STA365: Assignment 2

Eric Zhu

2021-04-23

Contents

Quick Introduction

In this assignment we are tasked with creating 3 models to model the minimum specific acceleration (MSA) of a Verreaux's eagle, a Raptor, which will let us investigate how this bird moves due to varying environmental conditions. As stated in the assignment handout, since these birds glide, their movement can be highly dependent on environmental factors, it's possible that there's a latent variable Z that is their energy state. So in this assignment, we will consider the log(MSA) at various time points (y_i), which we will model as:

$$y_i \mid Z, \mu, \sigma \sim N(\mu_z, \sigma_z)$$

In other words, because we have a latent variable we will use a Gaussian mixture model with Z indicating which Gaussian component observation i comes from. This will allow us to marginalize out the latent variable. Given that we consider y on the log scale, we should be wary that unit changes in y are multiplicative rather than additive.

Since our latent variable's distribution is by definition unknown, we will additionally model the latent variable using a multinomial distribution (where $\boldsymbol{\pi}$ is the vector of mixing proportions):

$$Z \sim \text{multinomial}(1, \boldsymbol{\pi})$$

Finally, we truly know very little about animal ecology from either the handout or background knowledge. In fact, we do not know anything about the bounds or plausible values for the means of our mixture components or information that would allow us to derive reasonable inferences about our model parameters, e.g., μ_z. Therefore, we will lean heavily into constructing extremely weakly informative priors to reflect our lack of domain knowledge. We will also talk about the MSA and log (MSA) as a measure of energy as differences in MSA lead us to believe we have two energy states, and practically, MSA is a proxy for energy. In reality, it may be erroneous to conflate MSA with energy, but we have no additional information to approach this from more ecologically rigorous point of view.

Task 1: Fitting a simple model on simulated data

First, we are tasked with simulating data from a mixture of two Gaussians. That is, we will use the following model:

$$\begin{aligned} Z &\sim \text{multinomial}(1, [p, 1-p]) \\ y_i \mid Z, \mu, \sigma &\sim N(\mu_Z, \sigma_Z) \end{aligned}$$

Note that we again we use Z above to denote the energy state of the raptor and p to be a fixed parameter for our raptor (p is the probability of being in a high energy state). Since we have that $Z \sim \text{multinomial}(1, [p, 1-p])$, we can also just sample Z from a Bernoulli distribution with p as the parameter, i.e., $Z \sim \text{Bernoulli}(p)$.

We set up our experiment as follows:

1. Define how many observation points we want to sample from this mixture (n samples)
2. For each of the n samples, we sample from the Bernoulli distribution, where a 1 corresponds to high energy state and 0 corresponds to the low energy state
3. Depending on the sample from the Bernoulli distribution, we sample from the appropriate Gaussian component

Before conducting our experiment, we will first define our experiment parameters: the parameters for the Gaussian components, our experiment size, and the probability of the raptor being in a high energy state. These parameters will be determined at a conceptual level, and represent what I believe to be sensible given my extremely limited domain knowledge.

We first define n to be 10000, which is an acceptable number of samples and provides good convergence to our defined distributions, namely we should see very clear normal distributions for both components irrespective of our choice for p, the probability of being in a high energy state. Second, we define the probability of our raptor being in a high energy state, p, as 0.2 because it seems sensible that a raptor really only spends 20% of its time in a truly high energy state, perhaps when hunting. Then we'll set the parameters for the two Gaussian components.

We'll first set the Gaussian component for the low energy state. Provided our essential lack of knowledge about raptor behaviour, we may say the raptor is not accelerating when at rest, i.e., perhaps the raptor is sleeping or preening. So then $\log(0)$ would form the centre of the low energy component but since $\log(0)$ is undefined, we instead will use a small number to substitute for 0, which we choose to be 0.01, and so we set the centre of the low energy component to be $\log(0.01) \approx -4.60517$. Zero in the base-10 scale is a good choice because it implies that the raptor is not moving at all, and could be doing low energy activities like sleeping, so the mean for this component represents the raptor at rest.

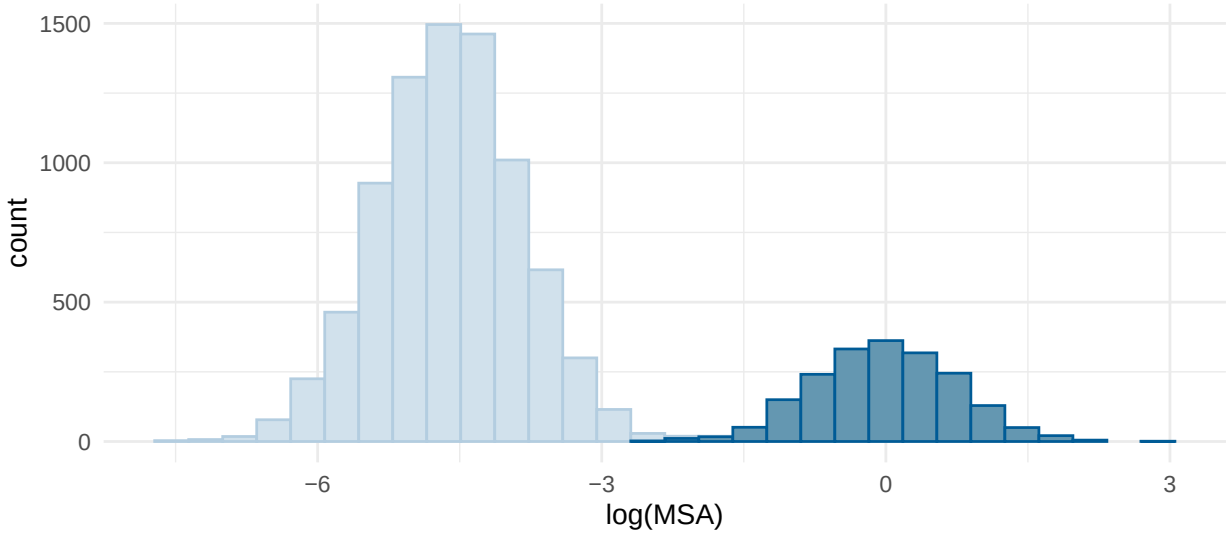
Then we'll set the standard deviation to 0.5 because $\sigma = 0.75 \Rightarrow 2\sigma = 1.5$, i.e., 95% of the distribution's mass falls within ± 1.5 on the log scale, making the 95^{th} percentile ≈ 4.5 times the centre (0.01) on the base-10 scale. This seems somewhat sensible since it could be possible to use 4 times preening than the energy used sleeping and still be considered "low energy" overall. To summarize, we'll set the Gaussian component for low energy to be $N(\log(0.01), 0.75)$.

Finally, we will define the parameters for the high energy component. Since we define the raptor to be only in this high energy state 20% of the time, we also define this state, for the purposes of this experiment, to be one that is representative of the raptor truly expending its energy, such as when the raptor is hunting or in a fight/flight state. That is, we expect the raptor to use far greater energy in this state than in the low energy state, such as when the raptor is at rest or sleeping, consuming minuscule amounts of energy. Thus, we set the mean of this distribution to be $\log(1)$, which is 100 times higher than 0.01, the mean of the low energy component on the base-10 scale, and captures our reasoning for this component quite well. It has the additional benefit of separating the two distributions well, which was a caveat Prof. Simpson made note of in week 11. We again set the standard deviation to be 0.75 because having $\approx 95\%$ of the distribution's mass fall

within ± 1.5 on the log scale seems to be sensible; it will have the effect of making the 95^{th} percentile on the base-10 scale be ≈ 4.5 times greater than 1, the mean of this component on the base-10 scale.

The code below shows the experiment setup and results using parameters we define above:

```
gaussian_params <- matrix(NA, nrow = 2, ncol=2)
gaussian_params[1,] <- c(log(0.01),0.75)
gaussian_params[2,] <- c(log(1), 0.75)
simulated_data_t1 <- simulate_gussian_mixture(10000, c(0.8, 0.2), gaussian_params)
```



The plot above shows the experiment results, where the darker blue histogram is the high energy component and the lighter blue histogram is the low energy component. Clearly, if we consider the marginal (ignoring the individual energy states), we see that there is far more mass around log(0.01), corresponding to the low energy state as we'd expect.

Next, we'll put some priors on our model parameters to examine the fit on our simulated data. Recall that we have two models:

$$Z \sim \text{Bernoulli}(p)$$
$$y_i \mid Z, \mu, \sigma \sim N(\mu_Z, \sigma_Z)$$

Thus, we'll need a prior on p, and priors on μ_z, σ_z for all states that Z can take, which in task 1 is simply our two energy states. We will put normal (or half-normal) priors on all of these model parameters because by the CLT we would expect a normal distribution, and we have a large sample size (10000). Note that p will be constrained to bounds of 0, 1 given that it is a probability, i.e., it will be given a `<lower = 0, upper = 1>` decorator in Stan. Provided that we already know the parameters of the distributions we simulate from, we will be hopelessly biased in determining our priors. In particular, these parameters, e.g., p, already encode my uninformative "beliefs" about raptor energy states. However, we will try to replicate the prior setting process as faithfully as possible, but it will be difficult coming up with really different centres for the prior distributions than simulation parameters as, again, they already represent what I believe to be sensible given my domain knowledge. So I will just use the same centres as they represent sensible choices given what I know about the problem.

First we will put a prior on p. We will again centre p at 0.2 because it seems reasonable that the raptor spends only really about 20% of its time in a high energy state. Then we'll figure out a τ_p , the standard

deviation of the prior. It seems extremely unlikely that the raptor would spend more than 30% of its time in a high energy state, so it is sensible that 95% of this prior mass would be on the interval $[0.1, 0.3]$. So then $\tau_p = 0.05$, and $p \sim N(0.2, 0.05)$.

Next, we'll put priors on our two Gaussian components. First we'll put a prior on the parameters for the low energy component, i.e, μ_1, σ_1. We will use the prior $N(\log(0.01), 0.35)$, meaning we expect the raptor to be at rest in this energy state, but will have $\approx 95\%$ of the mass within [-5.30517, -3.90517], which has the upper bound of the interval as approximately 2 times the mean of this prior on the base-10 scale. As for σ_1 we will use the prior $N_+(0.5, 0.125)$. We do this so that the 95^{th} percentile of this distribution is approximately 0.75. This entails that we expect to very rarely see log(MSA) above -3.15517, i.e., $\mu_{\mu_1} + 2\tau_{\mu_1} + \mu_{\sigma_1} + 2\tau_{\sigma_1} = -3.90517 + 0.75 = -3.15517$. In other words, when converted to the base 10 scale, MSA values more than 4.3 times our centre (0.01) should be rare, which seems very reasonable given our limited domain knowledge; we can interpret this as it's improbable for raptors to expend more than 4.3 times the energy than just resting in the low energy state.

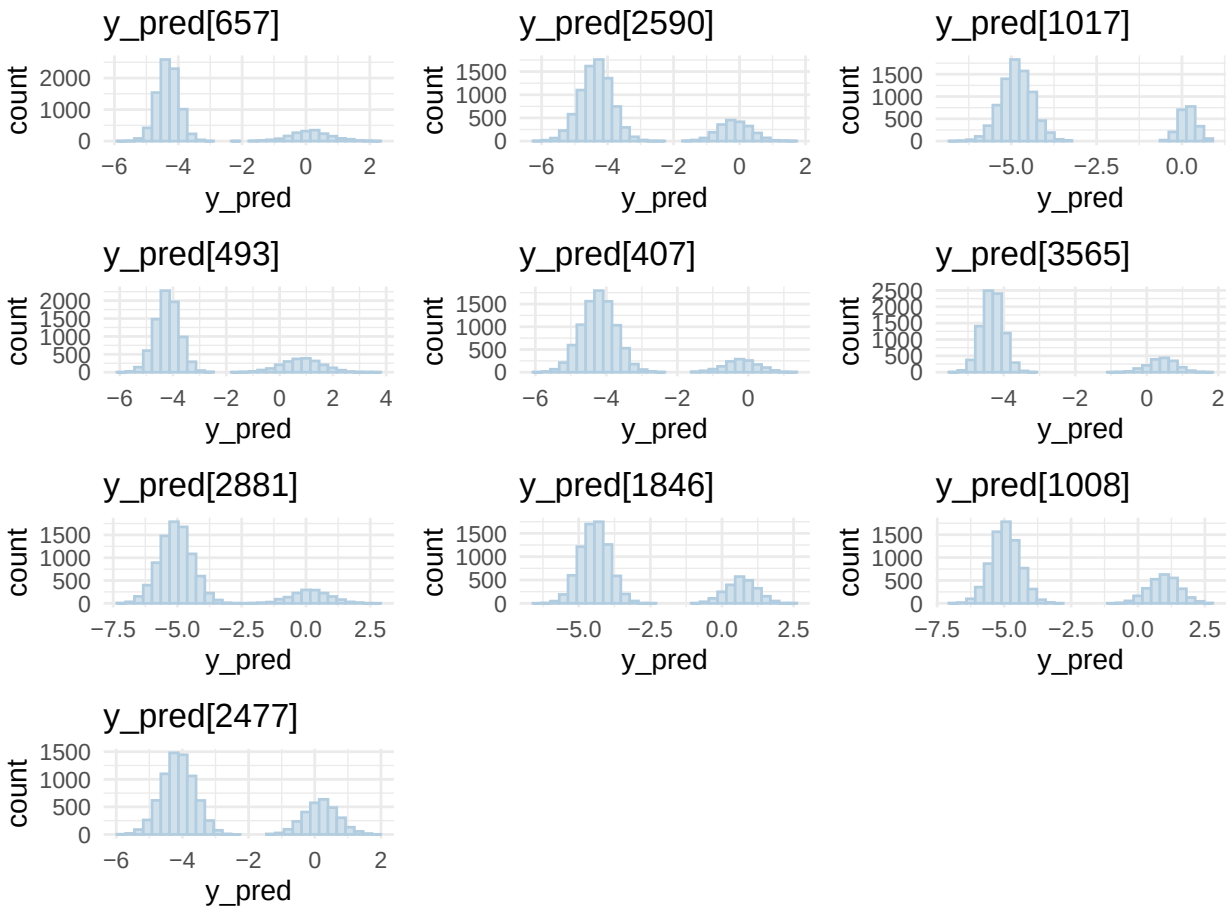
Finally, we'll put a prior on the parameters for our high energy component, i.e., μ_2, σ_2. We'll use the prior $N(\log(1), 0.35)$. This means $\approx 95\%$ of our prior mass is on the interval [-0.7, 0.7], which is reflective the nature of our extremely weakly informative priors. Then we'll use $N_+(0.5, 0.125)$, so that the 95^{th} percentile of this distribution is approximately 0.75. This entails that we expect to very rarely see $\log(MSA)$ above 0.75, i.e., $\mu_{\mu_2} + 2\tau_{\mu_2} + \mu_{\sigma_2} + 2\tau_{\sigma_2} = 0.7 + 0.75 = 1.45$; in other words we rarely expect to see values more than ≈ 4.3 times our centre for this component on the base-10 scale ($\log(1)$). Another implication is that we expect raptors to never expend more than approximately 430 times their resting energy, which seems like a lot but also may be sensible in certain extreme cases if the raptor truly uses tiny amounts of energy in a low energy state. We would be able to put tighter priors should we have had more domain knowledge.

To recap we set the following priors:

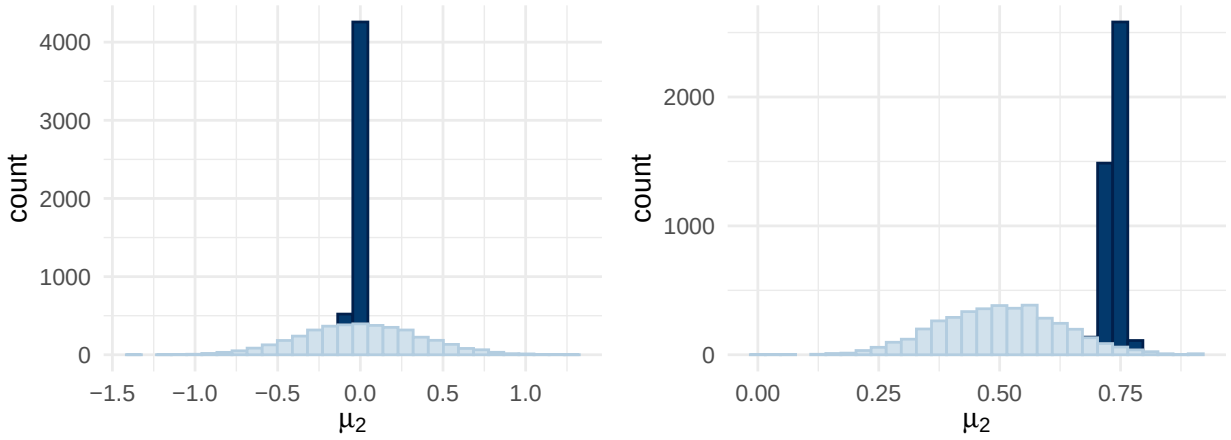
1. $p \sim N(0.2, 0.05)$
2. $\mu_1 \sim N(\log(0.01), 0.35)$
3. $\sigma_1 \sim N_+(0.5, 0.125)$
4. $\mu_2 \sim N(\log(1), 0.35)$
5. $\sigma_2 \sim N_+(0.5, 0.125)$

Prior predictive checks/Comparison of prior and posterior distributions

First, we'll take a look at the prior predictive distribution for 10 randomly sampled observations from the simulated data.

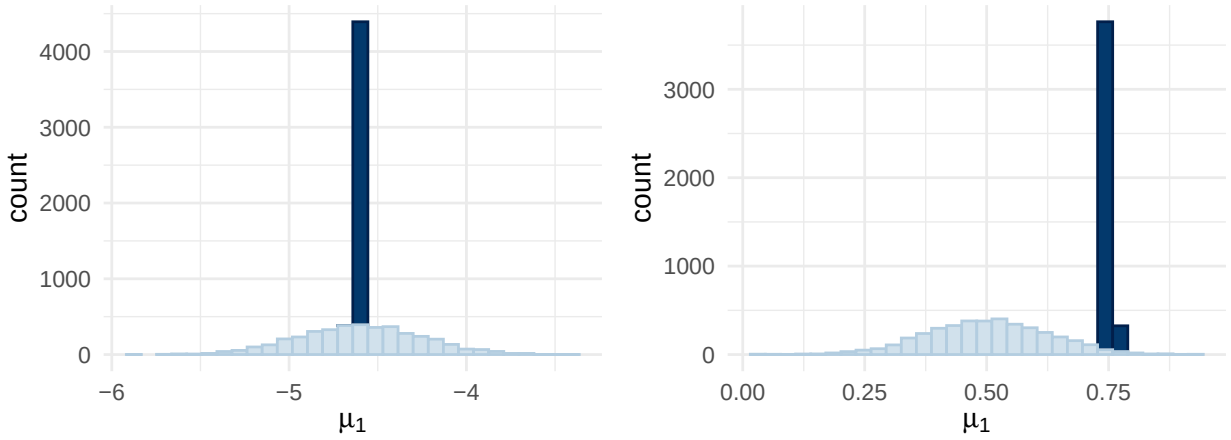


As we can see, they are all bimodal with a clear separation in the two distributions; this is clearly a mixture of Gaussians. This is what we would expect from what we know about the data, as we can see from the plot of the simulated data. They are also fairly wide spread as we would expect to see with our weakly informative priors. So there is no evidence of seriously misspecified priors here, so we will examine the prior/posterior comparison plots, where the darker blue is the posterior and the lighter blue is the prior:

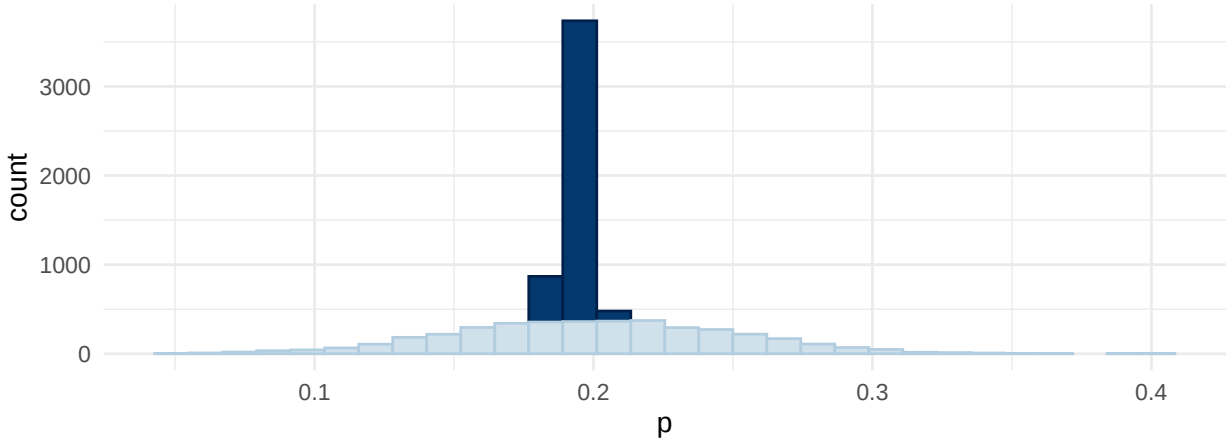


The above plots are the comparison plots of the parameters of the high energy component. As we can see both of the comparison plots show that the posterior (dark blue) contracts within the prior, more so with the comparison plot of μ_2 than that of σ_2. Although the posterior for σ_2 contracts on the right tail of the prior,

it is still where there is considerable prior mass. So there's no evidence of prior-data conflict here.



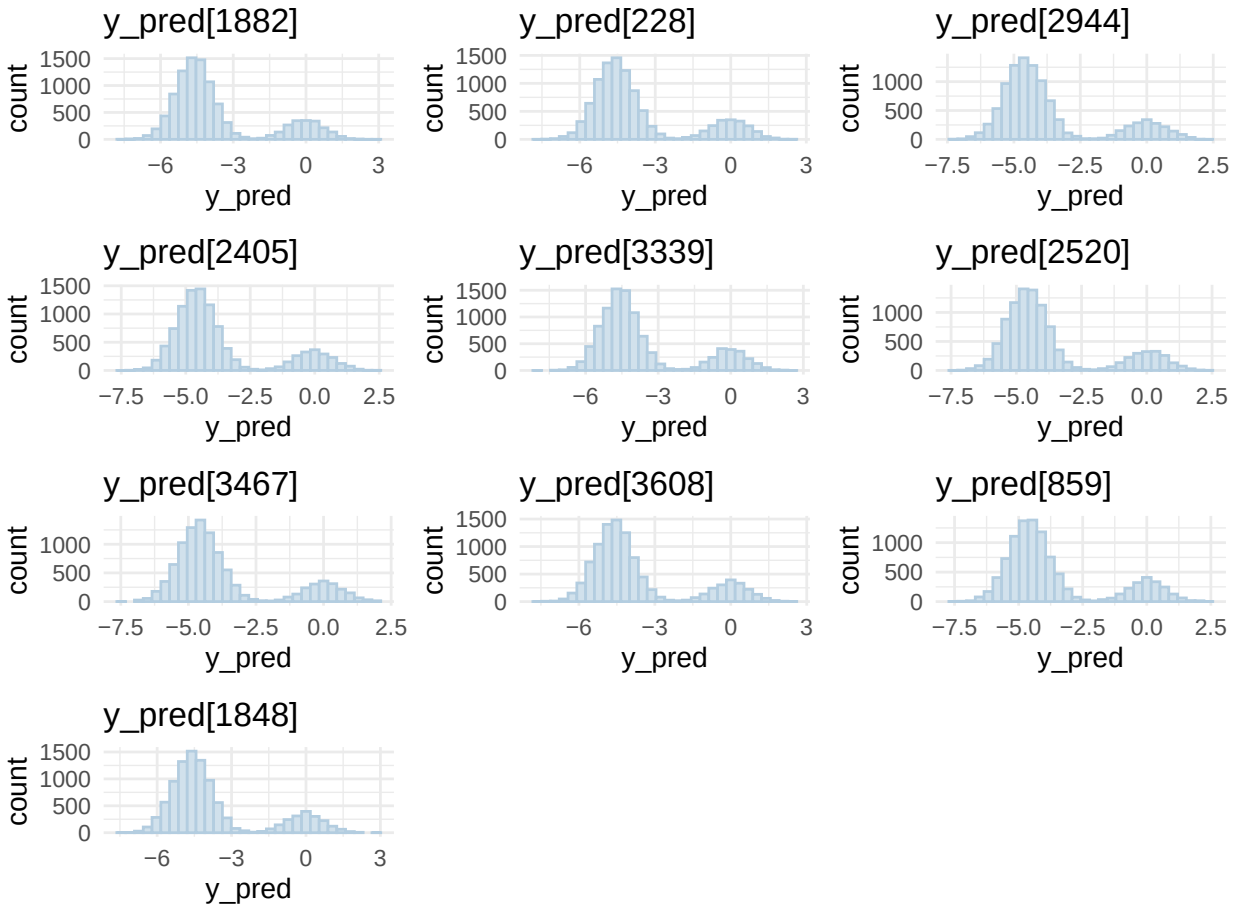
The above plots are the comparison plots of the parameters of the low energy component. Like the plots for the high energy component, our posterior distributions here contract both within the prior, with the posterior distributions behaving much like those of the high energy component. Again, the posterior for σ_1 contracts on the right tail of the prior, where there is considerable prior mass. So there's no evidence of prior-data conflict here.



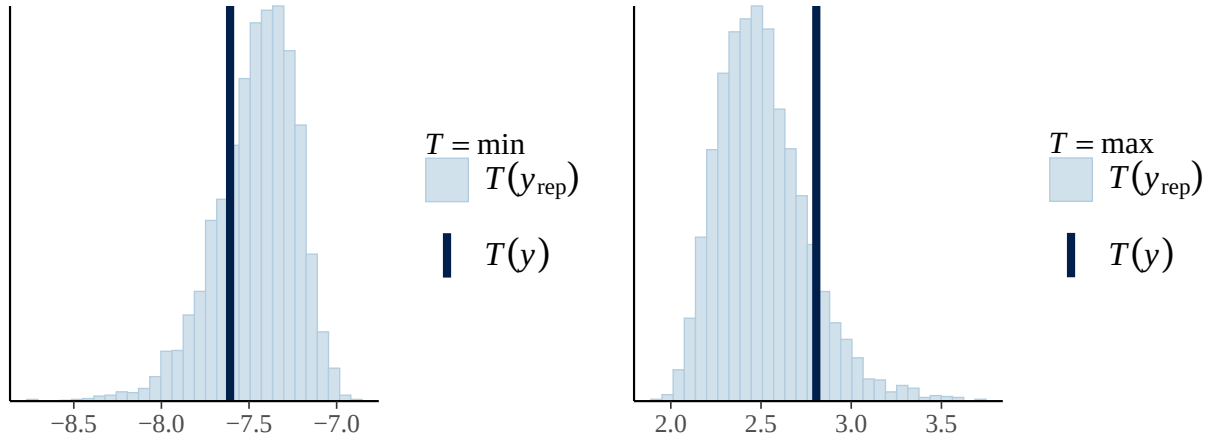
Examining the comparison plot of p, we see that the posterior contracts right in the centre of the prior. This is exactly what we expect to see from a weakly informative prior, as we have no evidence of prior-data conflict.

From the comparison plots in this section, we see that all of the posterior distributions contract within the priors and close to the centres of the priors, which all have long tails that are characteristic of our weakly informative priors. Thus, we evidence that we have justified priors, and more importantly, we have evidence that the model fits the data well.

Posterior predictive distribution/Evaluation of test statistics

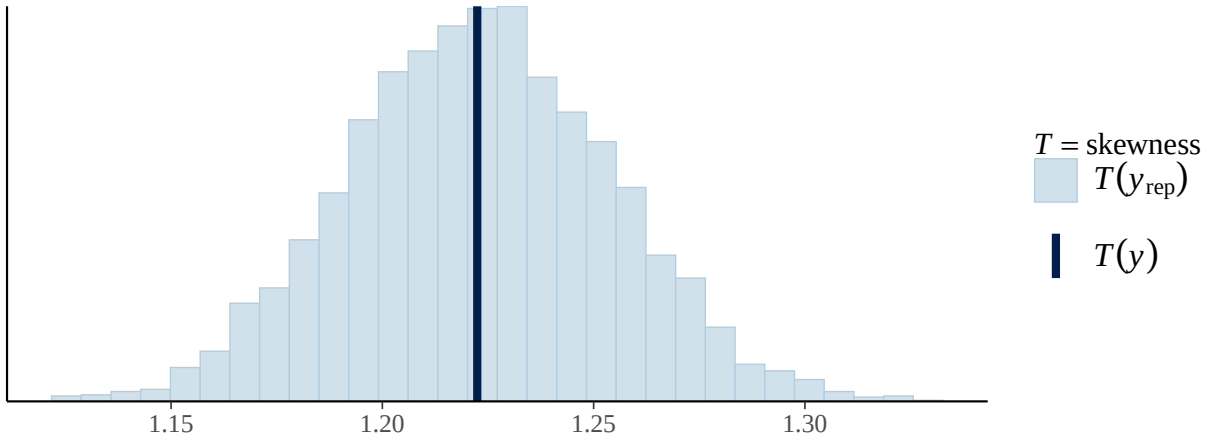


Our posterior predictive distribution plots are exactly how we'd expect them to look: bimodal with a lot less density on the second component. Since we had relatively wide priors, we also expect to see these posterior predictive distributions with relatively long tails too. Next we'll examine some test statistics to evaluate the model fit:

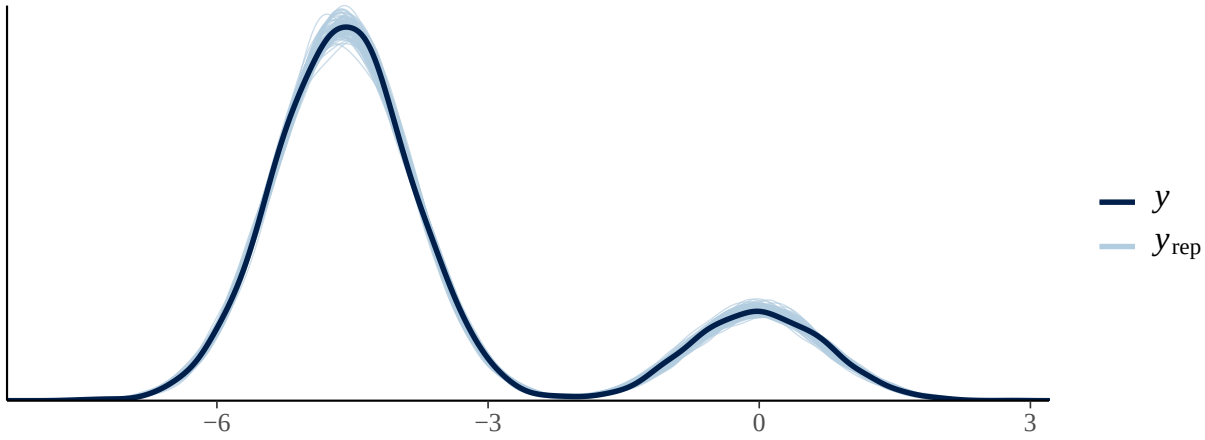


In both of the `min` and `max` test statistics plots we see that the test statistic line is very close to where most

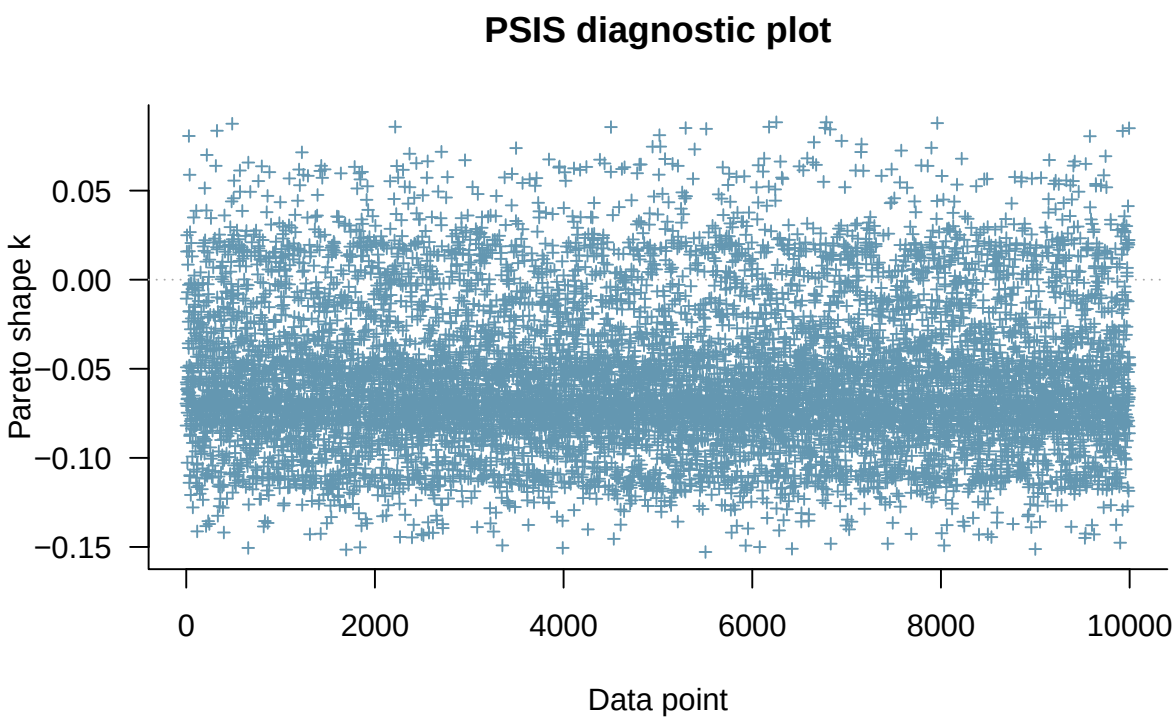
of the mass of $T(y_{rep})$ distribution is concentrated. In other words, the model was fully able to capture the `min` and `max` test statistics.



The observation we saw with the `min` and `max` test statistics held with the `skewness` test statistic, i.e., $T(y)$ is right in the centre of the $T(y_{rep})$ distribution. So the model was able to capture all three ancillary test statistics we use with our normal distribution.



Next, from the PPC density overlay, we see that the model was able to faithfully model the behaviour of the simulated data exactly, matching the density of any given value from the simulated data extremely well. In particular, the model was able to recreate the bimodal behaviour of the simulated data.



From the PSIS plot, we see that all the $\hat{k}$ values were well below 0.5, and so there were no influential points due to the model fit. Taking into account our previous analysis and this PSIS plot, we see that there is no concerning evidence of the model overfitting/underfitting to portions of the data, and so this model seems to have decent predictive performance as measured using LOO-CV. As always with interpreting $\hat{k}$ values, these aren't necessarily indicative of the exact predictions or how "close" the predictions are to the observed data.

Model Fit conclusions/sampler comments

Are we surprised by the extremely good fit this model has on the simulated data? No. Not at all. We were tasked with coming up with an experiment that matched the problem description, i.e., a raptor having a low and high energy state. We were given essentially no background knowledge about raptor energy states, rather being instructed to approach this from a conceptual manner. As a person with very little animal ecology knowledge (and even less avian knowledge), I approached this as best I could, but ultimately from only one uninformed point of view, meaning that both the data simulation experiment and modelling process came from one place. So when I put priors on the model parameters that were largely based off of the experiment parameter values, the model was essentially given enough information to recreate the "true" simulated distributions, and thus, it's no surprise the model fit the simulated data so well. It is worth noting that I did not decide to approach the simulation/modelling process by looking at the sample and making inferences about our raptor energy states that way because it would make our priors and experiment data-dependent, which was Prof. Simpson strongly cautioned against at the start of the course.

Since the model priors were essentially based off of the experiment's distribution parameters, we didn't expect the priors to be too constrained nor did we expect sampler problems to arise from issues like misspecified priors or a model misfit. Indeed we had no sampler/computational issues during the sampling phase and only very occasional proposal rejections in the warmup phase, which often happens in most models we've seen in this course.

Task 2: Fitting the data

In task 2 we are tasked with evaluating a new model and comparing its performance to our model with a fixed p from task 1 on real data (not used in determining appropriate priors):

Table 1: data used to fit models

date_time	msa	wind_speed	saws_temp	hrseg
2013-04-16 13:03:44	0.2378753	3.6	16.5	1
2013-04-16 13:05:38	0.0922250	3.6	16.5	1
2013-04-16 13:07:27	0.2739469	3.6	16.5	1
2013-04-16 13:37:51	0.2877608	3.6	16.5	5
2013-04-16 13:39:44	0.5984821	3.6	16.5	5
2013-04-16 13:41:36	0.1246620	3.6	16.5	5

This new model will use a logistic model to model p, the probability of the high energy state, where the predictors for the logistic model are: `wind, temp`. We will refer to this model as the **task 2 model**, while we will refer to the model with the fixed p as the **task 1 model**. Mathematically, we represent our new model as:

$$\begin{aligned} \text{logit}(p_i) &= \beta_0 + \beta_1 \texttt{wind} + \beta_2 \texttt{temp} \\ Z_i &\sim \text{Bernoulli}(p_i) \\ y_i \mid Z_i, \mu, \sigma &\sim N(\mu_{Z_i}, \sigma_{Z_i}) \end{aligned}$$

Therefore, in addition to the priors on our mixture components, we need additional priors on $\beta_0, \beta_1, \beta_2$ for this model. We will still need priors on the parameters for our two Gaussian components. However, since we're not given additional domain knowledge in this section to help us construct more informative priors, we will use the same priors determined in task 1 for these components. We do so because those priors are what I determined to be sensible for these two components given my knowledge about our problem. They were additionally not data-dependent priors in that I did not construct them based off of our sample, rather constructed conceptually.

So then, we'll focus on deriving priors for our logistic model. First we'll put a prior on β_0 then β_1 and finally β_2. We will put normal distribution on our priors since by the CLT it is sensible to have normally distributed priors. In setting priors, we will use the logistic function (inverse logit) to help examine the effects of varying parameter choices on our priors. Recall that the logistic function with Z as our linear function is:

$$\text{logistic}(Z) = \frac{\exp^Z}{\exp^Z + 1}$$

As we want to put a prior on β_0, we will consider that we are attempting to model the probability of being in a high energy state. Since β_0 is the intercept, logistic(β_0) would give us the probability of a high energy state if temperature and wind speed were both 0. In such a scenario, it would be sensible that the bird is trying to conserve as much energy as possible. But perhaps it is also probable that the bird is attempting to leave the environment and is expending a lot of energy to do so. In other words, there's no definitive association between cold temperatures, no wind, and bird activity, but it is more likely that the bird is trying to conserve energy. Given our uncertainty but our leaning towards a low energy state in such an scenario, we will set the prior mean to be the solution of the logistic equation when $p_i = 0.35$, a 35% chance that the bird is trying to expend a lot of energy for whatever reason. However, we will put a large standard deviation on this prior to

reflect our uncertainty. A sensible lower "bound" (5^{th} percentile) is the solution to the logistic equation when $p_i = 0.1$, a 10% chance that the bird is trying to expend a lot of energy. Computing these two numbers, we get that the solution to the logistic equation when $p_i = 0.35$ is -0.619039, and when $p_i = 0.1$, we get -2.19722. So if we have $\mu_{\beta_0} = -0.619039$ then we find $\tau_{\beta_0} = 0.789093$.

Next, we'll put a prior on β_1 the coefficient for wind speed. Again it is unclear how wind speed may affect the probability of the bird being in a high/low energy state as it may just be doing various actions. For example, it may be that given a certain wind speed the bird is just sleeping, requiring very little energy, or that the bird is flying furiously to get away from larger bird, requiring a lot of energy. So with our very limited domain knowledge, we posit that there is very little association with either energy state, so we will in fact centre this prior distribution around 0 with a relatively large spread. Then for the standard deviation, we will consider the interpretation of coefficients in logistic regression. That is, we consider the coefficients in logistic regression to be log-odds. Thus if we choose our standard deviation to be 0.5, we say that it is unlikely for $\beta_1 > 1$. So it follows that it is unlikely for the odds of a high energy state to increase multiplicatively by more than $e \approx 2.71828$ for an increase in one meter per second of wind speed, which is sensible as it reflects that it could potentially require expoentially more energy to move due to increasing wind speed but 2.71828 as the multiplier doesn't seem to increase the odds too fast (or too slow), reflecting our lack of domain knowledge encoded in our weakly informative priors. To recap, we put a $N(0, 0.5)$ prior on β_1

Finally, we'll put a prior on β_2 the coefficient for temperature. Like wind speed, it's unclear how temperature may affect the probability of the bird being in a high/low energy state as it may just be doing various actions. For example, it may be that given a certain temperature, e.g., day time temperature, the bird is just resting after a meal, requiring very little energy, or that the bird is out hunting, requiring a lot of energy. So with our very limited domain knowledge, we posit that there is very little association with either energy state, so we will in fact centre this prior distribution around 0 with the same standard devation as our prior for β_1, i.e., 0.5. We do this because having the odds of a high energy state increase multiplicatively by 2.71828 for every degree increase in temperature seems reasonable. To recap, we put a $N(0, 0.5)$ prior on β_2.

To summarise, we will use the following priors for the logistic model for p_i:

1. $\beta_0 \sim N(-0.619039, 0.789093)$
2. $\beta_1 \sim N(0, 0.5)$
3. $\beta_2 \sim N(0, 0.5)$

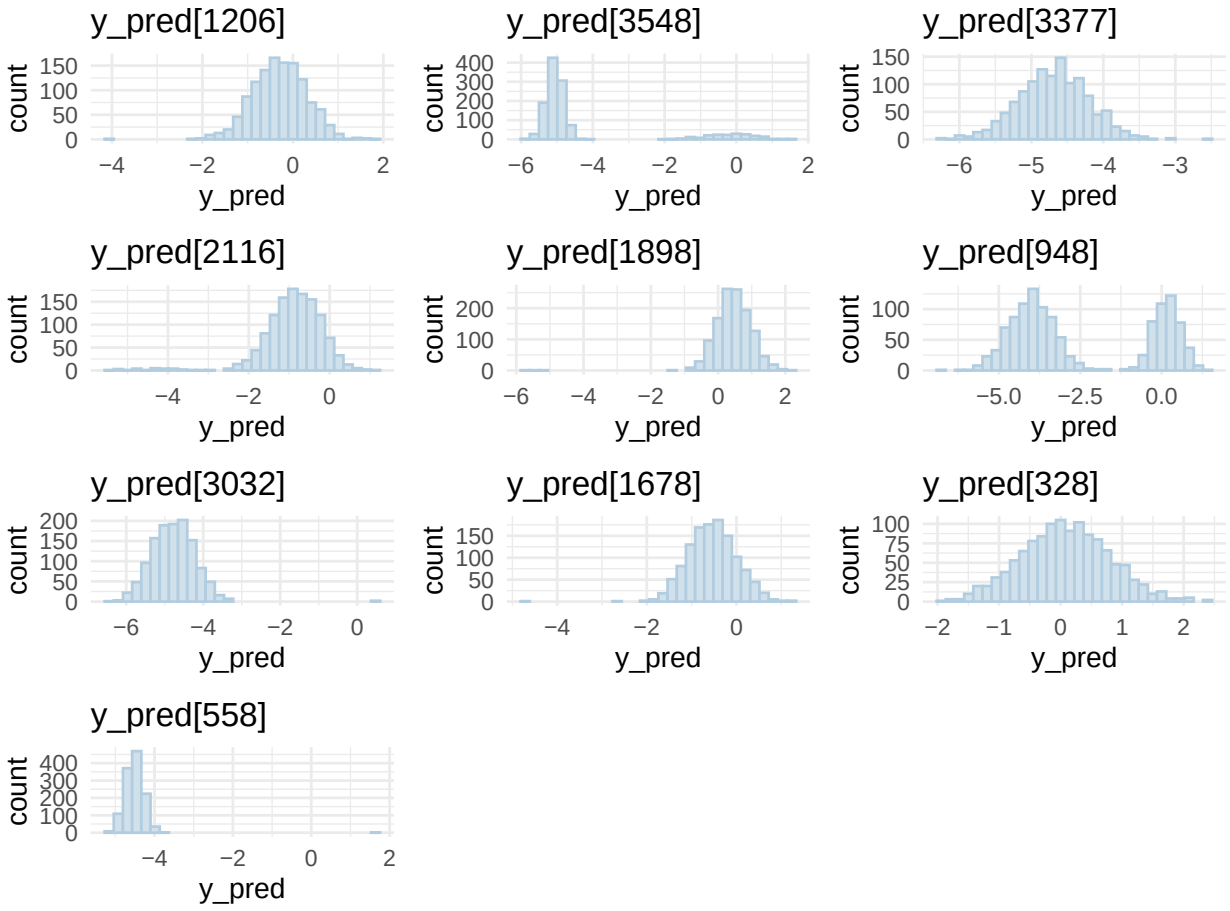
Additionally, we will use the following priors for the mixture model (copied from task 1):

1. $\mu_1 \sim N(\log(0.01), 0.35)$
2. $\sigma_1 \sim N_+(0.5, 0.125)$
3. $\mu_2 \sim N(\log(1), 0.35)$
4. $\sigma_2 \sim N_+(0.5, 0.125)$

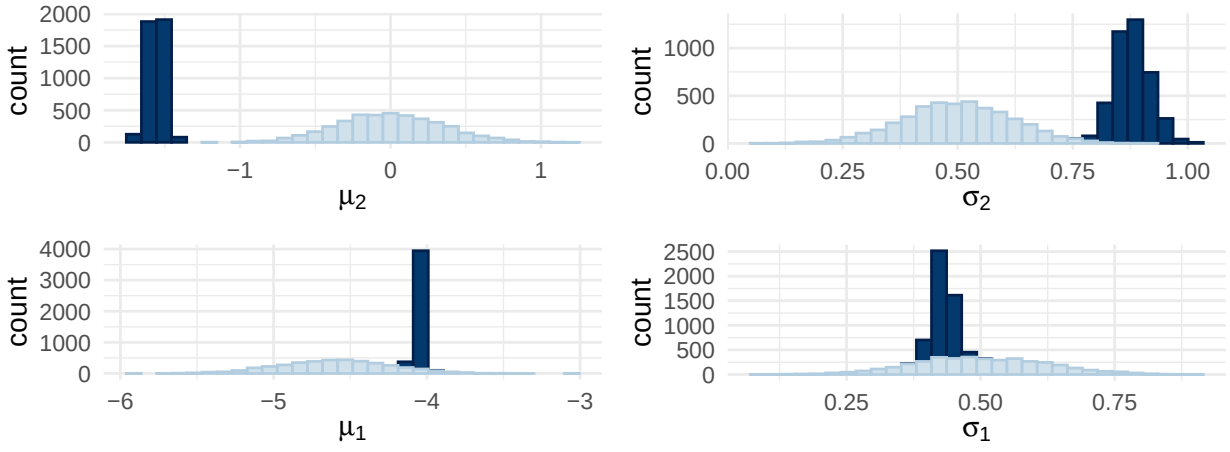
Prior predictive checks/Comparison of prior and posterior distributions

Here we will perform some basic prior predictive checks for both models and an evaluation of our priors in comparison to the posteriors. Note that for the comparison plots, the priors are the lighter blue distributions, and the posteriors are the darker blue distributions.

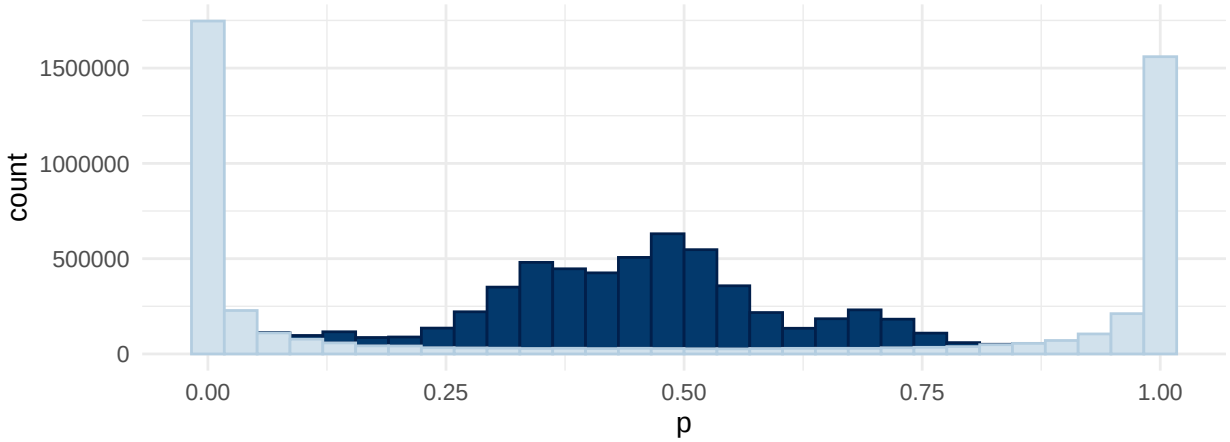
Evaluation of the task 2 model



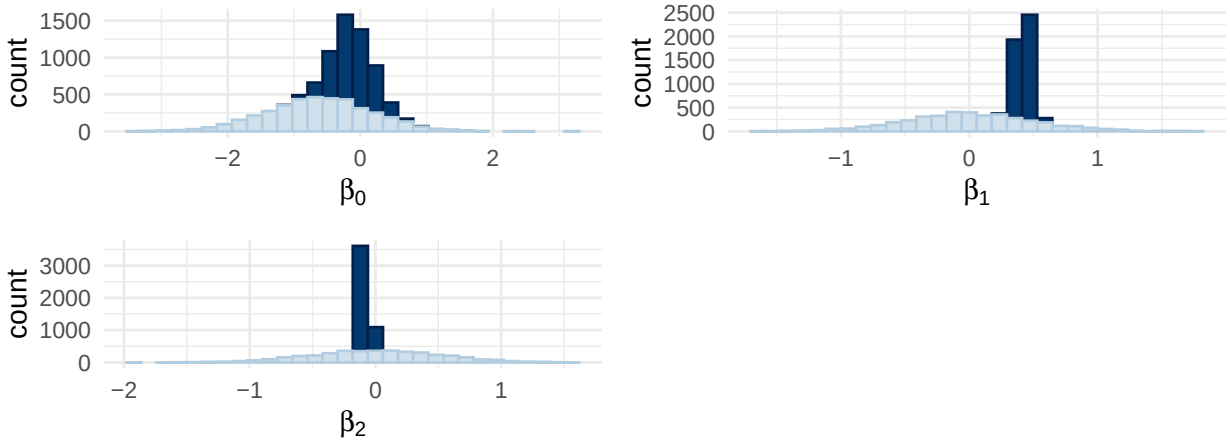
From the prior predictive distribution, we see both a mix of very clearly bimodal mixtures and mixtures that are essentially one normal distribution. At first, glance this is somewhat surprising, but after some careful considerations, the distributions that are essentially one normal distribution are possible given our priors since it is possible to sample a value for μ_1 from the upper tail of that prior and a value for μ_2 from the lower tail of that prior and some other values of σ_1, σ_2 that make it so that the two distributions are essentially overlapping. So there's no clear evidence of horribly misspecified priors here.



From the comparison plots for the priors/posteriors on the two mixture components, we see that our posterior distributions (darker blue) generally contract within their respective priors (lighter blue). The one exception is μ_2 where the posterior shows some signs of prior-data conflict as the posterior is located where there is little to no prior mass. This is likely due to our lack of domain knowledge that would allow us to set a more informed centre for the prior of the high energy component. Overall, our priors don't seem misspecified, and we have promising signs that our model was a good fit for the data, especially given the behaviour of the posteriors for μ_1 and σ_1, where they contracted a lot and close to the centre of the prior distribution.



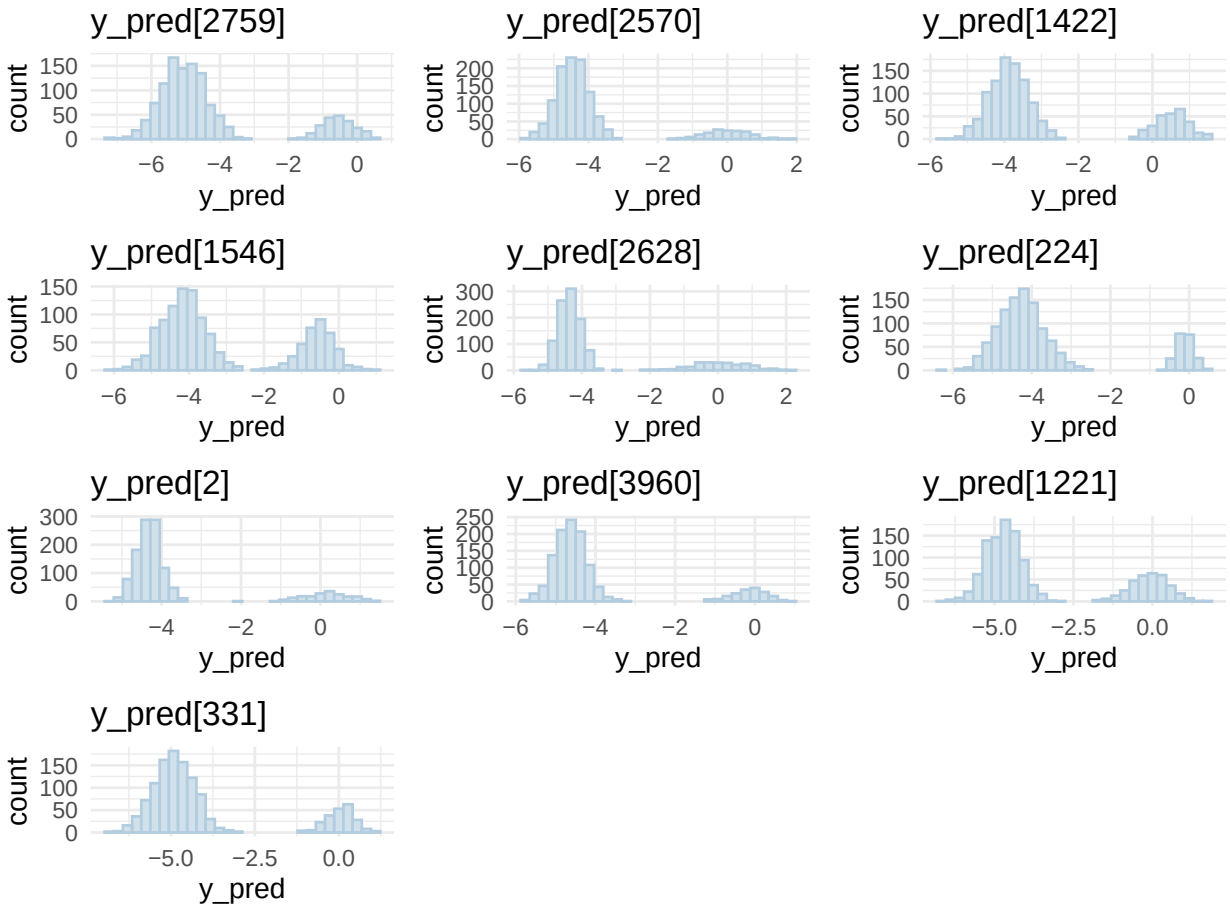
The plot above is comparison plot of p, the probability of the raptor being in a high energy state. While our model for p dictates that we get a different p for each observation, we have plotted the marginal distribution for p, marginalized over all observations. We do this to get an overall sense of the probability of the raptor being in a high energy state. We see that the prior distribution interestingly somewhat resembles the shape of a horseshoe prior with the mass being placed on both extremes of the distribution. Contrarily, the posterior contracts to the centre of the [0,1] interval, placing a lot of mass above 0.5 and implying that the chance of the raptor being in a high energy state is equiprobable as being in a low one. Although, the posterior distribution has mass over essentially the entire [0,1] interval, so we don't really have evidence of prior-data conflict.



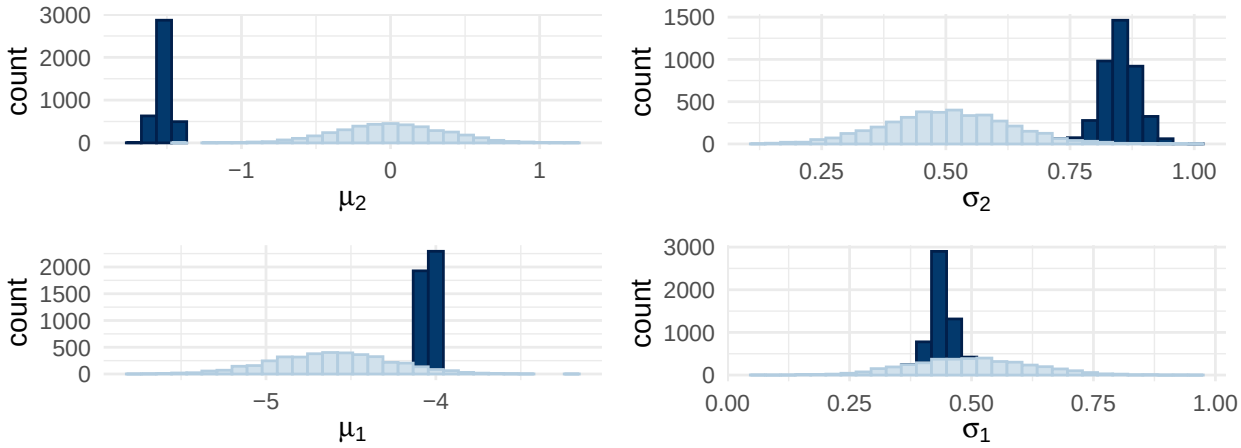
Finally, from the examination of the parameters for the logistic model, we see behaviour we'd expect from well specified weakly informative priors. That is, we see the posterior contracting within the priors and close to the centre. It is of note that β_1's posterior only assigned mass on positive numbers, which will factor into our analysis for model comparison later. But regardless, we have evidence of a well fit model and justified priors, especially since we have no evidence of prior-data conflict.

Overall, our model seems to be a decent fit so far for our data, despite the evidence of prior-data conflict with μ_2. In particular, most of the comparison plots show what we'd expect from not misspecified weakly informative priors.

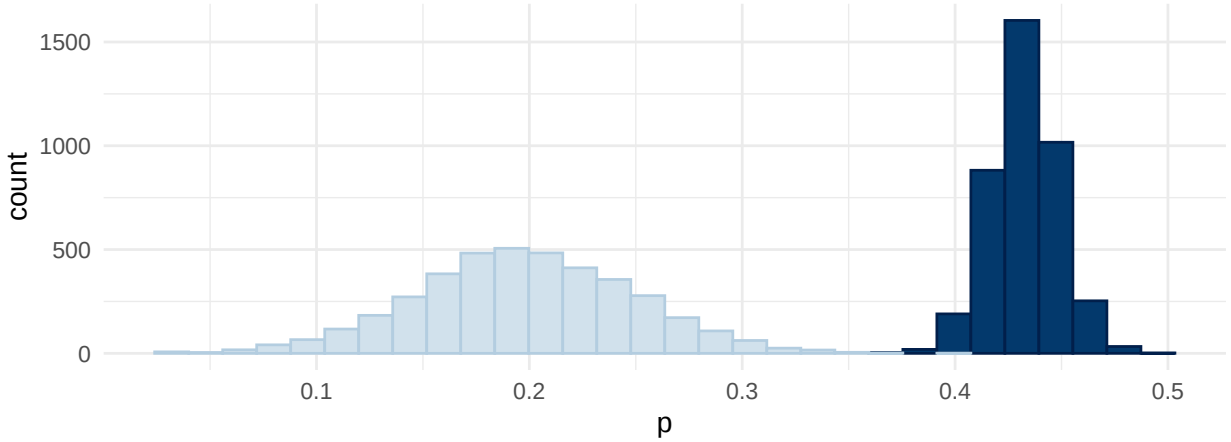
Evaluation of the task 1 model



From the prior predictive distribution, we see very clearly bimodal mixtures distributions, as we'd expect from how we specified our priors and from our results in task 1. So we have evidence that our priors were not really misspecified.



From the comparison plots for the priors/posteriors on the two mixture components, we see that our posterior distributions (darker blue) generally contract within their respective priors (lighter blue). The one exception is μ_2 where the posterior shows some signs of prior-data conflict as the posterior is located where there is little to no prior mass, similar to the previous model. It follows that we see similar behaviour for μ_2 for both models because we lack domain knowledge to make more informative priors. Overall, our priors don't seem misspecified, and we have promising signs that our model was a good fit for the data, especially given the behaviour of the posteriors for μ_1 and σ_1, where they contracted a lot and close to the centre of the prior distribution.

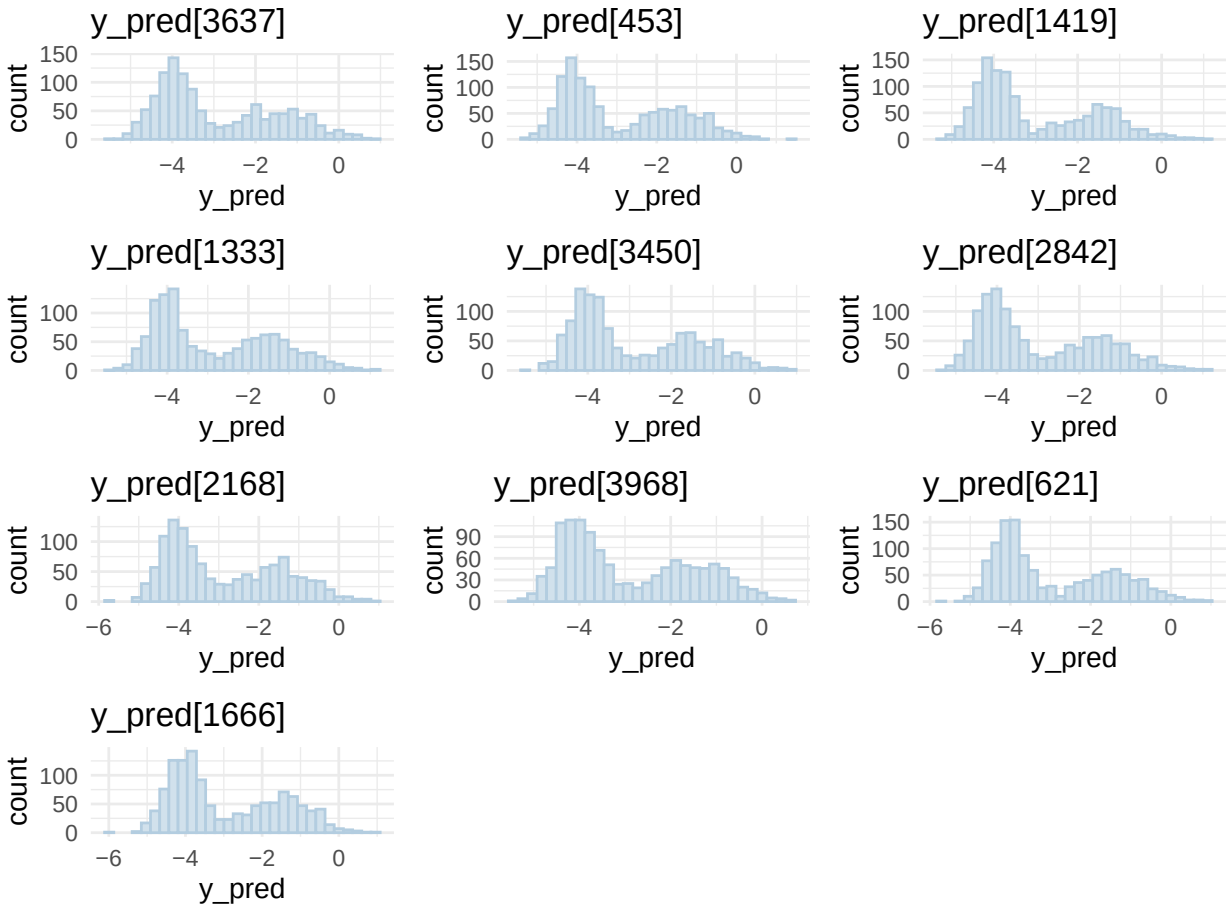


We see from the comparison plot for p that we have some evidence of prior-data conflict because the posterior contracts where there is minimal prior mass, and perhaps evidence that our model isn't a great fit for the data. It may also be due to our prior for μ_2, which clearly specified the centre for the high energy component to be too "high". Although overall, we don't have much evidence of misspecified priors, and our model is still possibly a good fit for our data, even if clearly it's not an amazing fit.

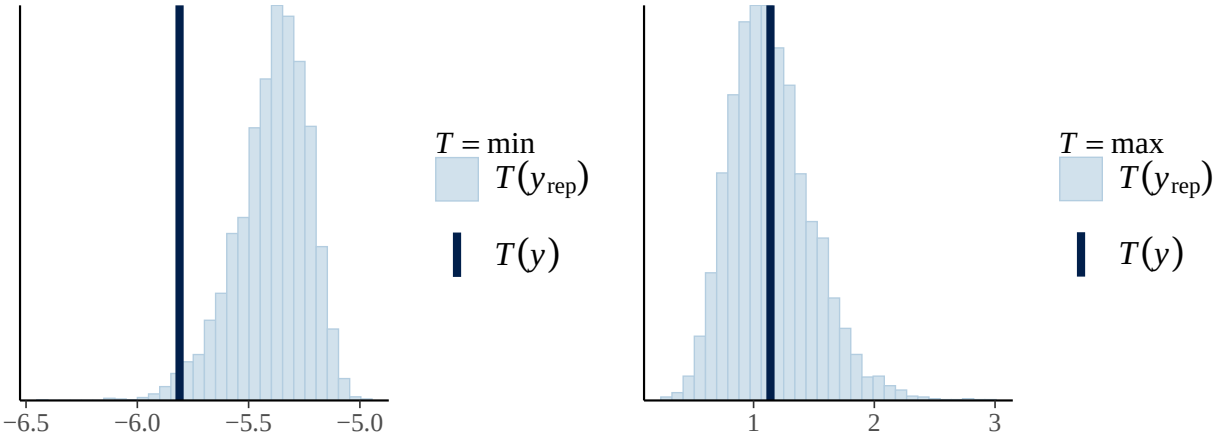
Posterior predictive distribution/Evaluation of test statistics

In this section, we'll evaluate the posterior predictive distributions for both models, and comment on how these two models fit the data. Like the section above, we will refer to the histogram of MSA values from the dataset we're given for this section, which is in the appendix under "additional plots".

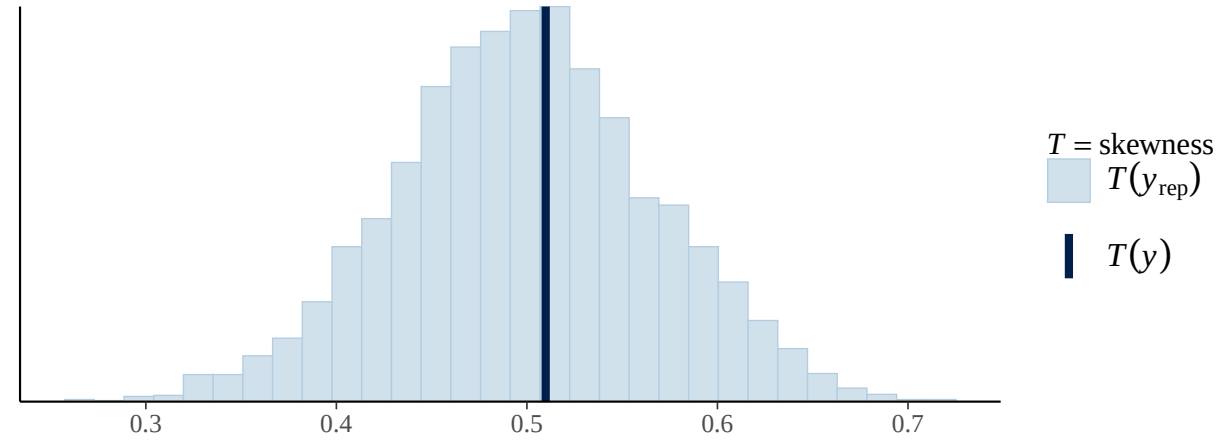
Evaluation of the task 2 model



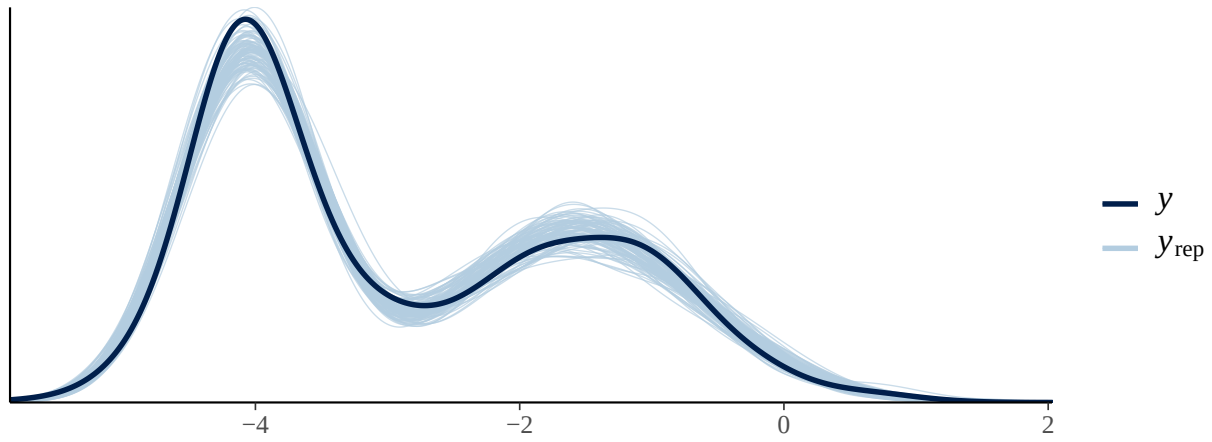
From the 10 randomly sampled posterior predictive distributions, we see that our posterior predictive distributions are all bimodal with two components and less mass on the high energy component. Additionally, the posterior predictive distribution comes out to resemble our data fairly well (seen in the appendix), with the observation that our predictive distributions assign more mass to the tails and especially around values close to 0 in comparison to the histogram of the data. The wider tails is expected behaviour since we had wide priors, and we don't expect a perfect fit. Overall our model seems to fit the data fairly well and has resonable predictive performance on our dataset. We will then examine some ancillary test statistics to explore our fit a bit more.



From the `max` and `min` test statistics, we see that the model was able to capture both test statistics with the model being able to capture the `max` test statistic very well and less so the `min` test statistic. This again gives us evidence that our model was a decent fit, with perhaps being less capable in predicting values in the lower extremities of our dataset.

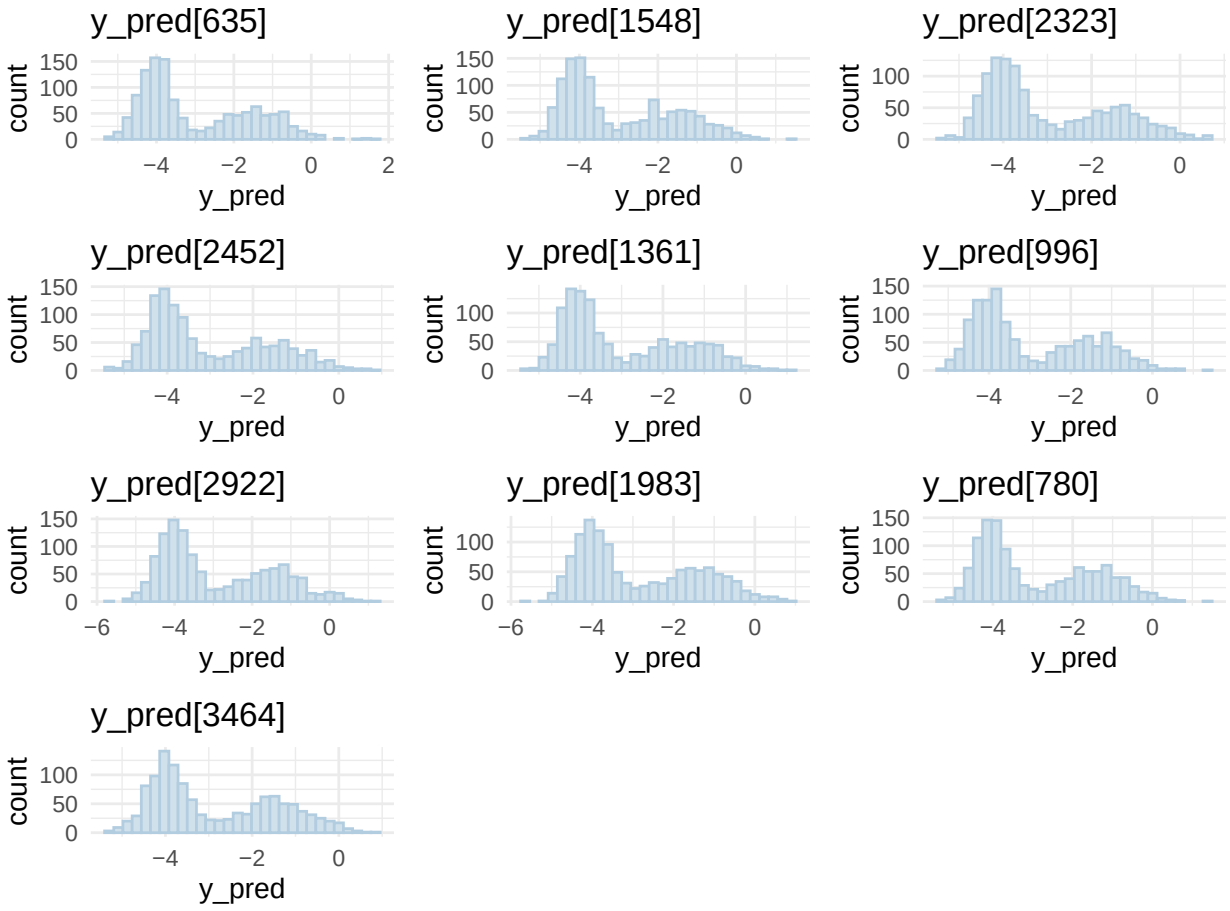


From the plot of the `skewness` test statistic we see that the model captured this test statistic extremely well, having $T(y)$ being almost exactly in the middle of the $T(y_{rep})$ distribution. Thus, we see more evidence that our model fit the data well.

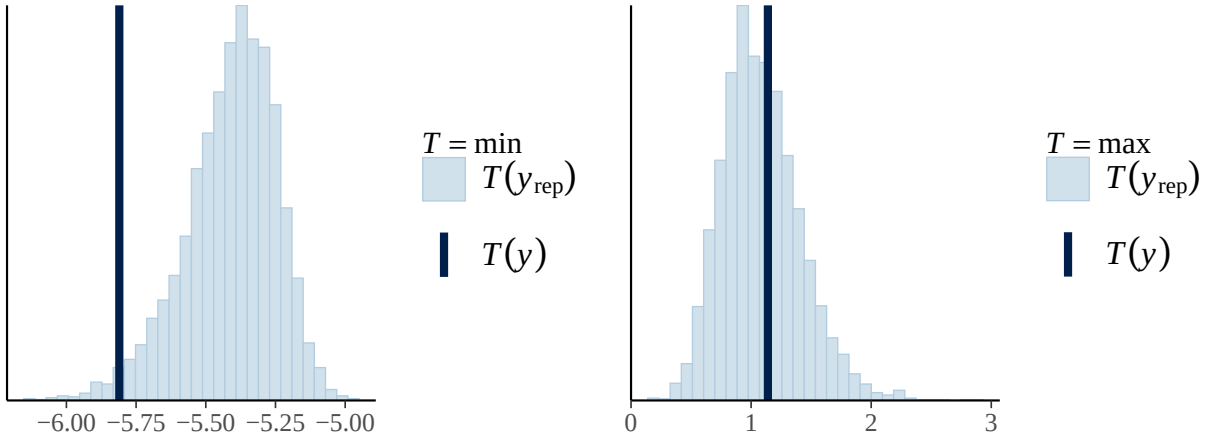


Finally, looking at the ppc density plot, we see that the density estimates of the predicted distributions largely mirror the curve of the density estimate of the observed data. Note that we're considering the overlaid plots of the various y_{rep} curves to be a sort of marginal density by ignoring the various wind and temperature conditions that individual y_{rep} curves are conditioned on, and doing so allows us to consider the performance of the model more holistically. Since the y_{rep} curves replicate the behaviour of the observed data so well, e.g., having a clear bimodal shape and replicating the density at dips and peaks so faithfully, we have strong evidence that our model fit the data quite well.

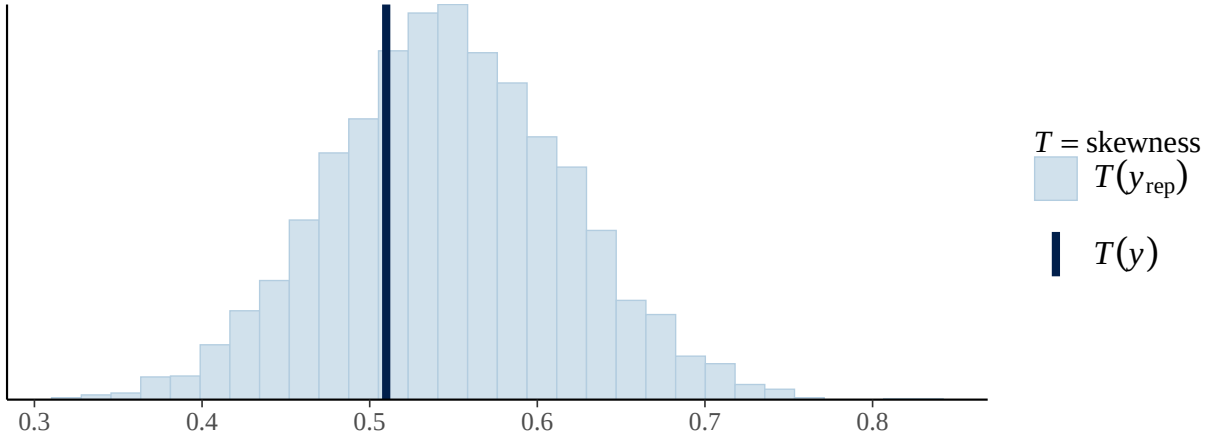
Evaluation of the task 1 model



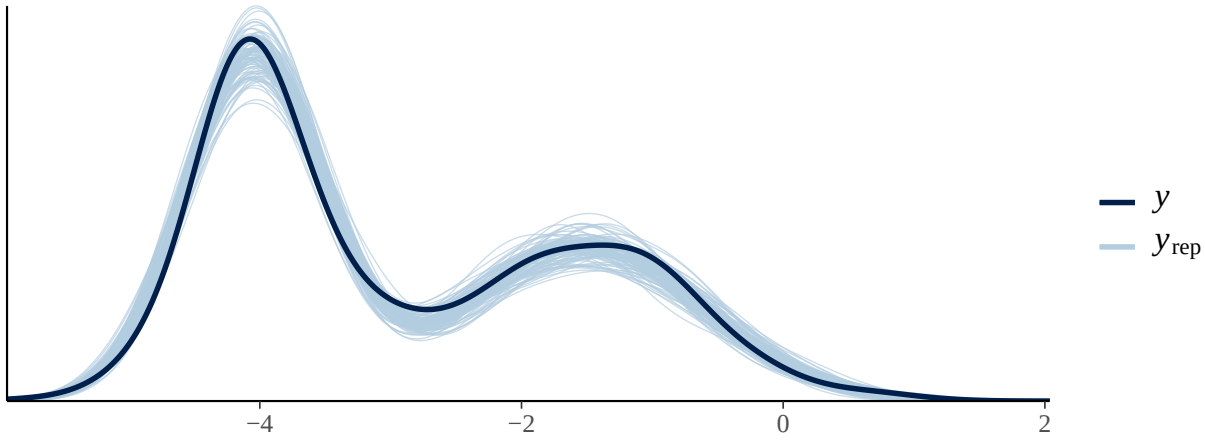
From the 10 randomly sampled posterior predictive distributions, we see that our posterior predictive distributions are all bimodal with two components and less mass on the high energy component, and they resemble those of the task 2 model quite similarly. One major but somewhat subtle difference is that the tails of the two mixture distributions are actually wider than those of the task 2 model and those of the observed data. Again, the wider tails is expected behaviour since we had wide priors, and we don't expect a perfect fit. Overall our model seems to fit the data fairly well, and it has reasonable predictive performance on our dataset, but possibly worse than the task 2 model due to longer tails. We will then examine some ancillary test statistics to explore our fit a bit more.



From the `max` and `min` test statistics, we see that the model was able to capture both test statistics with the model being able to capture the `max` test statistic very well and less so the `min` test statistic. This again gives us evidence that our model was a decent fit, with perhaps being less capable in predicting values in the lower extremities of our dataset similar to the task 2 model.



This model performs decently well in capturing the `skewness` test statistic, i.e., the test statistic is close to the centre of the distribution. Although, it didn't capture it as well as the task 2 model did.



Similar to the task 2 model, this model replicated the density estimate curve of the observed data quite well, being able to replicate behaviours of such as the bimodality and the dips and peaks. It is worth noting that this model was clearly a bit worse than the task 2 model in replicating the density of the dip around -3, where the model had less density there than the observed data and the task 2 model.

So we may have evidence that while this model was a good fit, it was mot quite as good as the task 2 model. In the next section, we'll examine the PSIS plots and results from LOO-CV.

Comparison of Models

First we'll take a look at the PSIS plots for both models to examine if there were any particularly influential points.

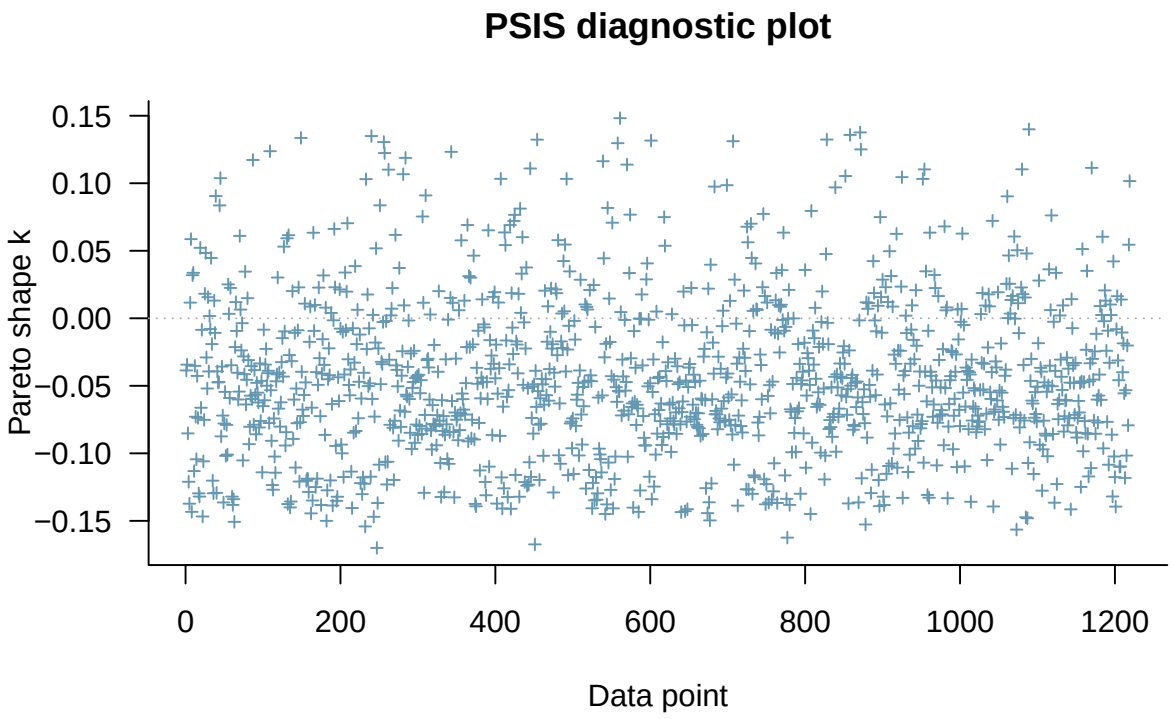


Figure 1: Task 1 model PSIS plot

From the PSIS plot of the task 1 model, we gather that there were no particularly influential points in the data as all $\hat{k}$ values were well below 0.5. Further, we have evidence that the LOO predictive distributions should replicate the full predictive distribution. Thus, we gather more evidence that the task 1 model as a good fit for the data.

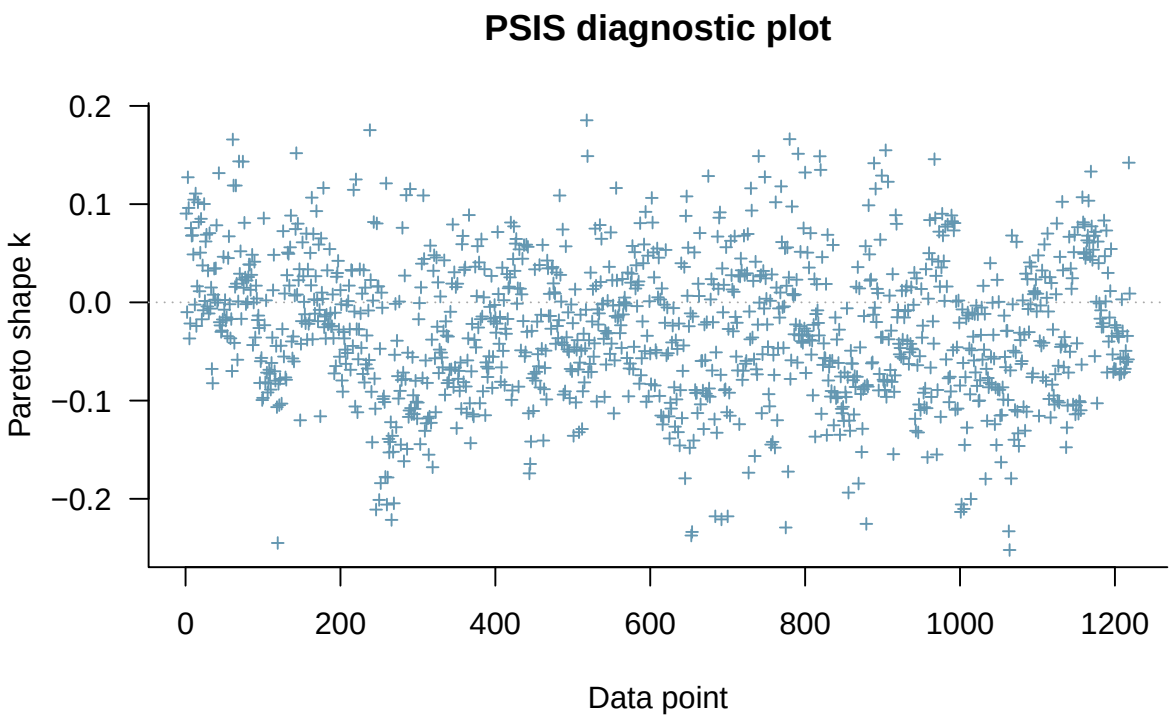


Figure 2: Task 2 model PSIS plot

Examining the PSIS plot for the task 2 model, we find that like the PSIS plot for the task 1 model, we have no influential points in the data (all $\hat{k}$ values are under 0.5). It is worth noting that while some of the $\hat{k}$ values were higher than those of the task 1 model as many are above 0.2, we can't conclude generalization performance from this as the model clearly hasn't overfit to certain parts of the data. Additionally, like the task 1 model, the LOO predictive distributions should resemble those of the full predictive distribution.

It is additionally worth noting that our $\hat{k}$ values for both models should clear up our concerns over potentially bad predictive performance on the lower extremities of the data. Thus, we'll look to the LOO-CV comparison to determine which model was a better model.

Table 2: Task 1 model versus task 2 model

	elpd_diff	se_diff	elpd_loo	se_elpd_loo	p_loo	se_p_loo	looic	se_looic
model2	0.00000	0.000000	-1812.309	29.39274	8.151553	0.4604867	3624.618	58.78548
model1	-40.60314	8.983848	-1852.912	28.73543	5.738143	0.4231169	3705.824	57.47085

Here we see that using LOO-CV, we find that model 2, our task 2 model, was the better performing model as it had a higher ELPD: -1812.309 versus -1852.912. Since the standard difference is 8.983, we have some evidence that the task 2 model is significantly better than the task 1 model, but generalization performance is never guaranteed. Although, since LOO-CV is a fairly good indicator of generalization performance, evidence points to the task 2 model as being the better predictive model. In the context of our other analysis, we have evidence pointing to the task 2 model as also fitting the data better. For example, it was better at replicating the dip in density around -3, and it generally captured the test statistics better. Thus, taking all of this into account, we conclude that the task 2 model was just a better model, fitting the data better (less bias) and

having better generalization performance (less variance) than the task 1 model. So having p_i be predicted using `wind` and `temp` as predictors leads us to capture more of the variation in the data, and can lead us to more accurate predictions than simply having a fixed p regardless of environmental conditions.

To further interpret our choice, we'll approach this from a higher level. From a high level, our conclusion is also sensical because having p_i be predicted by `wind` and `temp` should capture more of the variation in the data since it's likely that these two predictors influence what raptors do given certain conditions. In other words, environmental conditions seem to affect the energy state raptors are in, and thus, how raptors move. We also find evidence to support this "theory" as the posterior for β_1 had its mass completely above 0 and the posterior for β_2 had most of its mass above 0; there's definitely some association between these predictors and the probability of the raptor being in a high energy state. Overall, we conclude that taking environmental conditions into account lead us to more accurate predictions of raptor movement because it seems that they influence raptor energy states by having an association the probability of a raptor being in a high energy state.

Task 3: Spurious components

In task 3 we are asked to extend our fixed p model to a new scenario where we are uncertain if there are two or three energy states. That is, we will construct a model with a prior that puts a lot of mass on the two other components. We'll use much of the same procedure/approach as we did with task 1, and in fact, we'll use task 1 as a starting point. So in task 3, we'll aim to focus on what is new in task 3.

Starting off, we'll set up our experiments again. This time we'll suppose that there's an extremely high energy state that is actually separate from the current high energy state. We'll start from the parameters of the experiment we conducted in task 1, that is:

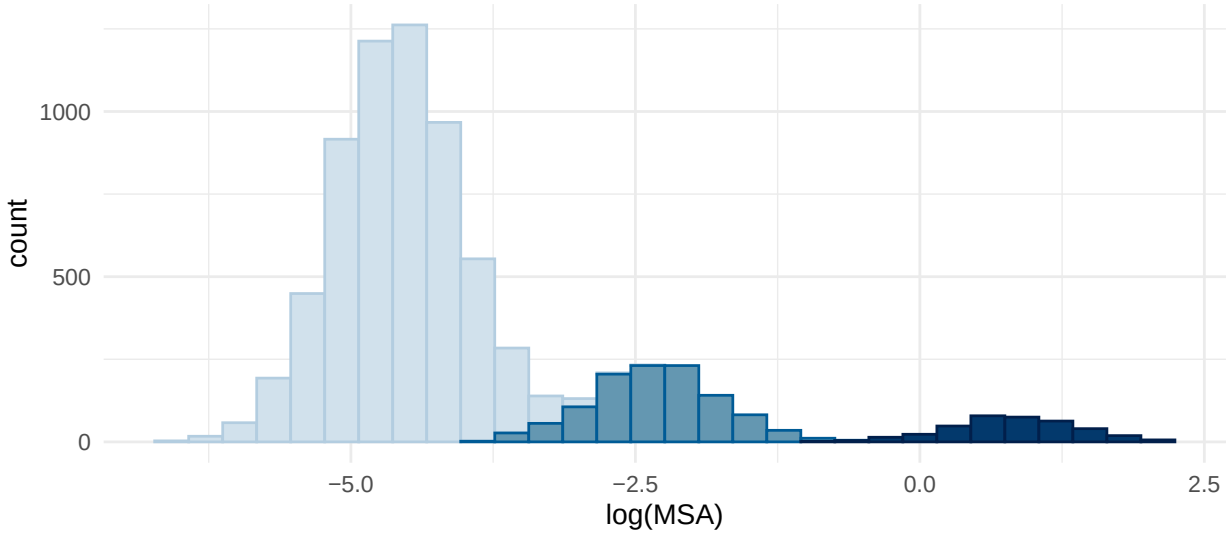
1. High energy component: $N(\log(0.01), 0.75)$
2. Low energy component: $N(\log(1), 0.75)$

Since we are still tasked with simulating from a model with two components, we will still use the experimental parameters from task 1 for simulating data in the 2 component case. For the 3 component case, we will instead split the high energy component. We will assign the new component to be located at approximately the 68^{th} percentile of old high energy component, i.e., plus one standard deviation so 0.75. We will also shift the centre of the old high energy component down to $\log(0.1)$, and we'll decrease the standard deviations from 0.75 to 0.55 for all of the components, to better reflect the clear difference in components while still covering the range of the components from task 1 did. In other words, this experiment still posits the same MSA range for this raptor that the task 1 experiment did, but just assign different mass to the values on this range.

Since we have an extra component, we'll also assign 3 new probabilities to each of the three energy states, where we again use the task 1 component probabilities as a starting point. In task 1 we assigned a 20% chance of the raptor being in a high energy state, and instead, for this task, we split that 20% up to 15% for the new lower high energy component and a 5% for the extremely high energy component. These probabilities are sensible since we'd still believe the raptor spends 80% of its time in a low energy state, while only spending 5% of its time ever in a extremely high energy state such as fleeing for its life or fighting hard to kill its prey. We then attribute the remaining 15% to the raptor spending its time in a still relatively high energy state (just not overly so), such as flying, hunting easy prey, or just any other activity that requires a fair bit more energy than low energy activites like preening or sleeping.

Like task 1, the code below shows the three component experiment setup and results using parameters we define above:

```
gaussian_params <- matrix(NA, nrow = 3, ncol=2)
gaussian_params[1,] <- c(log(0.01),0.55)
gaussian_params[2,] <- c(log(0.1), 0.55)
gaussian_params[3,] <- c(0.75, 0.55)
simulated_data_t3 <- simulate_gussian_mixture(7500, c(0.8, 0.15, 0.05), gaussian_params)
```



Note that in the simulation results above, the lightest blue is the low energy component, the middle shade of blue is the higher energy component, and the darkest blue is the highest energy component. Similarly to task 1, if we consider the marginal distribution, we see that indeed most of the mass is centred around $\log(0.01)$ as we'd expect, and the distribution is trimodal.

Next, recall that we have our fixed p model defined as:

$$\begin{aligned} Z &\sim \text{multinomial}(1, \boldsymbol{\pi}) \\ y_i \mid Z, \mu, \sigma &\sim N(\mu_Z, \sigma_Z) \end{aligned}$$

Note that above $\boldsymbol{\pi}$ is the vector of mixing proportions.

Since we consider 3 components with this model, we will instead have $\boldsymbol{\pi}$ be a 3-dimensional vector with the simplex constraint, i.e., $\sum_{k=1}^{K} p_k = 1$, where $K = 3$ in our case. Then the question becomes: what prior would we put on $\boldsymbol{\pi}$?

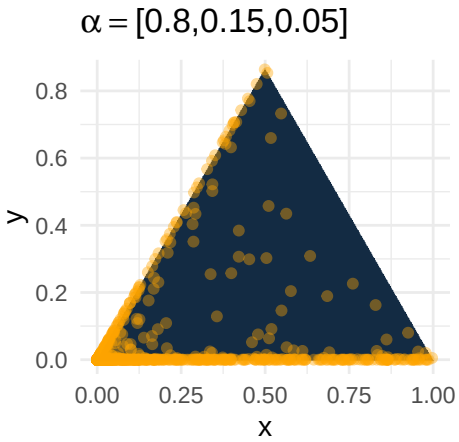
First, in a similar vein to considering how much mass to assign to the ends of a horseshoe prior, we can think about how to construct a 3-simplex that our assign a lot of mass onto the two lower dimensions. A distribution that would let us do this is the Dirichlet distribution, which is parameterized by a vector of reals, $\boldsymbol{\alpha}$, and it has a nice expectation for the probability of class/component k in the form of:

$$\mathbb{E}(p_k) = \frac{\alpha_k}{\sum_{k=1}^{K} \alpha_k}$$

So then, using the expectation of p_k formula, we can set each $\alpha_k < 0.1$ such that $\sum_{k=1}^{K} \alpha_k = 1$ for a couple of reasons.

First if we use these two conditions for α_k then it follows that we can consider each α_k to be a probability, i.e., $\mathbb{E}(p_k) = \alpha_k$. We believe this should work because approaching α_k as a probability is analogous to our process for setting priors in part 1, where we also had a fixed p. Namely, we posit a certain proportion of time the raptor spends in a certain energy state and assign that to be the probability of an energy state. Thus, we can easily encode our "beliefs" about the raptor energy states into $\boldsymbol{\alpha}$, the parameter for our Dirichlet prior. Since we believe that it's sensible for the raptor to spend 80% of its time in a low energy state, 15% of its time in a higher energy state, and 5% of its time in an extremely high energy state. So we have that $\boldsymbol{\alpha} = [0.8, 0.15, 0.05]$.

Our second reason is because having the alphas less than 1 also helps assign more mass to the corners of the 3-simplex, meaning that we are more explicitly assigning mass to the first two components. We'll visualize this below using the starter code from the `Dirichlet` package by `dkahle` on Github (link) using $\boldsymbol{\alpha} = [0.8, 0.15, 0.05]$:



As clearly shown by the plot above, most of the prior mass goes to the lower left corner, with less mass in the lower right corner, and the least amount of mass at the top vertex. This behaviour further justifies our prior because it captures much of our intuition for raptor energy states; namely, it captures the intuition that the low energy state is more probable, the higher energy state is less probable, and the extremely high energy state is least probable. Additionally, there is still notable mass in the central areas of the 3-simplex, so it doesn't seem too "constrained".

So to recap, we will use a Dirichlet prior for the mixing proportions, and specifically, we will have the Dirichlet parameter $\boldsymbol{\alpha} = [0.8, 0.15, 0.05]$, i.e., the prior is $\boldsymbol{\pi} \sim \text{Dirichlet}([0.8, 0.15, 0.05])$.

Next we'll put some priors on these distributions. Note that we'll need 3. We will again start with the priors we used in task 1. Recall that they were: 1. $p \sim N(0.2, 0.05)$ 2. $\mu_1 \sim N(\log(0.01), 0.35)$ 3. $\sigma_1 \sim N_+(0.5, 0.125)$ 4. $\mu_2 \sim N(\log(1), 0.35)$ 5. $\sigma_2 \sim N_+(0.5, 0.125)$

In this part we'll need to instead put priors on:

1. $\boldsymbol{\pi}$
2. μ_1
3. σ_1
4. μ_2
5. σ_2
6. μ_3
7. σ_3

We already have a prior for $\boldsymbol{\pi}$, i.e., $\boldsymbol{\pi} \sim \text{Dirichlet}([0.8, 0.15, 0.05])$. As for the other priors, we will use the same approach we took with task 1. We do this since I only have one set of "beliefs" about raptor energy states, and so it follows that any priors I set would be set with the same set of "beliefs" I used to come up with the experiment parameters. As such, I will keep the centres of the priors to be those I used in simulating the data. We can keep the two priors for μ_1, σ_1 as we only split the high energy component for this task. As for the priors on the two higher energy states, we will use the same standard deviations for our priors of the mean of each component and the same prior for the standard deviation of each component, that is: $\tau_{\mu_2} = \tau_{\mu_3} = 0.35$ and $\sigma_2, \sigma_3 \sim N(0.5, 0.125)$. We do so for the same reason we did so in task 1: we should very rarely see values more than ≈ 4.3 times than the centre we set for that component. Also we believe that is reasonable here as it was in task 1 because it seems sensible raptors don't really spend more than 4.3 times the mean energy given a specific energy state. So then, we'll set the centres of the priors for the two higher

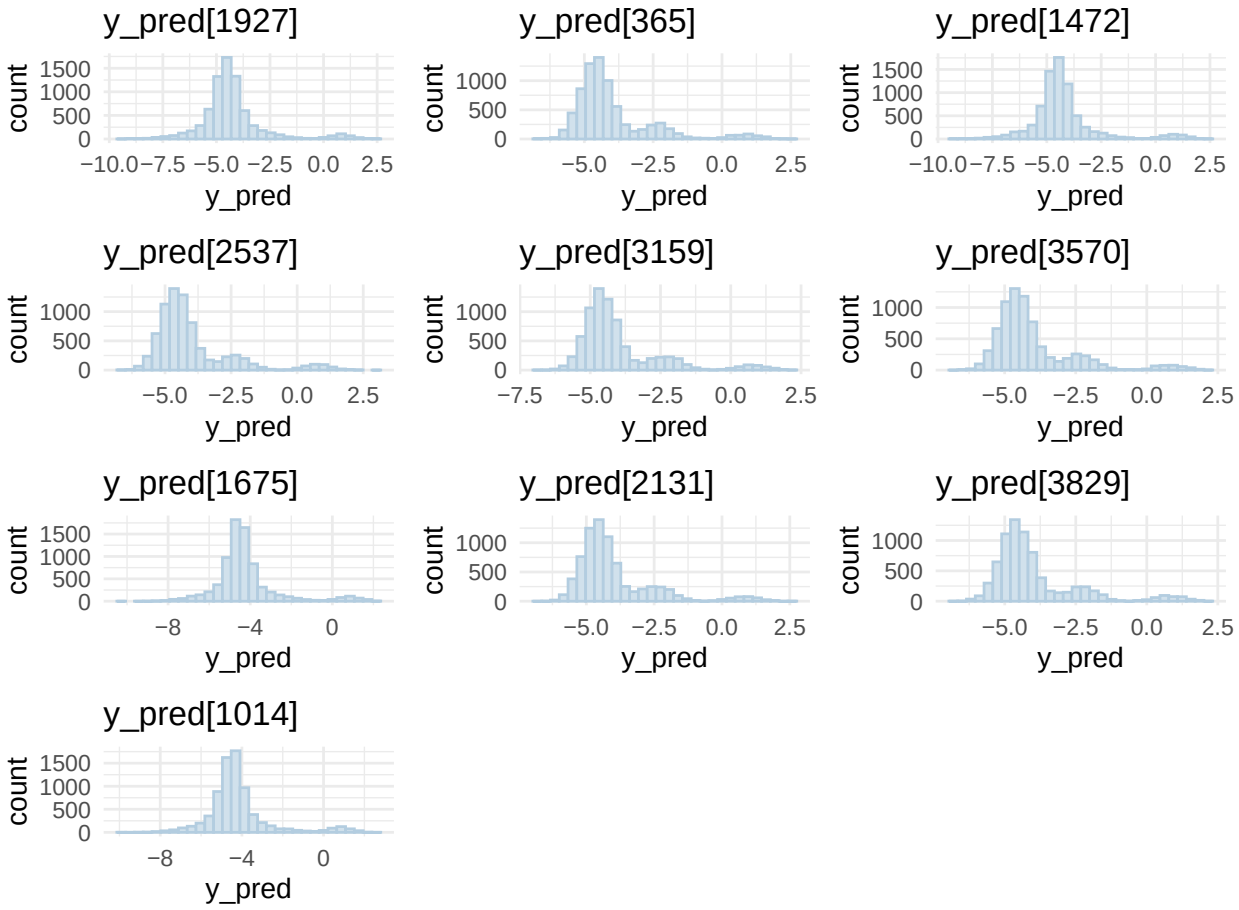
energy states. First, we set the centre of the μ_2 prior to be $\log(0.1)$. Second, we set the centre of the μ_3 prior to be 0.75. Thus we have our priors to be:

1. $\boldsymbol{\pi} \sim \text{Dirichlet}([0.8, 0.15, 0.05])$
2. $\mu_1 \sim N(\log(0.01), 0.35)$
3. $\sigma_1 \sim N(0.5, 0.125)$
4. $\mu_2 \sim N(\log(0.1), 0.35)$
5. $\sigma_2 \sim N(0.5, 0.125)$
6. $\mu_3 \sim N(0.75, 0.35)$
7. $\sigma_3 \sim N(0.5, 0.125)$

Evaluation of model on 3 component data

In this section, we will explore the fit for the 3 component model on 3 component data we simulated above at the start of this task. We will mainly evaluate if it was able to recover all 3 components, along with the usual posterior predictive checks to examine the model fit overall. Note that we will not evaluate prior checks as Prof. Simpson said they were unnecessary here in office hours.

We'll first examine the 10 randomly sampled posterior predictive distributions:

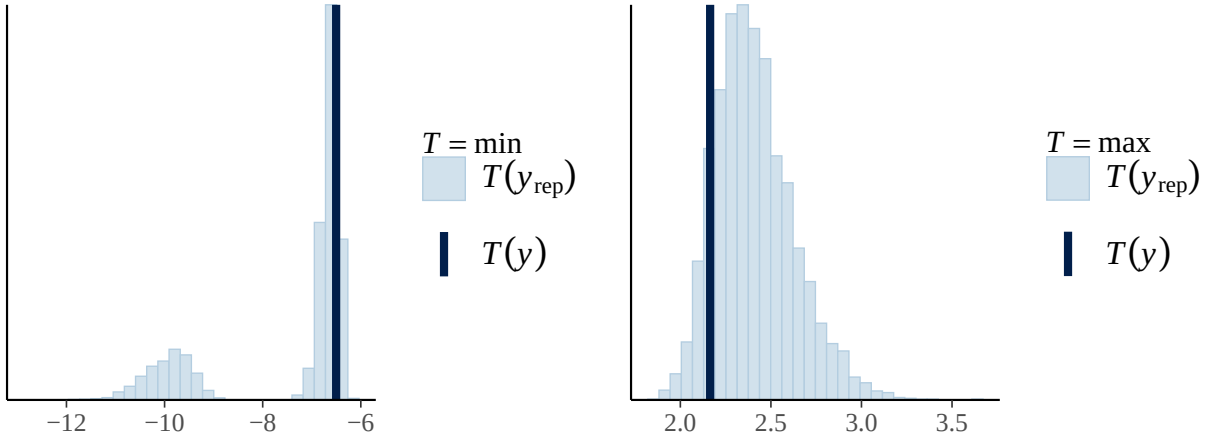


From the posterior predictive distribution, we see some interesting behaviour: some predictive distributions are essentially bimodal with one component that is shaped similarly to a Cauchy distribution and some distributions are trimodal with the densities of each component that we'd expect.

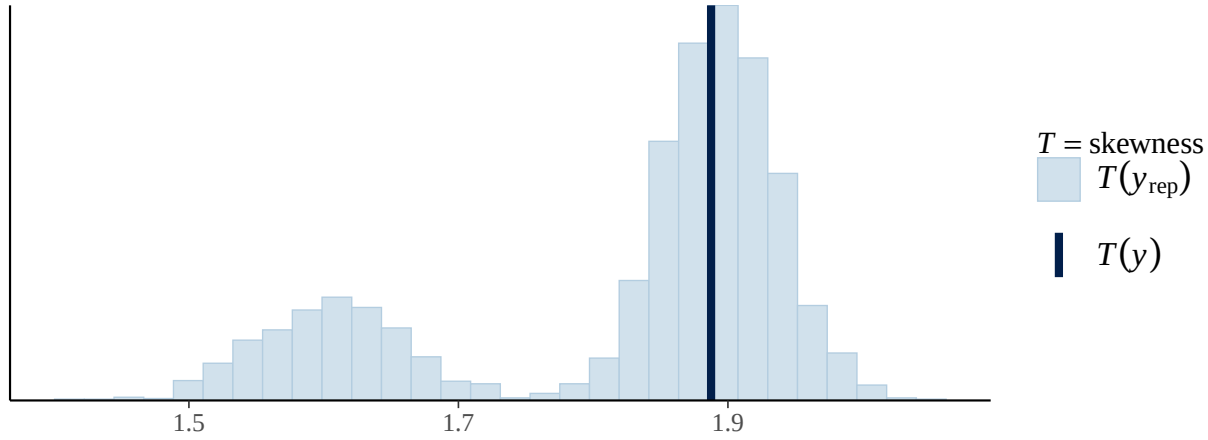
First, examining the clearly trimodal distributions, we see that the centres of each distribution are located

where we'd expect based on our simulation parameters. Clearly, in this case, the model was able to recover all 3 components, with roughly all 3 components having the "correct" amount of mass. In particular, the model was able to predict the moderate "dip" in mass between the low energy and higher energy components and the clear separation between the higher energy and extremely high energy components. This is promising evidence of a good fit.

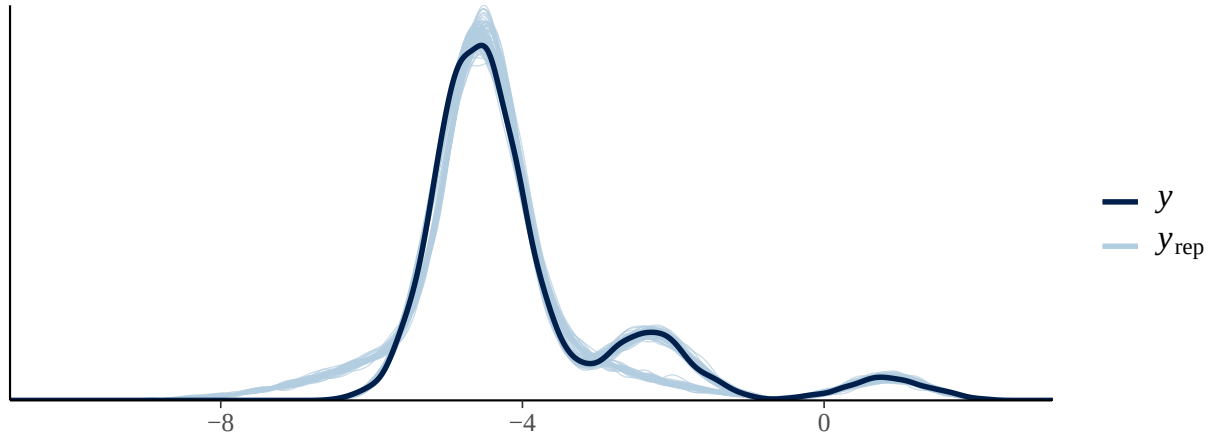
In the predictive distributions with two clear modes, we see extremely long tails in either direction with significant mass covering our simulated higher energy (second) component. Essentially the low energy and higher energy component "collapses" into each other in these cases, but the extremely high energy component seems to be predicted reasonably, i.e., closely resembles that of our simulated data. If we reflect back on the histogram of the simulated 3 component data, we see that this behaviour isn't wildly improbable because the low energy and higher energy component share regions of mass, namely in the lower tail of the higher energy component and upper tail of the low energy component. Additionally, there's no extremely clear "dip" between the low energy component and higher energy component, which we do see between the higher energy component and extremely high energy component. So then, such behaviour may be explained by Stan "choosing" a higher μ_1 and a lower μ_2 with relatively large σ_1, σ_2. This is possibly supported by the long tails indicated consistently by the tails "stretching" out to around -9. Despite the inability of the model in these cases to recover all three components, the predicted values are not wildly inaccurate but some values in the long tails are not seen in our simulated data, such as -9. We somewhat expect the range of our predictions to be wider than that of the simulated data though since we have weakly informative priors. Our concerns are mostly over the distribution of predictions, especially where the second component should have been. Still, the model looks promising in terms of providing a reasonable fit, but we will see this "idiosyncrasy" when we examine the ancillary test statistics next.



As noted in the previous paragraph, our bimodal predictive distributions play into these test statistics; namely, the `min` test statistic shows the effect of the model's inability to always recover all 3 components. In particular, the distribution around -10 in the $T(y_{rep})$ distribution is a reflection of these bimodal predictive distributions. If that distribution around -10 is ignored we see that the model was able to well capture the `min` test statistic. The story is even better when looking at the `max` test statistic plot in that the `max` test statistic was extremely well captured by the test statistic distribution as $T(y)$ is located exactly where most of the distribution's mass is. So clearly we have evidence that the model wasn't a great fit because of the occasional bimodal predictive distributions, but overall we don't have evidence that the model was really a misfit either.

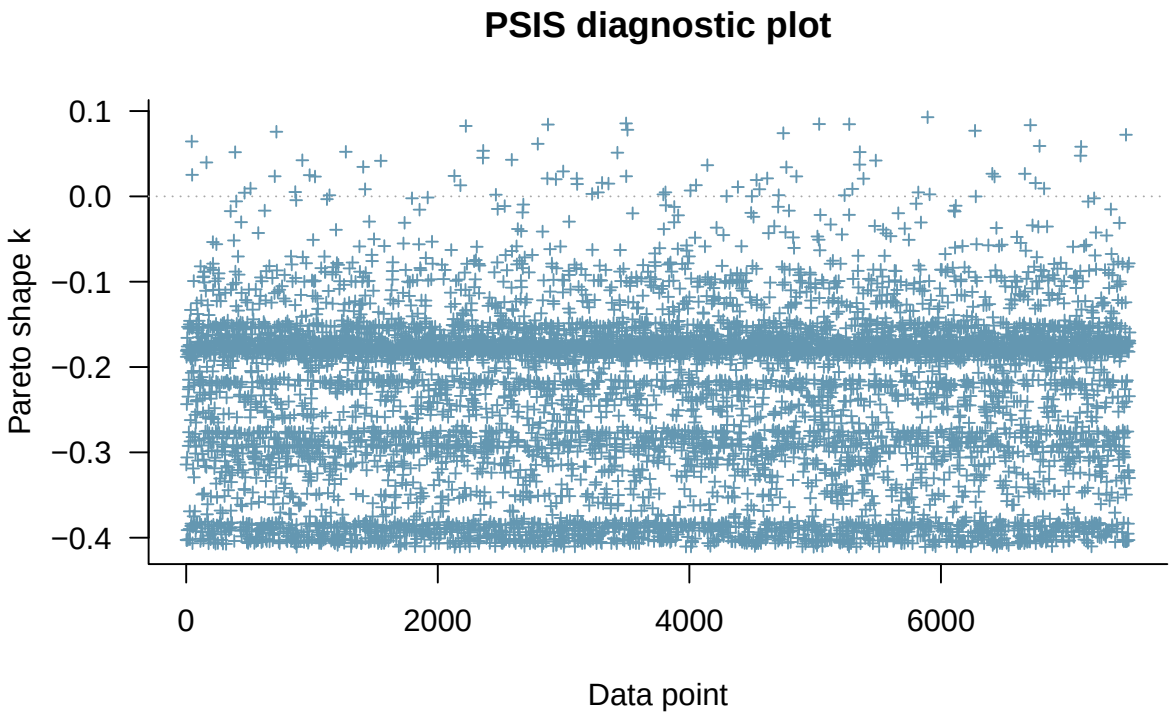


The story from the `min` test statistic plot carries over when we examine the `skewness` test statistic plot. There's a another smaller distribution centred around 1.6, while the vast majority is centred around ≈ 1.9. This is again likely due to the bimodal predictive distributions. However, like the `min` test statistic plot, the test statistic is located exactly in the centre of where most of the $T(y_{rep})$ mass is if we ignore the "weird" distribution around 1.6. As such, we continue to gather evidence that while this model wasn't a great fit, the model wasn't a horrible fit either.



From the density overlay plot we immediately see that the bimodal predictive distributions show up again. They are clearly visible from their wide tails. Building off of our observations in the previous paragraphs, the bimodal distributions assign not-insignificant amounts of mass to the wide tails, which extend outwards to about -8 on the low end and about -2 on the high end. Additionally, it's very clear that these bimodal distributions fail the capture the behaviour of the 2nd component, both it's dip around -4 and its peak around -3. Although interestingly, these bimodal distributions otherwise capture the behaviour of the data very well. We can clearly see that the bimodal distributions replicated the density of the values of the low energy component quite well, especially the peak around -5, only deviating with the long lower tails. Additionally, these bimodal distributions capture the behaviour of the extremely high energy component very well, notably being able to capture the dip around -1 and the peak around 1. We can also see that the trimodal predictive distributions replicate the behaviour of the data extremely faithfully, being able to capture all nuances of the data such as dips in density due to the separation of the components and all of the peaks of each component. So then, we again have evidence that the model fit the data fairly well, with the exception of the bimodal predictive distributions. Even then, after examining this plot, the bimodal predictive distribution may not be a truly severe issue as they don't show up very frequently and still captures the behaviour of the low energy and extremely high energy components well. We should be mainly concerned that in these bimodal

instances, the model overfit to the low and extremely high energy components and underfit the higher energy component. We will evaluate that concern using a PSIS plot for the model:



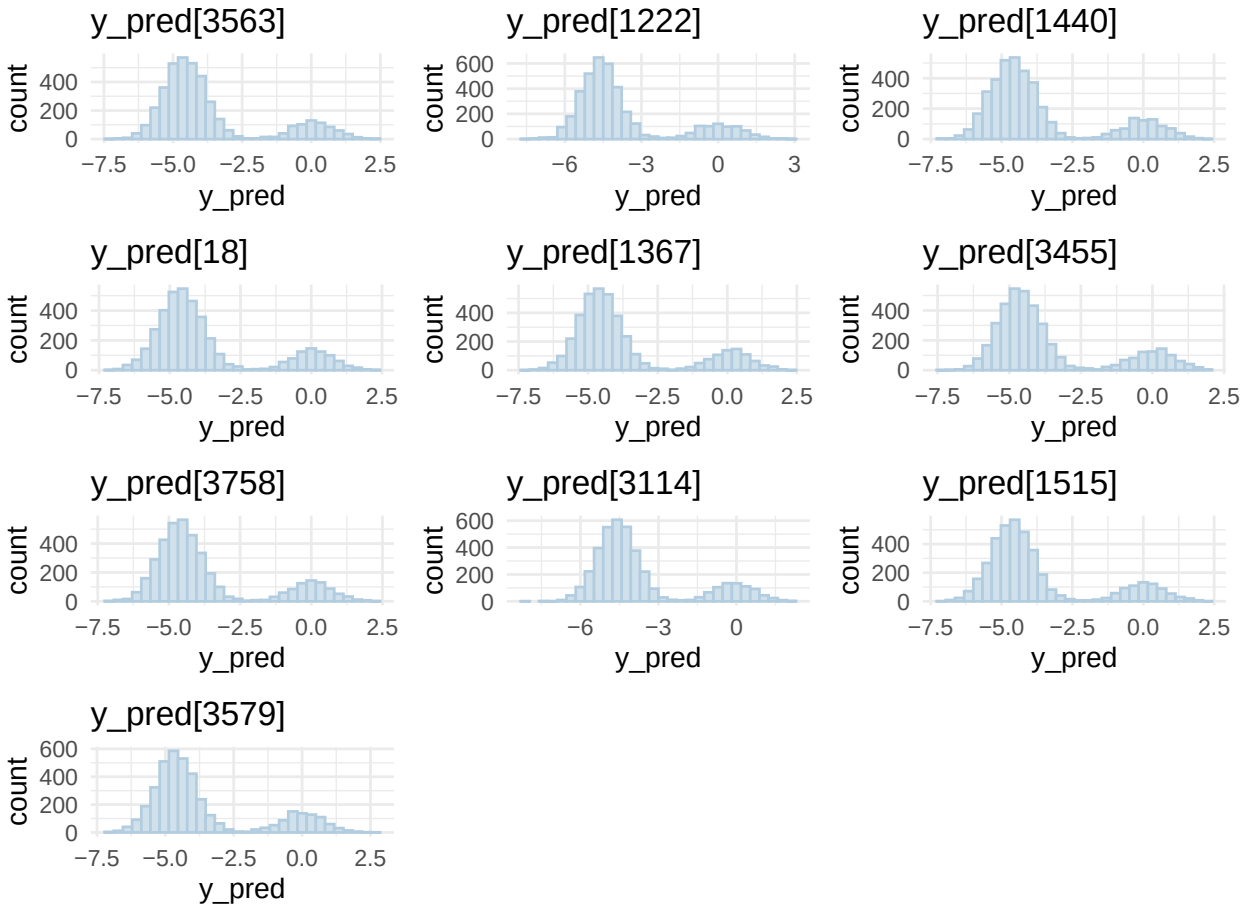
From the PSIS plot, we see that our concerns about the model potentially really underfitting the higher energy component were really supported. All of the $\hat{k}$ values for the model are well under 0.5, indicating that the model performed well in LOO-CV. Thus, even though the model sometimes had bimodal predictive distributions that failed to capture the behaviour of the higher energy component, it was still able to decently predict the data points, and none of the data points were influential as a result. Of course, we say this with the usual caveat that "good" $\hat{k}$ values are not necessarily indicative of actual predictions or how "close" predictions are to observed data.

So we conclude that our model was overall a decent fit with capable predictive ability, despite having the problem of occasionally generating predictive distributions that were bimodal and ignorant of the higher energy component.

Evaluation of model on 2 component data

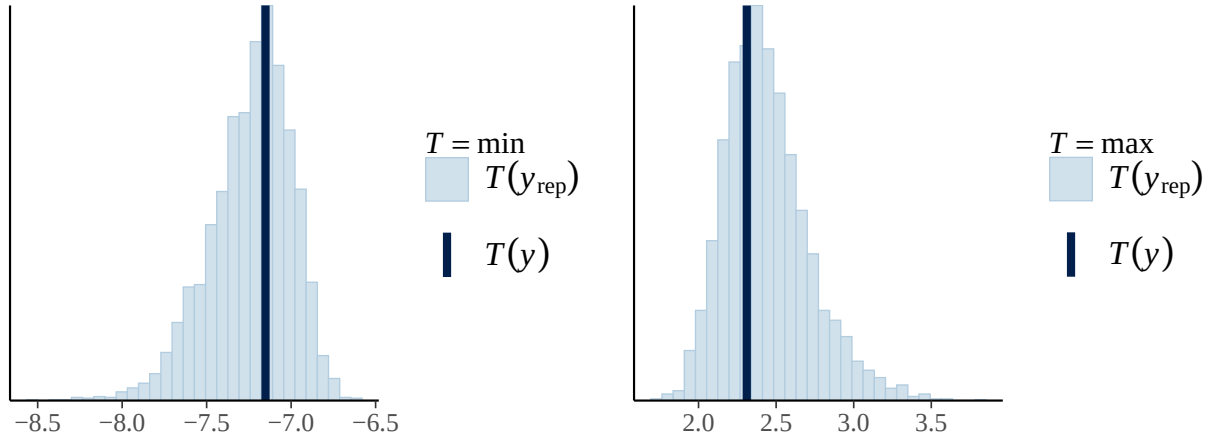
In this section, we'll evaluate the 3 component model's fit on 2 component data. We will use the same experimental set up as task 1, i.e., the same parameters for the distributions of the components, but we will use a much smaller n. We will choose $n = 4000$ since that should still give us about 800 samples for the high energy component (recall that p is 0.2); we should still have good convergence to a normal distribution as such. We choose to use a smaller n for computational reasons; using $n = 10000$ took far too long to fit, so much so that I had stop Stan before the model was fit. Note that we will not evaluate prior checks as Prof. Simpson said they were unnecessary here in office hours.

We'll start off by examining 10 randomly sampled posterior predictive distributions:



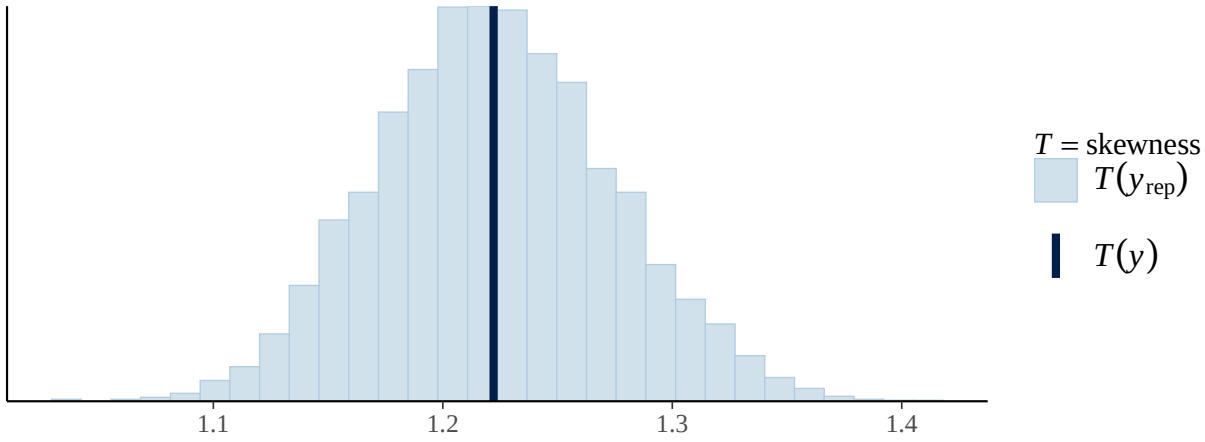
From the histograms of the posterior predictive distributions, we see that the distributions are overwhelmingly nearly identical in terms of tail length and behaviour. They almost all have a tail that reaches out to around 7.5, two peaks that are where we'd expect them $(\log(0.01), \log(1))$, and most notably, they're all bimodal. This is extremely promising behaviour of a very good fit in that the spurious 3^{rd} component was indeed ignored and the true two components were recovered faithfully with respect to the true distribution.

We'll examine the ancillary test statistics next:

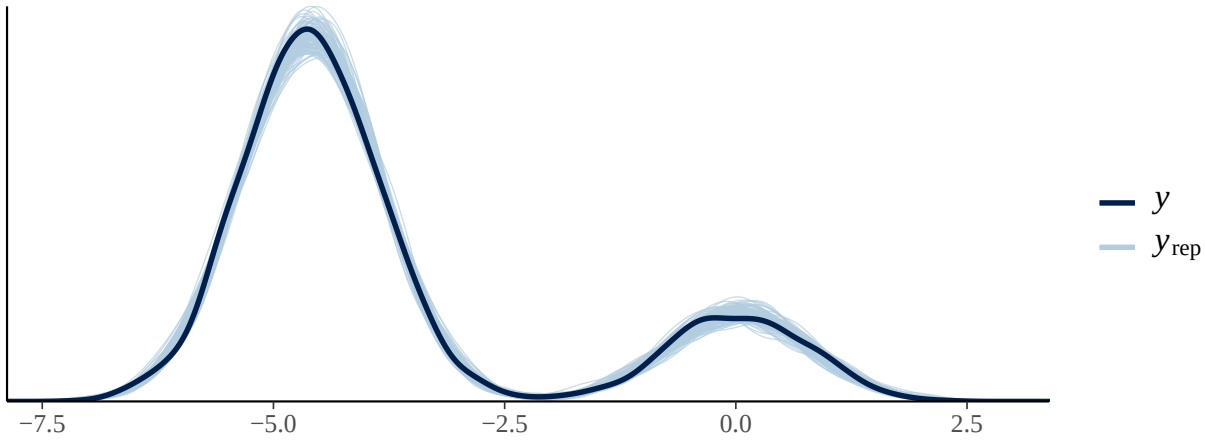


Immediately, we notice that the model was able to capture both `min` and `max` test statistics very well, i.e.,

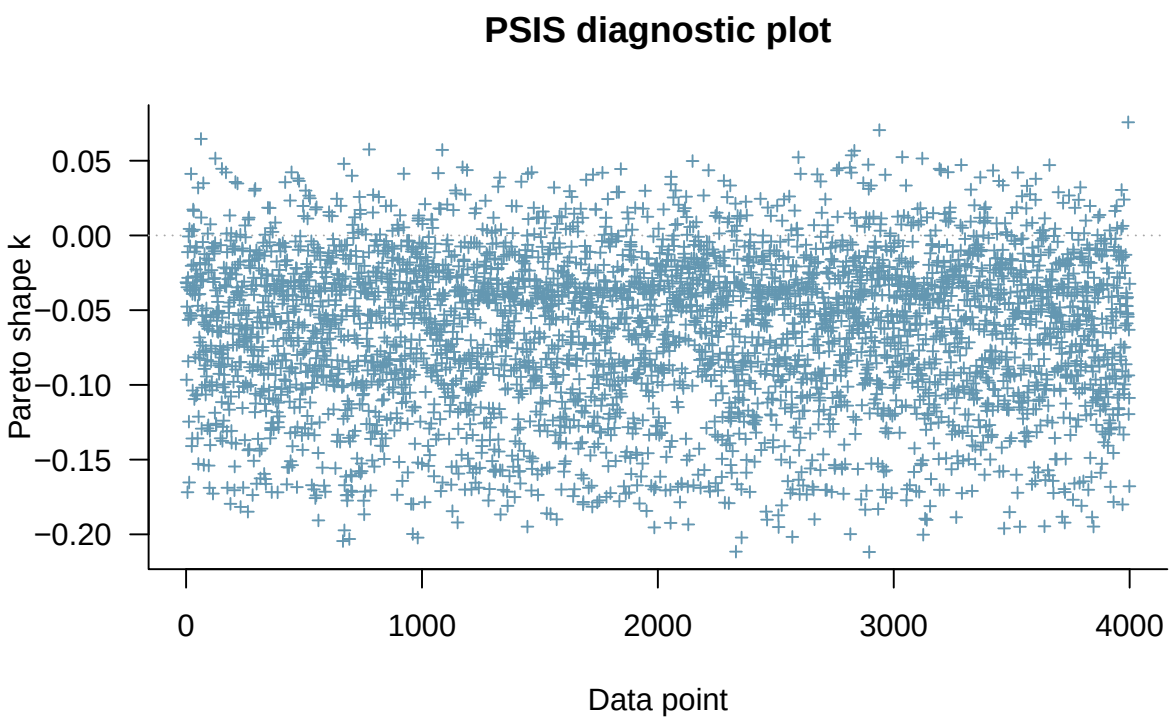
the respective test statistic was centred safely where the respective $T(y_{rep})$ distribution had the majority of its mass. Interestingly, the test statistics were comparatively even "better" centred than their counterparts from task 1, especially since they were fit on the same data.



The model was also completely able to capture the `skewness` test statistic because the test statistic is located in the centre of the $T(y_{rep})$ distribution. Again the 3 component model performed very similarly to the two component model in task 1. Overall the model was able to capture all three test statistics and is definitely indicative of a good fit. Next we'll examine the density overlay plot.



From the density overlay plot, we see that the model was able to capture the behaviour of the data extremely well. The density estimate curves of the posterior predictive distributions all followed the density estimate curve of the simulated data, and in particular, the model replicated the dip in density around -2 very well and the densities at the peaks (≈ -4.6 and 0) were replicated very well tooo. We can also clearly see again that the model did ignore the spurious component, as there are only two peaks, so we have more evidence of a well fit model.



From the PSIS plot we see more evidence that the model fit well on the data since all of the $\hat{k}$ values were well below 0.5. So we found no evidence of any simulated data points being influential points due to the model fit, and so we do have evidence that this model also has capable predictive ability as it is clearly not concerning overfit or underfit to portions of our data. But our $\hat{k}$ values aren't necessarily indicative of the exact predictions or how "close" the predictions are to the observed data.

Finally, for an interesting comparison, we fit the task 1 model to data we used here. Since we seemed to have a really promising fit on two component data using a three component model, did the two component model from task 1 out perform our three component model?

Table 3: Three component model versus two component model

	elpd_diff	se_diff	elpd_loo	se_elpd_loo	p_loo	se_p_loo	looic	se_looic
model1	0.0000000	0.0000000	-6485.126	56.81558	5.027647	0.1659321	12970.25	113.6312
model2	-0.1741097	0.4124792	-6485.300	56.87016	5.490584	0.2076069	12970.60	113.7403

So then while the task 1 model appears to be better, there's no super significant difference in performance since the ELPD difference is only ≈ -0.2 and the standard error difference is only ≈ 0.4! Not only did the three component model ignore the spurious component correctly, but it also performs essentially on par to the task 1 model, which is designed to fit this data; this model clearly fit well even on two component data. We also believe this is a fair comparison as the task 1 (two component) model was designed for this data.

Taking all of our analysis into account, we conclude that our three component model fit the two component data very well.

Overall conclusion of model fit and computational discussion

Overall, we conclude our three component model fit the data well.

Notably, the model sometimes failed to capture the higher energy component when fitting on three component data but still quite faithfully managed to capture the other two components, but it was able to capture the data's behaviour extremely well when it generated a trimodal posterior predictive distribution. Further, from the ancillary test statistics and PSIS plot, we saw more evidence that the model was a decent fit; in particular, despite sometimes only capturing two of the components, we had no evidence that the model was actually overfit to those portions of data.

Additionally, the model performed surprisingly well on the two component data. All of our evaluations for the model fit showed that we had a good fit on the two component data. Especially of note is that the model was able to always ignore the spurious component, and by using LOO-CV, we found that it performed essentially as well as the two component model.

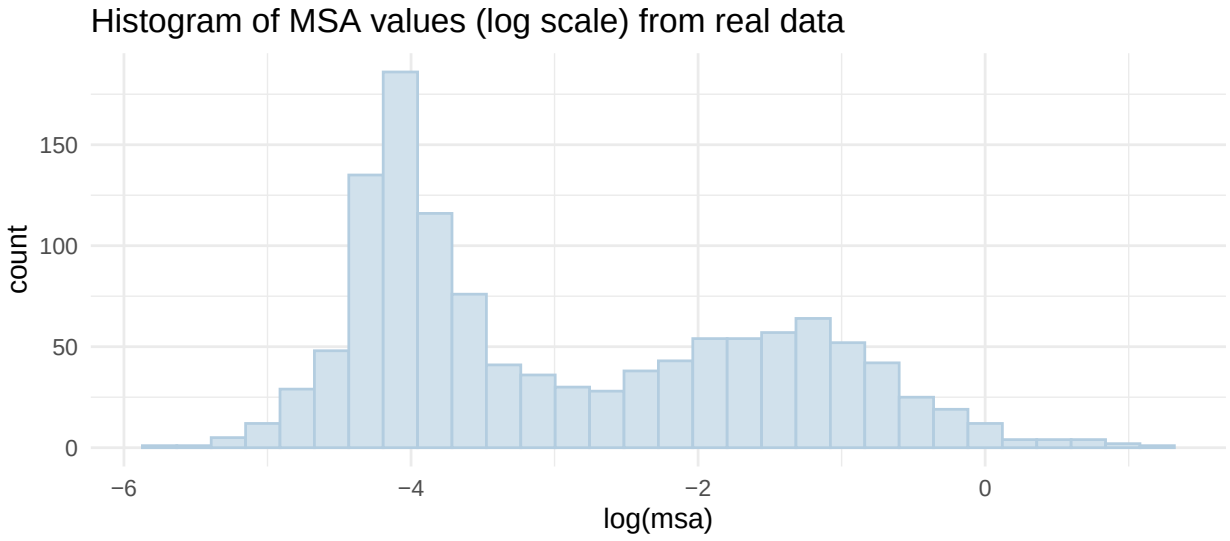
Like task 1, our models likely fit so well because we used one set of "beliefs" two both set the priors and create the experiment. Our situation here is a sort of "best case" in that our priors, despite being wide, are actually fairly informative of the simulated data, just not actual raptor data. In other words, giving the model access to the centres of the "true" distributions likely allowed it to fit so well. Had we needed to set priors based on background information and not been given the freedom to come up with the experiment as we've done with previous assignments/homeworks, we likely wouldn't have had such a good fit. Thus, suboptimal model behaviour such as occasionally ignoring the higher energy component would've been likely exacerbated. To reiterate then, I decided to give the model access to the centres of the "true" distributions because I only have one set of "beliefs" about the problem, and we weren't given constraints about how to set priors as in previous graded work. Thus, any priors (and conversely experiments) I would've come up with would've been hopelessly biased regardless of any attempts I could've made to simulate the "ignorance" to the true distribution parameters we usually have when setting priors in these assignments. Although, by approaching this as an "experiment", we were able to see that at least in theory a higher dimensional model is able to correctly fit lower dimensional data.

It follows then that perhaps if we were given the simulated data and asked to put priors on them as we have done before, we would've likely seen more computational issues. In fitting this model, the only computational issues were only occasional proposal rejections in the warmup phase, which often occurs in fitting models in Stan as it tries to find the optimal MCMC algorithm parameters, but also fitting this three component model on two component data took an absurdly long time in comparison to fitting it on three component data or just having a two component model fit two component data. To that last point, it seems like having Stan ignore the spurious component provides good results but at a large computational cost, perhaps due to the added computational complexity from the variable types and prior choice (Dirichlet versus Bernoulli).

Having finished our discussion, I've also reached the end of my STA365 journey! It has been quite an insightful and interesting experience.

Appendix

Additional Plots



R Code

Helpers

Simulating a Gaussian mixture

Samples `n_samples` times, given a vector of probabilities for the multinomial distribution (used as a categorical distribution) and parameter for the Gaussian components (formatted `[[mu_1, sigma_1], ...]` or alternatively as an $n \times 2$ matrix, where n is the number of components in the mixture)

```
simulate_gussian_mixture <- function(n_samples, probabilities, gaussian_params){
  #browser()
  samples <- c()
  mixtures <- c()
  data_matrix <- matrix(nrow =n_samples,ncol=2)
  colnames(data_matrix) <- c("logmsa", "distribution")
  for(i in 1:n_samples){
    distn <- rmultinomial(n = 1, probabilities)
    for(j in 1:length(distn)){
      if(distn[j] == 1){
        distn <- j
        break
      }
    }
    norm_params <- gaussian_params[distn, ]
    sample <- rnorm(1, norm_params[1], norm_params[2])
    samples <- c(samples, sample)
    mixtures <- c(mixtures, distn)
  }

  #browser()
  data_matrix[,1] <- samples
```

```
  data_matrix[,2] <- mixtures

  return(data.frame(data_matrix))
}
```

Posterior vs prior plots

```
plot_prior_posterior <- function(prior_fit, posterior_fit, variable_name="", x_lab = NA){

  if(is.na(x_lab)){
    x_lab = variable_name
  }

  posterior_plot <-melt(as_draws_matrix(subset_draws(posterior_fit,
                                                     regex =TRUE,
                                                     variable = variable_name))) %>%
    mutate(variable = str_replace_all(variable,
                                      pattern=paste(variable_name,".*", sep=""),
                                      replacement = "posterior"))
  prior_plot <- melt(as_draws_matrix(subset_draws(prior_fit, regex =TRUE,
                                                  variable = variable_name))) %>%
    mutate(variable = str_replace_all(variable,
                                      pattern=paste(variable_name,".*", sep=""),
                                      replacement = "prior"))

  comparison_df <- rbind(prior_plot, posterior_plot)

  comparison_plot <- ggplot(comparison_df,aes(x=value,
                                              fill = variable,
                                              color = variable)) +
    geom_histogram(alpha=1, stat = StatBin2) +
    scale_fill_manual(values=c(color_scheme_get()$dark,
                               color_scheme_get()$light)) +
    theme_minimal() +
    theme(legend.position="none") +
    labs(x=x_lab)+
    scale_color_manual(values =c(color_scheme_get()$dark_highlight,
                                 color_scheme_get()$light_highlight))

  return(comparison_plot)

}
```

Overlaid Histograms

```
plot_overlaid_hist <- function(df, variable_name=NA, data_name = NA, alt_var_name = "",
                               labels=c(), use_alt_var_name = FALSE, factor_convert = FALSE){

  #browser()
  plot_aes <- aes(x=data_name, fill=variable_name, color=variable_name)
  if (use_alt_var_name){
      variable_name <- ifelse(use_alt_var_name, alt_var_name, variable_name)
      plot_aes <- aes_string(x=data_name, fill=variable_name, color=variable_name)
  }
  if(factor_convert){
```

```
    variable_name <- as.factor(variable_name)
    plot_aes <- aes(x=data_name, fill=variable_name, color=variable_name)
  }

  fill_color_vec <- c()
  border_color_vec <- c()
  colors_length <- length(color_scheme_get())

  i <- 1
  while(i <= colors_length){
    temp_vec <- c(color_scheme_get()[[i]])
    #browser()
    if(i%%2==0){
      border_color_vec <- c(border_color_vec, temp_vec)
    }else{
      fill_color_vec <- c(fill_color_vec, temp_vec)
    }

    i <- i+1
  }

  plot <- ggplot(df,plot_aes) +
    geom_histogram(alpha=1, stat = StatBin2) +
    scale_fill_manual(values=fill_color_vec) +
    theme_minimal() +
    theme(legend.position="none") +
    labs(x=labels["x_lab"])+
    scale_color_manual(values =border_color_vec)


  return(plot)

}
```

Extract desired columns from Stan draws dataframe

```
subset_draws_df <- function(draws_df, column_names = c()){
  return(draws_df%>%select(starts_with(column_names)))
}
```

Plot PPD matrix

```
ppd_plot <- function(ppd_matrix, n_plots=10, file_name = "posterior_pre",
                     width = 3, height = 2, units = "in"){
    indicies <- sample(1:nrow(ppd_matrix), n_plots, replace = F)
    plot_list <- vector("list", n_plots)
    plot_list <- lapply(indicies, plot_helper, data = ppd_matrix)
    return(grid.arrange(grobs = plot_list))
}
```

Plot helper function

```
plot_helper <- function(i,data){
  title <- paste("y_pred[", i,"]",sep="")
  #print(title)
  y_pred <- data[i,]
```

```
  plot_tmp <- ggplot()+ aes_string("y_pred") +
        geom_histogram(alpha=1,
                       fill=color_scheme_get()$light,
                       color=color_scheme_get()$light_highlight,
                       stat = StatBin2)+
        labs(title=title)+
        theme_minimal()
  #print(plot_tmp)
  return(plot_tmp)
}
```

Statbin2 from StackOverflow to remove baseline colour: link

```
StatBin2 <- ggproto(
  "StatBin2",
  StatBin,
  compute_group = function (data, scales, binwidth = NULL, bins = NULL,
                            center = NULL, boundary = NULL,
                            closed = c("right", "left"), pad = FALSE,
                            breaks = NULL, origin = NULL, right = NULL,
                            drop = NULL, width = NULL) {
    if (!is.null(breaks)) {
      if (!scales$x$is_discrete()) {
        breaks <- scales$x$transform(breaks)
      }
      bins <- ggplot2:::bin_breaks(breaks, closed)
    }
    else if (!is.null(binwidth)) {
      if (is.function(binwidth)) {
        binwidth <- binwidth(data$x)
      }
      bins <- ggplot2:::bin_breaks_width(scales$x$dimension(), binwidth,
                                         center = center, boundary = boundary,
                                         closed = closed)
    }
    else {
      bins <- ggplot2:::bin_breaks_bins(scales$x$dimension(), bins,
                                        center = center, boundary = boundary,
                                        closed = closed)
    }
    res <- ggplot2:::bin_vector(data$x, bins, weight = data$weight, pad = pad)

    # drop 0-count bins completely before returning the dataframe
    res <- res[res$count > 0, ]

    res
  })
```

Markdown R Code

```
library(cmdstanr)
library(loo)
library(tidyverse)
library(posterior)
```

```
library(bayesplot)
library(latex2exp)
library(reshape2)
library(gridExtra)
library(PerformanceAnalytics)
library(knitr)
library(R.utils)
library(genpwr)
library(rlist)
library(DescTools)
library(multinomRob)
library(knitr)
library(dirichlet)
library(kableExtra)
# devtools::install_github("dkahle/dirichlet")

register_knitr_engine(override = TRUE)
set.seed(365)

# simulate data for task 1
gaussian_params <- matrix(NA, nrow = 2, ncol=2)
gaussian_params[1,] <- c(log(0.01),0.75)
gaussian_params[2,] <- c(log(1), 0.75)
simulated_data_t1 <- simulate_gussian_mixture(10000,
                                              c(0.8, 0.2), gaussian_params)

# plot simulated data
plot_overlaid_hist(simulated_data_t1,
                   variable_name = simulated_data_t1$distribution,
                   data_name = simulated_data_t1$logmsa,
                   labels = c(x_lab ="log(MSA)"),
                   factor_convert = TRUE)


task1_model <- cmdstan_model("task1_model.stan", compile = TRUE)

data_list_task1 <- list(y=simulated_data_t1$logmsa,
                        N = length(simulated_data_t1$logmsa),
                        mu_mlo = log(0.01), tau_mlo=0.35,
                        mu_mhi = log(1), tau_mhi = 0.35,
                        mu_sigmalo = 0.5, tau_sigmalo = 0.125,
                        mu_sigmahi = 0.5, tau_sigmahi=0.125,
                        mu_p = 0.2, tau_p = 0.05, only_prior = 1)

data_list_task1$only_prior <- 1
task1_prior <- task1_model$sample(data_list_task1,
                                  seed = 365,
                                  refresh = 500,
                                  parallel_chains = 4)
prior_t1 <- task1_prior$draws()

data_list_task1$only_prior <- 0
task1_posterior <- task1_model$sample(data_list_task1,
```

```
                                    seed = 365,
                                    refresh = 500,
                                    parallel_chains = 4)
posterior_t1 <- task1_posterior$draws()

t1_loo <- task1_posterior$loo(save_psis=T)
prior_t1_matrix <- as_draws_matrix(prior_t1)
posterior_t1_matrix <- as_draws_matrix(posterior_t1)
prior_t1_ppd<- subset_draws(prior_t1_matrix,
                            variable = c("y_pred"), regex=T)
posterior_t1_ppd<- subset_draws(posterior_t1_matrix,
                                variable = c("y_pred"), regex=T)


prior_pd_plots <- ppd_plot(prior_t1_ppd)

muh_comparison <- plot_prior_posterior(prior_t1_matrix,
                                       posterior_t1_matrix,
                                     "mu_hi", TeX(r'($\mu_2$)'))
sigmah_comparison <- plot_prior_posterior(prior_t1_matrix,
                                        posterior_t1_matrix,
                                       "sigma_hi", TeX(r'($\mu_2$)'))
mul_comparison <- plot_prior_posterior(prior_t1_matrix,
                                         posterior_t1_matrix,
                                     "mu_lo", TeX(r'($\mu_1$)'))
sigmal_comparison <- plot_prior_posterior(prior_t1_matrix,
                                        posterior_t1_matrix,
                                        "sigma_lo", TeX(r'($\mu_1$)'))
lambda_comparison <- plot_prior_posterior(prior_t1_matrix,
                                    posterior_t1_matrix,
                                    variable_name = "lambda", "p")

grid.arrange(muh_comparison, sigmah_comparison)

grid.arrange(mul_comparison, sigmal_comparison)

lambda_comparison

posterior_pd_plots <- ppd_plot(posterior_t1_ppd)

min_post <- ppc_stat(y = data_list_task1$y,
                     yrep=posterior_t1_ppd, stat = "min")
max_post <- ppc_stat(y = data_list_task1$y,
                     yrep=posterior_t1_ppd, stat = "max")
skewness_post <- ppc_stat(y = data_list_task1$y,
                          yrep= posterior_t1_ppd, stat = "skewness")
grid.arrange(min_post, max_post, ncol = 2)

skewness_post

sample_iters <- sample(1:nrow(posterior_t1_ppd), 100)
posterior_t1_sample <- posterior_t1_ppd[sample_iters,]
ppc_dens_overlay(y = data_list_task1$y,
```

```
                yrep = posterior_t1_sample)

plot(t1_loo)

data <- read.csv("Verreauxs.accel.txt", sep="\t")
kable(head(data), caption = "data used to fit models", booktabs = T)%>%
  kable_styling(latex_options = "HOLD_position")


task2_model <- cmdstan_model("task2_model.stan", compile = TRUE)

prior_matrix <- matrix(NA, nrow=7, ncol = 2)
# task 2 model, prior parameter matrix
prior_matrix[1,] <- c(log(0.01), 0.35) # mu_lo
prior_matrix[2,] <- c(log(1), 0.35) # mu_hi
prior_matrix[3,] <- c(0.5, 0.125) # sigma_lo
prior_matrix[4,] <- c(0.5, 0.125) # sigma_hi
prior_matrix[5,] <- c(-0.619039, 0.789093) # beta_0
prior_matrix[6,] <- c(0, 0.5) # beta_1
prior_matrix[7,] <- c(0, 0.5) # beta_2

data_list_task2 <- list(y=log(data$msa),
                        N = length(data$msa),
                        wind = data$wind_speed,
                        temp = data$saws_temp,
                        prior_params = prior_matrix,
                        only_prior = 1)

data_list_task2$only_prior <- 1
task2_prior <- task2_model$sample(data_list_task2,
                                  seed = 365,
                                  refresh = 500,
                                  parallel_chains = 4)
prior_t2 <- task2_prior$draws()

data_list_task2$only_prior <- 0
task2_posterior <- task2_model$sample(data_list_task2,
                                      seed = 365,
                                      refresh = 500,
                                      parallel_chains = 4)
posterior_t2 <- task2_posterior$draws()

prior_t2_matrix <- as_draws_matrix(prior_t2)
posterior_t2_matrix <- as_draws_matrix(posterior_t2)
prior_t2_ppd <- subset_draws(prior_t2_matrix,
                             variable = c("y_pred"), regex=T)
posterior_t2_ppd <- subset_draws(posterior_t2_matrix,
                                 variable = c("y_pred"), regex=T)

data_list_task2_m1 <- list(y=log(data$msa), N = length(data$msa),
                           mu_mlo = log(0.01), tau_mlo=0.35,
                           mu_mhi = log(1), tau_mhi = 0.35,
                           mu_sigmalo = 0.5, tau_sigmalo = 0.125,
```

```
                          mu_sigmahi = 0.5, tau_sigmahi=0.125,
                          mu_p = 0.2, tau_p = 0.05, only_prior = 1)

data_list_task2_m1$only_prior <- 1
task2_prior_m1 <- task1_model$sample(data_list_task2_m1,
                                     seed = 365,
                                     refresh = 500,
                                     parallel_chains = 4)
prior_t2_m1 <- task2_prior_m1$draws()

data_list_task2_m1$only_prior <- 0
task2_posterior_m1 <- task1_model$sample(data_list_task2_m1,
                                         seed = 365,
                                         refresh = 500,
                                         parallel_chains = 4)
posterior_t2_m1 <- task2_posterior_m1$draws()

model1_loo <- task2_posterior_m1$loo(save_psis = T)
model2_loo <- task2_posterior$loo(save_psis = T)
prior_t2_m1_matrix <- as_draws_matrix(prior_t2_m1)
posterior_t2_m1_matrix <- as_draws_matrix(posterior_t2_m1)
prior_t2_m1_ppd <- subset_draws(prior_t2_m1_matrix,
                                variable = c("y_pred"), regex=T)
posterior_t2_m1_ppd <- subset_draws(posterior_t2_m1_matrix,
                                    variable = c("y_pred"), regex=T)

prior_pd_plots_t2 <- ppd_plot(prior_t2_ppd)


muh_comparison_t2 <- plot_prior_posterior(prior_t2_matrix,
                                          posterior_t2_matrix,
                                        "mu_hi", TeX(r'($\mu_2$)'))
sigmah_comparison_t2 <- plot_prior_posterior(prior_t2_matrix,
                                           posterior_t2_matrix,
                                           "sigma_hi", TeX(r'($\sigma_2$)'))
mul_comparison_t2 <- plot_prior_posterior(prior_t2_matrix,
                                            posterior_t2_matrix,
                                        "mu_lo", TeX(r'($\mu_1$)'))
sigmal_comparison_t2 <- plot_prior_posterior(prior_t2_matrix,
                                       posterior_t2_matrix,
                                       "sigma_lo", TeX(r'($\sigma_1$)'))
lambda_comparison_t2 <- plot_prior_posterior(prior_t2_matrix,
                                     posterior_t2_matrix, "lambda", "p")
beta_0_comparison_t2 <- plot_prior_posterior(prior_t2_matrix,
                                     posterior_t2_matrix,
                                     "beta_0", TeX(r'($\beta_0$)'))
beta_1_comparison_t2 <- plot_prior_posterior(prior_t2_matrix,
                                     posterior_t2_matrix,
                                     "beta_1", TeX(r'($\beta_1$)'))
beta_2_comparison_t2 <- plot_prior_posterior(prior_t2_matrix,
                                     posterior_t2_matrix,
                                     "beta_2", TeX(r'($\beta_2$)'))
```

```
grid.arrange(muh_comparison_t2, sigmah_comparison_t2,
             mul_comparison_t2, sigmal_comparison_t2)

lambda_comparison_t2

grid.arrange(beta_0_comparison_t2,
             beta_1_comparison_t2,
             beta_2_comparison_t2)

prior_pd_plots_t2_m1 <- ppd_plot(prior_t2_m1_ppd)

muh_comparison_t2_m1 <- plot_prior_posterior(prior_t2_m1_matrix,
                                             posterior_t2_m1_matrix,
                                             "mu_hi", TeX(r'($\mu_2$)'))
sigmah_comparison_t2_m1 <- plot_prior_posterior(prior_t2_m1_matrix,
                                            posterior_t2_m1_matrix,
                                            "sigma_hi", TeX(r'($\sigma_2$)'))
mul_comparison_t2_m1 <- plot_prior_posterior(prior_t2_m1_matrix,
                                              posterior_t2_m1_matrix,
                                             "mu_lo", TeX(r'($\mu_1$)'))
sigmal_comparison_t2_m1 <- plot_prior_posterior(prior_t2_m1_matrix,
                                          posterior_t2_m1_matrix,
                                          "sigma_lo", TeX(r'($\sigma_1$)'))
lambda_comparison_t2_m1 <- plot_prior_posterior(prior_t2_m1_matrix,
                                       posterior_t2_m1_matrix,
                                       "lambda", "p")

grid.arrange(muh_comparison_t2_m1, sigmah_comparison_t2_m1,
             mul_comparison_t2_m1, sigmal_comparison_t2_m1)

lambda_comparison_t2_m1

posterior_pd_plots_t2 <- ppd_plot(posterior_t2_ppd)

min_t2 <- ppc_stat(y = data_list_task2$y,
                   yrep=posterior_t2_ppd, stat = "min")
max_t2 <- ppc_stat(y = data_list_task2$y,
                   yrep=posterior_t2_ppd, stat = "max")
skewness_t2 <- ppc_stat(y = data_list_task2$y,
                        yrep= posterior_t2_ppd, stat = "skewness")
grid.arrange(min_t2, max_t2, ncol = 2)

skewness_t2

sample_iters <- sample(1:nrow(posterior_t2_ppd), 100)
posterior_t2_sample <- posterior_t2_ppd[sample_iters,]
ppc_dens_overlay(y = data_list_task2$y,
                 yrep = posterior_t2_sample)

posterior_pd_plots_t2_m1 <- ppd_plot(posterior_t2_m1_ppd)


min_t2_m1 <- ppc_stat(y = data_list_task2$y,
```

```
                    yrep=posterior_t2_m1_ppd, stat = "min")
max_t2_m1 <- ppc_stat(y = data_list_task2$y,
                    yrep=posterior_t2_m1_ppd, stat = "max")
skewness_t2_m1 <- ppc_stat(y = data_list_task2$y,
                          yrep= posterior_t2_m1_ppd, stat = "skewness")
grid.arrange(min_t2_m1, max_t2_m1, ncol = 2)

plot(model1_loo)

plot(model2_loo)

# task 2 model comparison
knitr::kable(loo_compare(model1_loo, model2_loo),
             caption = "Task 1 model versus task 2 model", booktabs = T) %>%
  kable_styling(latex_options = "HOLD_position")

# task 3 experiment setup

gaussian_params <- matrix(NA, nrow = 3, ncol=2)
gaussian_params[1,] <- c(log(0.01),0.55)
gaussian_params[2,] <- c(log(0.1), 0.55)
gaussian_params[3,] <- c(0.75, 0.55)
simulated_data_t3 <- simulate_gussian_mixture(7500, c(0.8, 0.15, 0.05),
                                              gaussian_params)

plot_overlaid_hist(simulated_data_t3,
                   variable_name = simulated_data_t3$distribution,
                   data_name = simulated_data_t3$logmsa,
                   labels = c(x_lab ="log(MSA)"), factor_convert = TRUE)

# starter code from repo readme
# https://github.com/dkahle/dirichlet

f <- function(v) ddirichlet(v, c(0.8, 0.15, 0.05))
mesh <- simplex_mesh(.0025) %>% as.data.frame %>% tbl_df
mesh$f <- mesh %>% apply(1, function(v) f(bary2simp(v)))

p <- ggplot(mesh, aes(x, y)) +
  geom_raster(aes(fill = f)) +
  coord_equal(xlim = c(0,1), ylim = c(0, .85))
points <- rdirichlet(1e3, c(0.8, 0.15, 0.05)) %>% simp2bary %>%
  as.data.frame %>% tbl_df %>% rename(x = V1, y = V2)
p + geom_point(data = points, color = "orange", alpha = .4) +
  theme_minimal() + theme(legend.position="none") +
  labs(title = TeX(r'($\alpha = \[0.8, 0.15, 0.05\]$)'))

task3_model <- cmdstan_model("task3_model.stan", compile = TRUE)

prior_matrix <- matrix(NA, nrow=6, ncol = 2)

prior_matrix[1,] <- c(log(0.01), 0.35) # low energy, mu_1
prior_matrix[2,] <- c(log(0.1), 0.35) # higher energy, mu_2
prior_matrix[3,] <- c(0.75, 0.35) # extreme high energy, mu_3
```

```
prior_matrix[4,] <- c(0.5, 0.125) # sigma_lo, sigma_1
prior_matrix[5,] <- c(0.5, 0.125) # sigma_higher, sigma_2
prior_matrix[6,] <- c(0.5, 0.125) # sigma_super_hi, sigma_3
data_list_task3 <- list(y=simulated_data_t3$logmsa,
                        N = length(simulated_data_t3$logmsa),
                        categories = 3, prior_params=prior_matrix,
                        alpha = c(0.8, 0.15, 0.05), only_prior = 1)

data_list_task3$only_prior <- 0
task3_posterior <- task3_model$sample(data_list_task3, seed = 365,
                                      refresh = 500,parallel_chains = 4)
posterior_t3 <- task3_posterior$draws()

t3_3c_loo <- task3_posterior$loo(save_psis=T)
posterior_t3_matrix <- as_draws_matrix(posterior_t3)
posterior_t3_ppd<- subset_draws(posterior_t3_matrix,
                                variable = c("y_pred"), regex=T)

posterior_pd_plots_t3 <- ppd_plot(posterior_t3_ppd)

min_t3_3c <- ppc_stat(y = data_list_task3$y,
                      yrep=posterior_t3_ppd, stat = "min")
max_t3_3c <- ppc_stat(y = data_list_task3$y,
                      yrep=posterior_t3_ppd, stat = "max")
skewness_t3_3c <- ppc_stat(y = data_list_task3$y,
                           yrep= posterior_t3_ppd, stat = "skewness")
grid.arrange(min_t3_3c, max_t3_3c, ncol = 2)

skewness_t3_2c

sample_iters <- sample(1:nrow(posterior_t3_ppd), 100)
posterior_t3_3c_sample <- posterior_t3_ppd[sample_iters,]
ppc_dens_overlay(y = data_list_task3$y,
                 yrep = posterior_t3_3c_sample)

plot(t3_3c_loo)

gaussian_params <- matrix(NA, nrow = 2, ncol=2)
gaussian_params[1,] <- c(log(0.01),0.75)
gaussian_params[2,] <- c(log(1), 0.75)
simulated_data_t3_2c <- simulate_gussian_mixture(4000, c(0.8, 0.2),
                                                 gaussian_params)
data_list_task3_2c <- list(y=simulated_data_t3_2c$logmsa,
                           N = length(simulated_data_t3_2c$logmsa),
                           categories = 3,
                           prior_params=prior_matrix,
                           alpha = c(0.8, 0.15, 0.05),
                           only_prior = 1)

data_list_task3_2c$only_prior <- 0
task3_posterior_2c <- task3_model$sample(data_list_task3_2c,
                                         seed = 365,refresh = 500,
                                         parallel_chains = 4)
```

```
posterior_t3_2c <- task3_posterior_2c$draws()

t3_2c_loo <- task3_posterior_2c$loo(save_psis=T)
posterior_t3_2c_matrix <- as_draws_matrix(posterior_t3_2c)
posterior_t3_2c_ppd<- subset_draws(posterior_t3_2c_matrix,
                                   variable = c("y_pred"), regex=T)

posterior_pd_plots_t3_2c <- ppd_plot(posterior_t3_2c_ppd)

min_t3_2c <- ppc_stat(y = data_list_task3_2c$y,
                      yrep=posterior_t3_2c_ppd, stat = "min")
max_t3_2c <- ppc_stat(y = data_list_task3_2c$y,
                      yrep=posterior_t3_2c_ppd, stat = "max")
skewness_t3_2c <- ppc_stat(y = data_list_task3_2c$y,
                           yrep= posterior_t3_2c_ppd, stat = "skewness")
grid.arrange(min_t3_2c, max_t3_2c, ncol = 2)

skewness_t3_2c

sample_iters <- sample(1:nrow(posterior_t3_2c_ppd), 100)
posterior_t3_2c_sample <- posterior_t3_2c_ppd[sample_iters,]
ppc_dens_overlay(y = data_list_task3_2c$y,
                 yrep = posterior_t3_2c_sample)

plot(t3_2c_loo)

# fit task 1 model with task 3 2 comp data
data_list_task3_m1 <- list(y=simulated_data_t3_2c$logmsa,
                           N = length(simulated_data_t3_2c$logmsa),
                           mu_mlo = log(0.01), tau_mlo=0.35,
                           mu_mhi = log(1), tau_mhi = 0.35,
                           mu_sigmalo = 0.5, tau_sigmalo = 0.125,
                           mu_sigmahi = 0.5, tau_sigmahi=0.125, mu_p = 0.2,
                           tau_p = 0.05, only_prior = 0)
task1_posterior_t3 <- task1_model$sample(data_list_task3_m1,
                                         seed = 365,refresh = 500,
                                         parallel_chains = 4)
t1_model_t3_loo <- task1_posterior_t3$loo(save_psis =T)

knitr::kable(loo_compare(t1_model_t3_loo, t3_2c_loo),
             caption = "Three component model versus two component model",
             booktabs = T) %>%
  kable_styling(latex_options = "HOLD_position")
```

Stan code

Task 1 model (fixed p)

```
data {
  int<lower=0> N;
  vector[N] y;
  real mu_mlo;
  real tau_mlo;
  real mu_sigmalo;
  real tau_sigmalo;
  real mu_mhi;
  real tau_mhi;
  real mu_sigmahi;
  real tau_sigmahi;
  real mu_p;
  real tau_p;
  int<lower=0, upper=1> only_prior;
}

parameters {
  real mu_lo;
  real<lower=0> sigma_lo;

  real<lower = mu_lo> mu_hi;
  real<lower=0> sigma_hi;

  real<lower = 0, upper = 1> lambda;
}

model {
  //priors
  mu_lo ~ normal(mu_mlo, tau_mlo);
  mu_hi ~ normal(mu_mhi, tau_mhi);
  sigma_lo ~ normal(mu_sigmalo, tau_sigmalo);
  sigma_hi ~ normal(mu_sigmahi, tau_sigmahi);

  // theta param is the prob of y==1, i.e., z==2 (high energy state)
  lambda ~ normal(mu_p, tau_p);

  // mixture model, likelihood: y is p(y|z) marginalized over z
  if(only_prior == 0){
    for (n in 1:N){
      target += log_mix(lambda,
                        normal_lpdf(y[n] | mu_hi, sigma_hi),
                        normal_lpdf(y[n] | mu_lo, sigma_lo));
    }
  }
}

generated quantities{
  vector[N] y_pred;
  vector[N] log_lik;
  for(n in 1:N){
    // first sample a distribution based on lambda
```

```
    real distn;
    distn = bernoulli_rng(lambda);
    // then sample a distribution based on lambda
    // 1 indicates high energy state, 0 otherwise
    y_pred[n] = distn==1 ? normal_rng(mu_hi, sigma_hi) : normal_rng(mu_lo, sigma_lo);
    log_lik[n] = log_mix(lambda,
                normal_lpdf(y[n] | mu_hi, sigma_hi),
                normal_lpdf(y[n] | mu_lo, sigma_lo));
  }

}
```

Task 2 model (logistic model for p)

```
data {
  int<lower=0> N;
  vector[N] y;
  vector[N] wind;
  vector[N] temp;
  matrix[7, 2] prior_params;
  int<lower = 0, upper = 1> only_prior;
}

parameters {
  real mu_lo;
  real<lower=0> sigma_lo;

  real<lower = mu_lo> mu_hi;
  real<lower=0> sigma_hi;

  real beta_0;
  real beta_1;
  real beta_2;
}

transformed parameters{
  vector[N] lambda;
  // invert the logit to get the probabilities
  lambda = inv_logit(beta_0 + beta_1*wind + beta_2*temp);
}


model {
  //priors
  mu_lo ~ normal(prior_params[1,1], prior_params[1,2]);
  mu_hi ~ normal(prior_params[2,1], prior_params[2,2]);
  sigma_lo ~ normal(prior_params[3,1], prior_params[3,2]);
  sigma_hi ~ normal(prior_params[4,1], prior_params[4,2]);
  beta_0 ~ normal(prior_params[5,1],prior_params[5,2]);
  beta_1 ~ normal(prior_params[6,1],prior_params[6,2]);
  beta_2 ~ normal(prior_params[7,1],prior_params[7,2]);

  // mixture model
  if(only_prior==0){
```

```
    for (n in 1:N){
      target += log_mix(lambda[n],
                    normal_lpdf(y[n] | mu_hi, sigma_hi),
                    normal_lpdf(y[n] | mu_lo, sigma_lo));
    }
  }
}

generated quantities{
  vector[N] y_pred;
  vector[N] log_lik;

  for(n in 1:N){
    // first sample a distribution based on lambda
    real distn;
    distn = bernoulli_rng(lambda[n]);
    // then using that distribution, sample a prediction
    // 1 indicates high energy state, 0 otherwise
    y_pred[n] = distn==1 ? normal_rng(mu_hi, sigma_hi) : normal_rng(mu_lo, sigma_lo);
    log_lik[n] = log_mix(lambda[n],
                    normal_lpdf(y[n] | mu_hi, sigma_hi),
                    normal_lpdf(y[n] | mu_lo, sigma_lo));
  }

}
```

Task 3 model (3 component model)

```
data {
  int<lower=0> N;
  int<lower=0> categories;
  vector[N] y;
  // dirichlet shape param
  vector[categories] alpha;
  // parameters for the component priors
  matrix[6, 2] prior_params;
  int<lower = 0, upper = 1> only_prior;
}

parameters {
  ordered[categories] mu;
  vector<lower = 0>[categories] sigma;
  // the mixing proportions
  simplex[categories] theta;
}

model {
  // from stan docs
  // cache log calculations, i.e., log prior
  vector[categories] log_theta = log(theta);
  // theta is the list of probabilites for each category
  theta ~ dirichlet(alpha);
  //print(theta);
  //priors
```

```
  mu[1] ~ normal(prior_params[1,1], prior_params[1,2]);
  mu[2] ~ normal(prior_params[2,1], prior_params[2,2]);
  mu[3] ~ normal(prior_params[3,1], prior_params[3,2]);
  sigma[1] ~ normal(prior_params[4,1], prior_params[4,2]);
  sigma[2] ~ normal(prior_params[5,1], prior_params[5,2]);
  sigma[3] ~ normal(prior_params[6,1], prior_params[6,2]);

  // mixture model
  if(only_prior==0){
    for (n in 1:N){
      // from stan docs: reassign lps to log_theta
      // stores log posterior for each class of observation n
      vector[categories] lps = log_theta;
      // the log posterior is log lik + log prior
      for(k in 1:categories){
        //print(k);
        lps[k] += normal_lpdf(y[n] | mu[k], sigma[k]);
      }
      target+= log_sum_exp(lps);
    }
  }

}

generated quantities{
  vector[N] y_pred;
  vector[N] log_lik;
  vector[categories] lps;
  for(n in 1:N){
    // first sample a distribution based on lambda
    int distn;
    // the categorical is also known as the multinoulli!
    // samples one category from a K-dim vector of probabilities
    distn = categorical_rng(theta);
    // then predict from the corresponding distribution
    // 1 indicates high energy state, 0 for low energy, 3 for 3rd energy state
    y_pred[n] = normal_rng(mu[distn], sigma[distn]);

    // calculate the log likelihood for each observation
    lps = log(theta);
    // the log posterior is log lik + log prior
    for(k in 1:categories){
      lps[k] += normal_lpdf(y[n] | mu[k], sigma[k]);
    }
    log_lik[n] = log_sum_exp(lps);
  }

}
```