refresh = 1000, chains = 3, paralle

data_list_single$only_prior <- 0
model_fit_p4 <- ode_model$sample(data_list_single, seed = 365, refresh = 1000, chains = 3, parallel_cha

prior_p4 <- model_prior_fit_p4$draws()
posterior_p4 <- model_fit_p4$draws()

ypreds_p4 <- posterior_p4 %>%
  reshape2::melt() %>%
  filter(str_detect(variable, "y_pred")) %>%
  tidyr::extract(
    col = variable, into = "ind",
    regex = "y_pred\\[([0-9]*)\\]",
    convert = TRUE
  )
ypreds_p4 <- ypreds_p4 %>%
```

```
  mutate(
    time = data_list_single$days_data[ind],
    chain_iter = glue::glue("chain {chain}, iteration {iteration}"),
    .keep = "unused"
  ) %>%
  rename(y_pred = value)
ypreds_p4 <- ypreds_p4 %>% mutate(time = (time + 1) * seq_timestep)


k1_posterior_p4 <- melt(as_draws_matrix(subset_draws(model_fit_p4$draws(), regex = TRUE, variable = "k1
k1_prior_p4 <- melt(as_draws_matrix(subset_draws(model_prior_fit_p4$draws(), regex = TRUE, variable = "

k1_p4_comparison_df <- rbind(k1_prior_p4, k1_posterior_p4)

k1_p4_plt <- ggplot(k1_p4_comparison_df, aes(x = value, fill = variable, color = variable)) +
  geom_histogram(alpha = 1) +
  scale_fill_manual(values = c(color_scheme_get()$dark, color_scheme_get()$light)) +
  theme_minimal() +
  theme(legend.position = "none") +
  labs(x = "k1, Guinea Pig number 2") +
  scale_color_manual(values = c(color_scheme_get()$dark_highlight, color_scheme_get()$light_highlight))

k2_posterior_p4 <- melt(as_draws_matrix(subset_draws(model_fit_p4$draws(), regex = TRUE, variable = "k2
k2_prior_p4 <- melt(as_draws_matrix(subset_draws(model_prior_fit_p4$draws(), regex = TRUE, variable = "

k2_p4_comparison_df <- rbind(k2_prior_p4, k2_posterior_p4)

k2_p4_plt <- ggplot(k2_p4_comparison_df, aes(x = value, fill = variable, color = variable)) +
  geom_histogram(alpha = 1) +
  scale_fill_manual(values = c(color_scheme_get()$dark, color_scheme_get()$light)) +
  theme_minimal() +
  theme(legend.position = "none") +
  labs(x = "k2, Guinea Pig number 2") +
  scale_color_manual(values = c(color_scheme_get()$dark_highlight, color_scheme_get()$light_highlight))



gamma_p4_posterior <- melt(as_draws_matrix(subset_draws(model_fit_p4$draws(), regex = TRUE, variable =
gamma_p4_prior <- melt(as_draws_matrix(subset_draws(model_prior_fit_p4$draws(), regex = TRUE, variable =

gamma_p4_comparison_df <- rbind(gamma_p4_posterior, gamma_p4_prior)
# comparison_prior_df <- comparison_prior_df  %>% mutate(variable = str_replace_all(comparison_prior_df

gamma_p4_plt <- ggplot(gamma_p4_comparison_df, aes(x = value, fill = variable, color = variable)) +
  geom_histogram(alpha = 1) +
  scale_fill_manual(values = c(color_scheme_get()$dark, color_scheme_get()$light)) +
  theme_minimal() +
  theme(legend.position = "none") +
  labs(x = "gamma") +
  scale_color_manual(values = c(color_scheme_get()$dark_highlight, color_scheme_get()$light_highlight))


P_posterior_p4 <- melt(as_draws_matrix(subset_draws(model_fit_p4$draws(), regex = TRUE, variable = "P"))
```

```
P_prior_p4 <- melt(as_draws_matrix(subset_draws(model_prior_fit_p4$draws(), regex = TRUE, variable = "P

P_p4_comparison_df <- rbind(P_posterior_p4, P_prior_p4)
# comparison_prior_df <- comparison_prior_df  %>% mutate(variable = str_replace_all(comparison_prior_df

P_p4_plt <- ggplot(P_p4_comparison_df, aes(x = value, fill = variable, color = variable)) +
  geom_histogram(alpha = 1) +
  scale_fill_manual(values = c(color_scheme_get()$dark, color_scheme_get()$light)) +
  theme_minimal() +
  theme(legend.position = "none") +
  labs(x = "P") +
  scale_color_manual(values = c(color_scheme_get()$dark_highlight, color_scheme_get()$light_highlight))


mu_posterior_p4 <- melt(as_draws_matrix(subset_draws(model_fit_p4$draws(), regex = TRUE, variable = "mu
mu_prior_p4 <- melt(as_draws_matrix(subset_draws(model_prior_fit_p4$draws(), regex = TRUE, variable = "

mu_p4_comparison_df <- rbind(mu_posterior_p4, mu_prior_p4)
# comparison_prior_df <- comparison_prior_df  %>% mutate(variable = str_replace_all(comparison_prior_df

mu_p4_plt <- ggplot(mu_p4_comparison_df, aes(x = value, fill = variable, color = variable)) +
  geom_histogram(size = 0.25) +
  scale_fill_manual(values = c(color_scheme_get()$dark, color_scheme_get()$light)) +
  theme_minimal() +
  theme(legend.position = "none") +
  labs(x = "mu") +
  scale_color_manual(values = c(color_scheme_get()$dark_highlight, color_scheme_get()$light_highlight))

sigma_posterior_p4 <- melt(as_draws_matrix(subset_draws(model_fit_p4$draws(), regex = TRUE, variable =
sigma_prior_p4 <- melt(as_draws_matrix(subset_draws(model_prior_fit_p4$draws(), regex = TRUE, variable

sigma_p4_comparison_df <- rbind(sigma_posterior_p4, sigma_prior_p4)
# comparison_prior_df <- comparison_prior_df  %>% mutate(variable = str_replace_all(comparison_prior_df

sigma_p4_plt <- ggplot(sigma_p4_comparison_df, aes(x = value, fill = variable, color = variable)) +
  geom_histogram(alpha = 1, size = 0.25) +
  scale_fill_manual(values = c(color_scheme_get()$dark, color_scheme_get()$light)) +
  theme_minimal() +
  theme(legend.position = "none") +
  labs(x = "sigma") +
  scale_color_manual(values = c(color_scheme_get()$dark_highlight,
                                color_scheme_get()$light_highlight))


p4_prior_pred_draws <- model_prior_fit_p4$draws(c("y_pred[1]", "y_pred[2]",
                                                  "y_pred[3]", "y_pred[4]",
                                                  "y_pred[5]", "y_pred[6]",
                                                  "y_pred[7]", "y_pred[8]",
                                                  "y_pred[9]", "y_pred[10]"))
mcmc_hist(p4_prior_pred_draws) + scale_fill_manual(values =
                                                     c(color_scheme_get()$light))

grid.arrange(k1_p4_plt, k2_p4_plt, mu_p4_plt, P_p4_plt, gamma_p4_plt,
```

```
            sigma_p4_plt, ncol = 2, nrow = 3)

min_p4 <- ppc_stat(y = data_list_single$y,
                   yrep = as_draws_matrix(subset_draws(posterior_p4,
                                                       regex = TRUE,
                                                       variable = "y_pred")),
                   stat = "min")
max_p4 <- ppc_stat(y = data_list_single$y,
                   yrep = as_draws_matrix(subset_draws(posterior_p4,
                                                       regex = TRUE,
                                                       variable = "y_pred")),
                   stat = "max")
grid.arrange(min_p4, max_p4, ncol = 2)

ppc_stat(y = data_list_single$y, yrep =
           as_draws_matrix(subset_draws(
             posterior_p4, regex = TRUE,
             variable = "y_pred")), stat = "skewness")

ypreds_p4[sample(nrow(ypreds_p4), 30000), ] %>%
  ggplot(aes(time, y_pred, group = chain_iter)) +
  geom_line(alpha = 0.007) +
  geom_point(data = data %>% filter(number == 5),
             mapping = aes(t, C), inherit.aes = FALSE) +
  theme_minimal() +
  labs(x = "t", y = "y/y_pred")

ppc_dens_overlay(
  y = data_list_single$y,
  yrep = head(as_draws_matrix(
    subset_draws(posterior_p4, regex = TRUE, variable = "y_pred")), 100)
)

opencl_options <- list(
  stan_opencl = TRUE,
  opencl_platform_id = 0,
  opencl_device_id = 0,
  ldflags_opencl =
    "-L'C:\\Program Files\\NVIDIA GPU Computing Toolkit\\CUDA\\v11.2\\lib\\x64'"
)
imputation_model <- cmdstan_model("a1_task3.stan", compile = TRUE, threads = 12)

draws_df <- as_draws_df(posterior_p4)
random_row <- sample(1:nrow(draws_df), 1)
days_data_raw <- sexual %>%
  arrange(t) %>%
  distinct(t)
days_data_conv <- days_data_raw %>%
  mutate(t = ((t - seq_start) / seq_timestep + 1)) %>%
  pull(t)
participant_data_sexual <- sexual %>%
  select(c(t, C, number)) %>%
  arrange(t, number) %>%
```

```
  pivot_wider(names_from = number, values_from = C) %>%
  select(-c(t)) %>%
  pull("11")
pred_times_t3 <- sexual %>%
  select(t) %>%
  pull(t)
pred_times_seq_t3 <- seq(0.1, max(pred_times_t3), by = 0.01)

for (i in 1:100) {
  sampled_row <- draws_df[random_row, ]
  conditional_params <- list(
    participants = 1,
    n_days = length(pred_times_seq),
    pred_times = pred_times_seq,
    initial_cell_counts = c(9600 * 1e-4, 0.0),
    n_measurements = sexual %>% select(t) %>% n_distinct(),
    days_data = days_data_conv,
    y = participant_data_sexual,
    Pe = 40 * 1e-4,
    delta_e = 2,
    k1_mu = sampled_row %>% pull(k1),
    k1_tau = 0.1,
    k2_mu = sampled_row %>% pull(k2),
    k2_tau = 0.1,
    P_mu = sampled_row %>% pull(P),
    P_tau = 0.1,
    mu_param_mu = sampled_row %>% pull(mu_param),
    mu_param_tau = 0.1,
    gamma_mu = sampled_row %>% pull(gamma),
    gamma_tau = 0.1,
    sigma_mu = sampled_row %>% pull(sigma),
    sigma_tau = 0.1,
    c0_mu = 1e2 * 1e-4,
    c0_tau = 4e2 * 1e-5,
    only_prior = 0
  )
  multi_imputation_fit <- imputation_model$sample(conditional_params,
                                                  seed = 365,
                                                  refresh = 1000,
                                                  parallel_chains = 4,
                                                  iter_warmup = 1000,
                                                  iter_sampling = 200)
  draws_df <- as_draws_df(multi_imputation_fit$draws())
  random_row <- sample(1:nrow(draws_df), 1)
}


ini_cell_counts <- c(E = 9600 * 1e-4, C = 1, I = 0)
pred_times <- data %>%
  filter(number == 2) %>%
  select(t) %>%
  pull(t)
# pred_times <- pred_times %>% insert(ats = 1, values = c(0.0))
```

```
pred_times_seq <- seq(0.0001, max(pred_times), by = 0.01)

test_params_hi$initial_cell_counts["C"] <- 1
test_params_hi_ct <- param_model$sample(test_params_hi, seed = 365, refresh = 0,
                                        fixed_param = T, chains = 1,
                                        iter_warmup = 10,
                                        iter_sampling = 1)

test_params_lo$initial_cell_counts["C"] <- 1
test_params_lo_ct <- param_model$sample(test_params_lo, seed = 365, refresh = 0,
                                        fixed_param = T, chains = 1,
                                        iter_warmup = 10, iter_sampling = 1)
cs <- test_params_hi_ct$draws() %>%
  reshape2::melt() %>%
  dplyr::filter(str_detect(variable, "C")) %>%
  tidyr::extract(
    col = variable, into = "ind",
    regex = "C\\[([0-9]*)\\]",
    convert = TRUE
  ) %>%
  dplyr::mutate(
    time = test_params_hi$pred_times[ind],
    chain_iter = glue::glue("chain {chain}, iteration {iteration}"),
    .keep = "unused"
  ) %>%
  rename(mu = value) %>%
  mutate(cell_type = rep("c", test_params_hi$n_days))




cs %>% ggplot(aes(x = time, y = mu)) +
  geom_line(alpha = 1) +
  theme_minimal() +
  labs(x = "t, day", y = "C(t), EB count scaled by 10^-4")

cs <- test_params_lo_ct$draws() %>%
  reshape2::melt() %>%
  dplyr::filter(str_detect(variable, "C")) %>%
  tidyr::extract(
    col = variable, into = "ind",
    regex = "C\\[([0-9]*)\\]",
    convert = TRUE
  ) %>%
  dplyr::mutate(
    time = test_params_hi$pred_times[ind],
    chain_iter = glue::glue("chain {chain}, iteration {iteration}"),
    .keep = "unused"
  ) %>%
  rename(mu = value) %>%
  mutate(cell_type = rep("c", test_params_hi$n_days))
```

```r
cs %>% ggplot(aes(x = time, y = mu)) +
  geom_line(alpha = 1) +
  theme_minimal() +
  labs(x = "t, day", y = "C(t), EB count scaled by 10^-4")
```

Stan code

Prior exploration code

```
functions{
  // cell counts:
  // element 1 is the e(t), 2 is c(t), 3 is I(t)
  // t is the inital time,
  vector chl_model(real t, vector cell_counts,
             real Pe, real delta_e, real k1, real k2, real P, real mu, real gamma) {
  vector[3] dydt;
  // dE/dt
  dydt[1] = Pe - delta_e*cell_counts[1] - k1*cell_counts[2]*cell_counts[1];
  // dC/dt
  dydt[2] = P*k2*cell_counts[3] - mu*cell_counts[2] - k1*cell_counts[2]*cell_counts[1];
  // dI/dt
  dydt[3] = k1*cell_counts[2]*cell_counts[1] - gamma*cell_counts[3] - k2*cell_counts[3];
  return dydt;
  }
}


data {
 // our data are the quantities needed to solve our ODEs
 vector[3] initial_cell_counts;
 real Pe;
 real delta_e;
 real k1;
 real k2;
 real P;
 real gamma;
 real mu_param;
 int n_days;
 real pred_times[n_days]; // t[N] in lecture

}

generated quantities{
  vector[n_days] C;
  vector[n_days] E;
  vector[n_days] I;
  {
    // remember to transpose the inital quantities vector!
    // ode, ini_state, ini_time, times, model_params
    vector[3] soln[n_days] = ode_bdf(chl_model, initial_cell_counts, 0.0, pred_times, Pe, delta_e, k1, k
    for(i in 1:n_days){
      C[i] = soln[i, 2];
```

```
      E[i] = soln[i, 1];
      I[i] = soln[i, 3];
    }
  }
}
```

Model code

Single subject:

```
functions{
  // cell counts:
  // element 1 is the e(t), 2 is c(t), 3 is I(t)
  // t is the time,
  vector chl_model(real t, vector cell_counts,
             real pe, real delta_e, real k1, real k2, real p, real mu, real gamma) {
  vector[3] dydt;
  // dE/dt
  dydt[1] = pe - delta_e*cell_counts[1] - k1*cell_counts[2]*cell_counts[1];
  // dC/dt
  dydt[2] = p*k2*cell_counts[3] - mu*cell_counts[2] - k1*cell_counts[2]*cell_counts[1];
  // dI/dt
  dydt[3] = k1*cell_counts[2]*cell_counts[1] - gamma*cell_counts[3] - k2*cell_counts[3];
  return dydt;
  }
}


data {
 int participants; // number of participants in study
 int n_days;
 real pred_times[n_days]; // t[N] in lecture, seq(...)
 // initial state of our system
 vector[3] initial_cell_counts;
 int n_measurements; // n_measurements in data
 int days_data[n_measurements]; // data days
 real<lower = 0> y[n_measurements];/// our data
 real Pe;
 real delta_e;
 real k1_mu;
 real k1_tau;
 real k2_mu;
 real k2_tau;
 real P_mu;
 real P_tau;
 real mu_param_mu;
 real mu_param_tau;
 real gamma_mu;
 real gamma_tau;
 real sigma_tau;
 int only_prior;
}

parameters {
  real k1;
```

```
  real k2;
  real P;
  real mu_param;
  real gamma;
  real<lower=0> sigma;
}

transformed parameters{
  real C[n_days]; // this should be a 2d array, either [n_subject, t_days] or transpose
  //real E[n_days];
  //real I[n_days];
  real C_real[n_measurements]; // ode solns for C on days when measurements were taken
  {

    // ode, ini_state, ini_time, times, model_params
    //vector[3] participant_ini = initial_cell_counts;
    vector[3] soln[n_days] = ode_rk45(chl_model, initial_cell_counts, 0.0, pred_times, Pe, delta_e, k1,
    C = soln[, 2];
    //E = soln[, 1];
    //I = soln[, 3];

    /*for(day in 1:n_days){
      C[day] = soln[day, 2];
      E[day] = soln[day, 1];
      I[day] = soln[day, 3];
    }*/

    for(day_idx in 1:n_measurements){
      int day = days_data[day_idx];
      C_real[day_idx] = C[day];
    }
  }
}

model {
  // priors
  k1 ~ normal(k1_mu, k1_tau);
  k2 ~ normal(k2_mu, k2_tau);
  P ~ normal(P_mu, P_tau);
  mu_param ~ normal(mu_param_mu, mu_param_tau);
  gamma ~ normal(gamma_mu, gamma_tau);
  sigma ~ normal(0, sigma_tau);


  //likelihood of C(t), eb counts
  if(only_prior == 0) {
     y ~ normal(C_real, sigma);
  }

}

generated quantities{
  real y_pred[n_measurements];
  real log_lik[n_measurements];
```

```
  for (day_index in 1:n_measurements){
    //int day = days_data[day_index];
    log_lik[day_index] = normal_lpdf(y[day_index] | C_real[day_index], sigma);
    y_pred[day_index] = normal_rng(C_real[day_index], sigma);
    //print("day, participant, y_pred", day, participant, y_pred);
  }
}
```

Task 3

```
functions{
  // cell counts:
  // element 1 is the e(t), 2 is c(t), 3 is I(t)
  // t is the time,
  vector chl_model(real t, vector cell_counts,
             real pe, real delta_e, real k1, real k2, real p, real mu, real gamma) {
  vector[3] dydt;
  // dE/dt
  dydt[1] = pe - delta_e*cell_counts[1] - k1*cell_counts[2]*cell_counts[1];
  // dC/dt
  dydt[2] = p*k2*cell_counts[3] - mu*cell_counts[2] - k1*cell_counts[2]*cell_counts[1];
  // dI/dt
  dydt[3] = k1*cell_counts[2]*cell_counts[1] - gamma*cell_counts[3] - k2*cell_counts[3];
  return dydt;
  }
}


data {
 int participants; // number of participants in study
 int n_days;
 real pred_times[n_days]; // t[N] in lecture, seq(...)
 // initial state of our system
 vector[2] initial_cell_counts;
 int n_measurements; // n_measurements in data
 int days_data[n_measurements]; // data days
 real<lower = 0> y[n_measurements];/// our data
 real Pe;
 real delta_e;
 real k1;
 real k2;
 real P;
 real mu_param;
 real gamma;
 real sigma;
 real c0_mu;
 real c0_tau;
 int only_prior;
}

parameters {
  real C_0;
}

transformed parameters{
```

```
  real C[n_days];
  real C_real[n_measurements]; // ode solns for C on days when measurements were taken
  {
    // solve the ode
    vector[3] soln[n_days] = ode_bdf(chl_model, [initial_cell_counts[1], C_0, initial_cell_counts[2]]',
    C = soln[, 2];

    // get the days that we also measured on
    for(day_idx in 1:n_measurements){
      int ode_soln_idx = days_data[day_idx];
      C_real[day_idx] = C[ode_soln_idx];
    }
  }
}

model {
  //prior
  C_0 ~ normal(c0_mu, c0_tau);

  // likelihood
  y ~ normal(C_real, sigma);

}
```