## PROBLEM 3

In this context, it's noted that approximately 2/3 of all outcomes are rejected. However, as long as the distribution generating my guesses remains symmetric, convergence will occur. When running the chain in Problem 3 with independent outcomes from the posterior distribution, the goal is to simulate 5000 LCQs, indicating potential signs of good convergence. The recommendation is to ensure the last 5000 iterations exhibit convergence. There are a couple of options, such as taking the last 5000 from a pool of 15000 or running 5 chains of 5000 each with different initial values and selecting the last 1000 from each chain to confirm convergence. It's acceptable to run a single chain if preferred, as it doesn't affect the results for Liam, as long as the last 5000 iterations demonstrate convergence. Additionally, it's crucial to remember the initial guess when running 5 chains.

Regarding simulating all three parameters simultaneously, this entails observing a simultaneous posterior distribution by running a single chain and monitoring the convergence of all three parameters. After achieving convergence, 1000 outcomes are selected, resulting in a vector of 1000 values. Each outcome "i" corresponds to one point on the plot of "theta," "alpha," and "beta." It's acknowledged that there may be some unusual estimates in the scatterplot due to limited data for each skateboarder, preventing perfect convergence. Nevertheless, such deviations are expected to be acceptable.

Finally, for RUNS, the focus is on eliminating everything related to "theta." The same process is repeated with the exclusion of "theta," now incorporating Run ratings. The initial guess is based on the moment method for the population, taking into account the run ratings.

**PROBLEM 3**

***1. Data Assumption:*** As long as the distribution generating your guesses is symmetrical, it will converge.

***2. Simulation Process:***

- When running the chain in Problem3, assuming we have independent outcomes from the posterior distribution...
- You aim to simulate 5,000 LCQs.
- There are indications of good convergence.
- Ideally, you should:
  - Run the chain ensuring at least the last 5,000 look good.
  - Consider taking 15,000 and selecting the final 5,000.
  - Even better, run 5 chains of 5,000 length each with different initial values. From these, take the last 1,000 from each chain.
  - This process can verify good convergence.
  - It's fine to run a single chain for Liam's purpose, but ensure to take the last 5,000 from a chain where you've achieved convergence. Always check for convergence over the last 5,000 outcomes.
  - If running 5 chains, remember the initial guesses.

***3. Simulating Parameters:***

- You're simulating all 3 parameters at the same time.
- This is a simultaneous posterior distribution. After achieving convergence for all 3 parameters, take 1,000 outcomes, resulting in a vector for all 1,000.
- An outcome "i" for plots labelled "theta", "alpha", and "beta" provides a specific result.
- There's a question about whether you'll avoid strange estimations in your scatterplot. Some might appear odd due to limited data per skateboarder, affecting convergence perfection. However, the expectation is that it will be generally okay.

***4. Runs:***

- Everything related to "theta" has been excluded for "RUNS".
- The same process is run, but "theta" is removed.
- This is now based on Run-ratings. The initial guess is the moment method for a population based on run ratings.

## PROBLEM 3

In Problem 3, let's take a closer look at those plots that seem a bit sparse. It's natural to wonder why this is happening. Well, one possible explanation is that a large chunk of the population is getting rejected. This suggests that early on, the method stumbled upon a pretty good set of parameters, which, in turn, made it difficult for the chain to explore further. We call this issue a "mixing problem," where the chain isn't really delving into the posterior as much as we'd like, although it still manages to spread things out decently.

Now, why might this be occurring in our case? It could be linked to our jumping distribution for the theta parameter. The thing is, it doesn't take the previous value into account. Consequently, it occasionally suggests some rather wild options for theta, leading to frequent rejections. So, how do we tackle this? Well, we've got a couple of options. One is to run longer chains, giving the chain more time to "mix" and explore. Another option is to switch to a different jumping distribution for theta.

However, here's the catch: we need to make sure that the proposed theta values fall within the [0,1] interval. We could use some fancy mathematical tricks to handle this, but here's a quick and somewhat 'hacky' alternative. We can stick with the same normal jumping distribution for theta and then add a rejection loop. This loop ensures that we keep trying to jump until our proposed value lands within the desired interval [0,1]. It might not be the most elegant solution, but it gets the job done and helps us improve our parameter space exploration.

## METROPOLIS ALGORITHM

Let's talk about the Metropolis algorithm. The thing with the Metropolis algorithm is that it tends to take significantly more time to converge when dealing with a higher number of parameters. This happens especially in very high-dimensional spaces that we're exploring. In our Chapter 11 notes, what we mentioned is that we will generate samples from a certain point.

For instance, in the Rat example, we generated samples from a specific point based on the data we had. We also mentioned that we marginalized certain parameters, meaning we computed what the posterior distribution looks like. This is feasible because we selected conjugate distributions. However, if you opt not to choose conjugate distributions, you might encounter challenges in computing the posterior distribution and formulating a straightforward function. In such cases, you'd need to apply the Metropolis algorithm to all of the parameters. This is necessary to account for the interdependencies between them.

So, if you decide not to go the conjugate route, you'd have to simulate the process. You'd take the function you specified, multiply it by the data distribution you've chosen, and then apply the Metropolis algorithm to that combination. This approach ensures that you adequately address the complexities introduced by multiple parameters and their dependencies.

## PROBLEM 4

Let's delve into this scenario involving Santiago and different locations. The professor brings up an interesting point regarding the competitions. We have three events, each with a preliminary and a final round, totaling six competitions. So, it's essential to break them down since they happen on separate dates and provide higher resolution. We could consider them separately by location, but that would lead to questions like, "Did the skateboarder participate in this location?" This is where theta comes into play, as Santiago participated in Jacksonville, and we're modeling theta specifically.

Now, when it comes to the data from X, it tells us how often Santiago landed the trick. If we break it down by competition, the first thing we'll have is a Binomial distribution, denoted as Bin(4, theta_i), with the parameter theta_i. Essentially, we're using the rat model example for our data. We start with the Binomial distribution and then work backward using conjugate families—Binomial to Beta distribution. The leftmost hyperparameter represents the skill level across all competitions for one skateboarder, with outcomes considered independent based on this underlying skill level. However, the potential outcome can vary across different days and competitions. For instance, it might be super-hot in Jacksonville.

So, we've taken theta to be Beta-distributed, and for generating the score, we use the same prior with Alpha and Beta parameters. We can apply the same function and formula used in PROBLEM3, and the data will help shift the function to determine how well the skateboarder is doing because we're only looking at the posterior.

In PROBLEM4, the professor suggested using the frequentist point estimate for Alpha and Beta for the tricks, rather than computing a super complicated model. Essentially, we're applying Bayesian techniques, similar to the rat example in this situation. We'll use the Metropolis algorithm to generate another theta, representing the LCQ for all 5000 samples. This theta will be used to determine whether they land a trick or not, resulting in a sequence of 4 Bernoulli outcomes. We'll then use our frequentist estimate for Alpha and Beta for the trick score for each case. This encompasses what the problem is asking us to do.

In principle, there's room for exploring the possibility of having a prior for Alpha and Beta, but that would be a more complex undertaking, involving estimating a posterior distribution for all parameters together while traversing the entire graph from left to right.

In this model, the runs don't play a significant role. We could choose to use the frequentist approach or the Bayesian one, depending on our preference, as runs aren't a major concern in PROBLEM4. The professor's emphasis is on making reasonable choices and constructing a model, without a single correct solution in mind.

## PROBLEM 4

To summarize, in the data distribution, we organize the data based on the competition. This segmentation provides us with a substantial amount of data. Vahid introduced an interesting concept involving permutations. When dealing with a vector where the order doesn't matter because it originates from the same distribution, permuting the vector can help us generate more samples. This can be particularly useful when working with a limited number of data points.

Lima emphasized that we don't need to resort to bootstrapping. Instead, we can simply take four data points and consider them as Binomial(4) trials. Since we've separated the data by competition, we have multiple sets of these four data points, which we then transform into Beta distributions. The significant "Phi" parameter on the far left requires a function to be applied to it. Remember that 10.8 function? We apply a similar process as in PROBLEM 3, utilizing the Metropolis algorithm to generate samples from the posterior represented by that "Big Phi."

We follow a similar procedure to what we did in PROBLEM 3. Yes, this process primarily concerns the parameter theta. As for alpha and Beta, we can opt for either the frequentist or Bayesian models. In the case of RUNS, we follow a similar approach. After generating the LCQs, we proceed to compare the models in step 5.

In principle, our objective is to treat this situation as we did in Chapter 11, where we considered the Binomial distribution as conjugate to the Beta distribution. That's why we employ a conjugate prior. Handling more parameters can be challenging, which is why we're simplifying the approach for the sake of practicality.