**Computer Lab 7 – Phylogenetics III:**

**Bayesian Methods and MCMC**

**Conservation Genetics (BIOL 4174 / 5174)**

Part II – MrBayes

MrBayes is one of the most popular options for building phylogenetic trees in a Bayesian framework. We will install the version hosted by bioconda. Install by typing:

conda install -c bioconda mrbayes

To begin, download the lab 7 assignment file from Blackboard and unzip it to your personal folder on the desktop. Start by opening your helo-atp86.nex file in Text Wrangler. You should see a data block that contains information describing the size and contents of the data followed by matrix of sequences. The data block contains both genes ATP8 and ATP6 for each taxon. In our analysis, we are going to use MrBayes to simultaneously apply different models of nucleotide substitution to each of the two genes.

Before running MrBayes, we first have to provide it with commands. This is best accomplished by writing a MrBayes block in your Nexus file. This block follows the same rules as other blocks that you will find in Nexus files, however it has its own unique commands. At the end of the file, after **all** of the other text in the file (after the existing “end;”), start making a MrBayes block. Type all commands. **Do not copy / paste** your commands from this document because Word may have made slight formatting changes to the text that will prevent commands from running properly.

begin mrbayes;

end;

Next, add a command to the block (as shown below) that will set S\_punctatus as the outgroup. This means that the trees we produce will be automatically rooted by *S. punctatus*.

begin mrbayes;

outgroup S\_punctatus;

end;

Next the dataset will be partitioned into individual genes, which will allow MrBayes to optimize substitution rates (etc) separately for the two genes in our dataset. For this, we will use the charset command. ATP8 represents the first 168 bases of our alignment. From this point on, continue to insert all additional commands within your MrBayes block as shown below. **IMPORTANT:** **When finished entering all commands today, you should have only a single MrBayes block in your file that contains all of the commands from this handout.**

begin mrbayes;

outgroup S\_punctatus;

charset atp8 = 1-168;

end;

**Now, use the charset command to insert another line of code to designate the rest of the alignment (characters 169 to 852) as belonging to the ATP6 gene. Model this command after the above charset command.**

Because we are applying different mutation models to our two genes, we must use the partition command. You will use the **partition** command to create a partition named “genes,” tell the program that there will be two partitions, and then designate which charsets will be assigned to these partitions. From this point on, your partitions will be referred to by numbers. The order in which you input them using the partition command will determine their numbers. ATP8 will be (1) and ATP6 will be (2). The **set** command then makes the partitions active. Insert the following code into your MrBayes block in the lines after your charset commands.

partition genes = 2: atp8, atp6;

set partition = genes;

Now that this has been done, we can apply the two models to the data using the **lset** command. (read this as “L-set” not “one-set”). Within the **lset** command, we will use **Applyto** for the purpose of applying different models to different genes, **Nst** to designate the nucleotide substitution model, and Rates to apply the invariant sites and gamma rate parameters. For ATP8, your line of code should read as follows:

lset Applyto=(1) Nst=6 Rates=gamma ngammacat=4;

MrBayes uses numbers to designate the nucleotide substitution models. **Nst=1** represents the Jukes Cantor (JC) model, **Nst=2** designates the Hasegawa, Kishino, Yano (HKY) model, and **Nst=6** is used to specify the General Time Reversible (GTR) model. Under **Rates**, **gamma** is used to specify the “+G” parameter of a model, and **invgamma** would be used to designate “+I+G.” The **ngammacat** command is used to divide the gamma distribution into discrete rate categories because using the continuous gamma distribution to calculate likelihoods is computationally difficult. Increasing the number of categories improves the approximation of the gamma distribution, but increases calculation time. **Add a second lset line to apply GTR+I+G, with 4 rate categories to atp6, modelled after this last lset command**.

**Provide the line of code you will use to apply the GTR+I+G model to the ATP6 gene in the homework document.**

We also want to allow our two genes to evolve at different rates. For this we will use **ratepr** within the **prset** command. Insert the following line of code into your MrBayes block:

prset ratepr=variable;

We will also want to use **unlink** to issue a few more commands to the program. This allows parameters to be estimated at different values for each gene. For example, both of your models are using the gamma rate parameter. By default, the rate parameter for both genes is linked – i.e., both genes will sample from the same gamma distribution. By unlinking this parameter, the program is now free to sample from different gamma distributions for each gene. We will also unlink the substitution rates for the GTR model (**revmat**), character state frequencies (**statefreq**), and proportion of invariable sites (**pinvar**).

unlink shape=(all) pinvar=(all) statefreq=(all) revmat=(all);

Lastly, we will set some of the conditions of our Markov Chain before starting. Insert the following line of code into your MrBayes block:

mcmcp ngen=100000 printfreq=100 samplefreq=100 nruns=2 nchains=4 savebrlens=yes relburnin=no burnin=0 mcmcdiagn=yes diagnfreq=5000;

In this line of code, we are using the mcmcp command to designate a lot of conditions. There is a lot happening here, so we will go through them one by one.

* **ngen** is used to specify the number of generations in our MCMC. For now, we have set it to 100,000 to run a short, exploratory test. Normally, you would run it longer
* **printfreq** is used to tell the program how often it should refresh the screen while the MCMC is running. In this case, it will update after every 100 generations have been run. For very long analyses, it is a good idea to increase this number because each time this prints to your screen it slows down the analysis a little.
* **samplefreq** is used to tell how often a tree should be saved. In Bayesian analyses, you typically do not save the output of each generation of your MCMC because there may be autocorrelation among sequential generations, which can bias your results. By only sampling every 100th generation, we should be avoiding this problem. Later in the semester I will show you how to make a more informed choice for the sample frequency.
* **nruns** is used to designate the number of independent runs that take place simultaneously. It is a good idea in Bayesian analyses to do multiple independent runs with different starting values to determine whether or not the analysis is converging on different answers.
* **nchains** is used to designate how many chains will be used in each independent run of your analysis. The MCMC simultaneously runs multiple chains, and will jump from one to another while it is running. If one chain finds a better result than the others [ i.e., a lower –log(likelihood) ], the MCMC will begin sampling from that chain. Also, it will use an algorithm to decide when to periodically jump to a new chain with a slightly worse result. This is done to help prevent the analysis from becoming stuck in local minima.
* **savebrlens** is used to save branch lengths of your trees.
* **relburnin** is set to “yes” by default. When it is set to yes, burnin is automatically calculated as the first 25% of trees. The burnin is subsequently discarded when consensus trees and other summary statistics are calculated. We have set it to “no” for this lab so you can see how we can manually determine burnin.
* **burnin** is used to manually set burnin. Here, we have set it to 0 so that all trees will be saved in our initial exploratory analysis. When relburnin=yes, the burnin command is ignored. **It is important to remember that in MrBayes the burnin refers to the number of trees sampled rather than the number of generations.**
* **mcmcdiagn** tells the program to calculate a diagnostic value of convergence when it is set to “yes.” This value is calculated based on the most recent 75% of trees that have been saved. For example, if you had run MrBayes for 10,000 generations and saved a tree from every generation, then this would be calculated from trees number 2,501 through 10,000 (i.e., this is meant to coincide with the relburnin command). The authors of the program state that the value of this diagnostic signifies convergence when it drops below 0.01, although for most cases a value under 0.05 is acceptable.
* **diagnfreq** is used to specify how often MrBayes calculates its diagnostic value of convergence. While MrBayes is running, you will see this value calculated after every 5000 generations as the “Average standard deviation of split frequencies.”

Once you have finished modifying the Nexus file, **save your file**, open a terminal window (if you have not already done so) and then cd into your lab\_7 folder. MrBayes is preinstalled and can be launched by issuing the command “**mb**” (without quotes). You can load your helo-atp86.nex data file by typing the following at the command prompt:

exe helo-atp86.nex

The program will now read the file. **Stop, and carefully read over the text in the terminal window to make sure the file has been read properly.** Before proceeding, be sure of the following:

1. The most recent output in your terminal window should say “Reading mrbayes block,” followed by several commands, and then it should say “Exiting mrbayes block.”
2. There should be absolutely no messages that say “error” or “warning.”

If it appears everything worked properly, type the following at the command prompt and hit enter to start your analysis:

mcmc

By default the program simultaneously runs two independent analyses, each with four chains. The numbers you see scrolling past are the

-log(likelihood) values of the tree found by each chain for every 100th generation. The numbers surrounded by square brackets [ ] represent the cold chains. The numbers surrounded by the parentheses ( ) are the values for the heated chains. When the program completes, it will ask you if you want to continue with the analysis. Type “no” and press the return key.

We will now calculate some summary statistics that will allow us to determine the burnin. Enter the following command at the prompt:

sump

The sump command should produce a plot that looks something like the following (you may have to scroll back in your terminal window to see it). From here, you can decide how many generations need to be discarded. Read the X axis scale to determine approximately when the chains converge (i.e., when there is nothing but a solid line of asterisks across the top of the plot). This is the point, counted in the number of generations, where your burnin period will end.



To calculate your burnin, you must consider how many trees you saved in your original settings (samplefreq). Since only every 100th tree was saved, and we ran the analysis for 100,000 generations, we have sampled 1000 trees (100,000 / 100 = 1000). Now, decide on the number of generations you think you should discard based on the results you received, and divide this number by 100 to calculate your burnin value. When deciding on a burnin value it is better to throw away too many trees than not enough.

**What did you decide upon for your burnin value? Put this in your homework document.**

Now, enter the following command, where you **substitute your burnin value for X**

sump burnin=X

If you are asked to overwrite files, type “yes.” The above command should again provide you with a plot, except now it should look like “white noise.” If your plot looks something like the following (i.e., there is no discernible pattern), you have selected an appropriate burnin value.



Now at the command prompt, type the following command:

sumt

This will provide you with your consensus tree with support values (probabilities) and branch lengths. Your consensus tree is saved in your MrBayes directory with the file name “helo-atp86.nex.con.tre.” This file can be opened in FigTree. The program also saves all of your sampled trees in files with a .t extension, and the estimated parameters for each tree in .p files. You will notice there are two copies of each of these files – a “run1” and “run2” file. Remember that MrBayes by default conducts two independent analyses simultaneously, and these files represent those two analyses.

Now that we have our code working properly and we have completed our exploratory analysis, we will set up MrBayes to do a longer analysis. We will also set it to execute all commands that we otherwise enter manually. With these settings, you will be able to set up an analysis, execute your file, and allow it to complete without any additional input.

First we want to add an extra line of code to our MrBayes block that will prevent the program from prompting the user for additional input during a run.

Set autoclose=yes nowarnings=yes autoreplace=yes;

Next, we will modify a few more settings in our MrBayes block.

* Change the number of generations to 500,000.
* Change your samplefreq value to 200.
* Increase printfreq to 1000
* Based on your new samplefreq value, recalculate your burn-in, and change the burn-in value from 0 to whatever value you determined to be appropriate. **Don’t forget that burn-in in MrBayes refers to the number of trees you wish to discard**. If your burn-in exceeds the total number of trees that were sampled, the sump and sumt commands will fail to execute. For example, if you determined to discard 100 trees as burn-in: 100 trees \* 200 iterations per tree = 20,000 iterations discarded as burn-in.

After all of the other code in your MrBayes block, enter the following three lines:

mcmc;

sump;

sumt;

Your MrBayes block should now be complete. **Save your helo-atp86.nex file.** If it is functioning properly, you should be able to type “exe helo-atp86.nex” in MrBayes and your analysis will run to completion without any more input from you. This analysis will take some time to complete.

**\*\*The program will run for about 30-40 minutes. Use this time to complete the Part II activity. \*\***

**Viewing your Tree**

When MrBayes has completed its run, open your consensus tree in FigTree (File 🡪 Open…; then navigate to your “helo-atp86.nex.con.tre” file).

* If necessary, root the tree by S\_punctatus. Click on this taxon, then click the “Reroot” button.
* To make the tree easier to read, click the “Tree” menu at the top of your screen and go to “Increasing Node Order.”
* Display posterior probability values on your tree by checking “Node Labels” in the left pane. Then expand this option, and select “prob” from the “Display” drop-down box. Also change it to display only 2 significant digits (also found under the “Node Labels” menu).
* Under the “Trees” menu in the left pane, check the “Transform branches” box and select “proportional” from the drop-down menu.
* Finally, export your tree as a PDF (File 🡪 Export PDF…) with the filename **atp86\_mrbayes.pdf**. Save it in your lab\_7 folder.

**Answer the last question for Part I in your homework document.**

Part II – Simulating MCMC with iMCMC

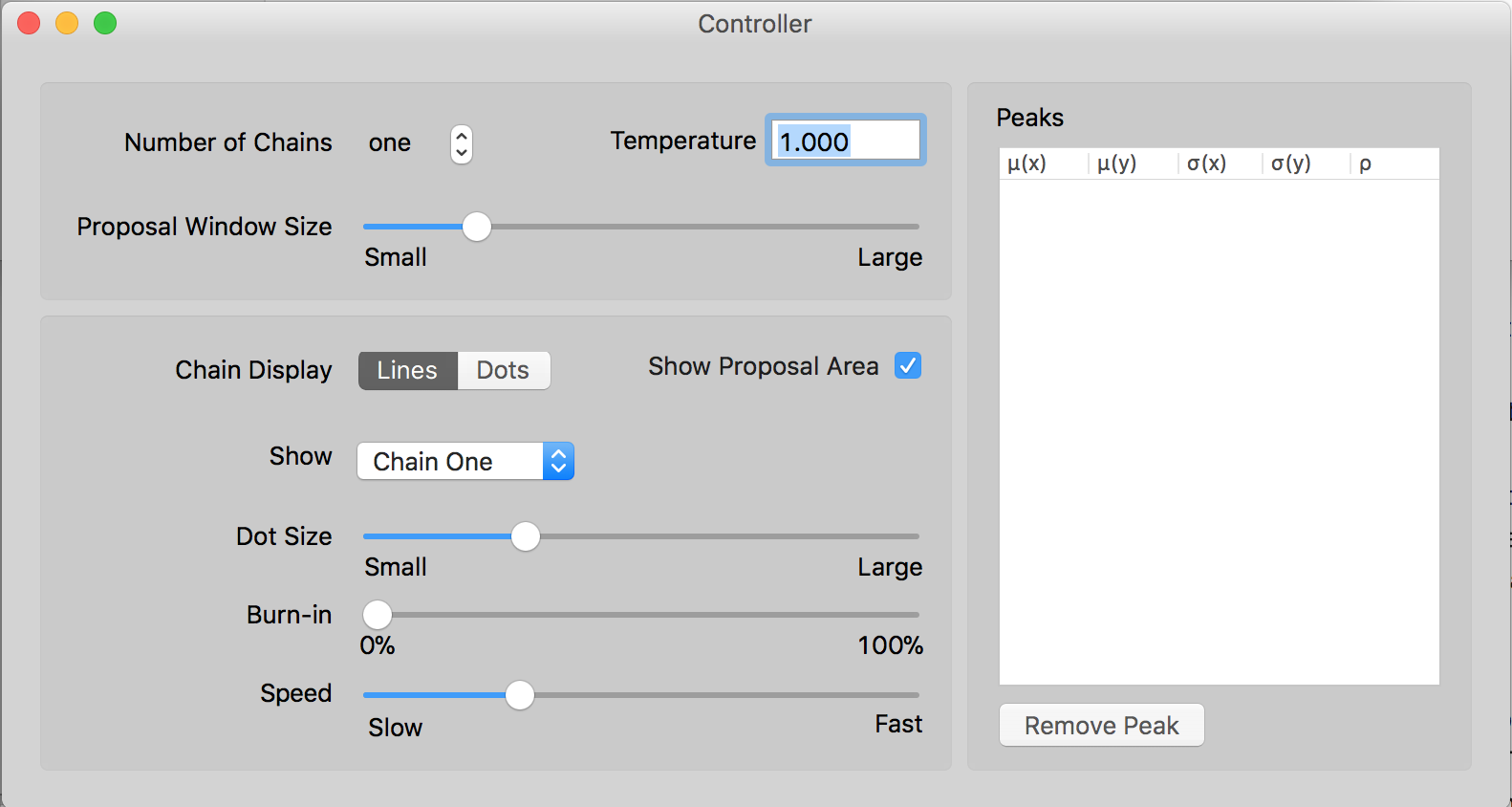
**\*\*I recommend you first start the analysis in Part I first, and come back to Part II while that is running, but you are all adults and can do what you want\*\***

In Bayesian inference of phylogenies, directly computing the (posterior) probability distribution of trees is very difficult because it would involve considering the probability of all possible hypotheses (which, for even a modest number of taxa, can be very large). Instead, we use an algorithm called Markov Chain Monte Carlo (MCMC) is to sample from that distribution. The general process as it relates to phylogenetics is:

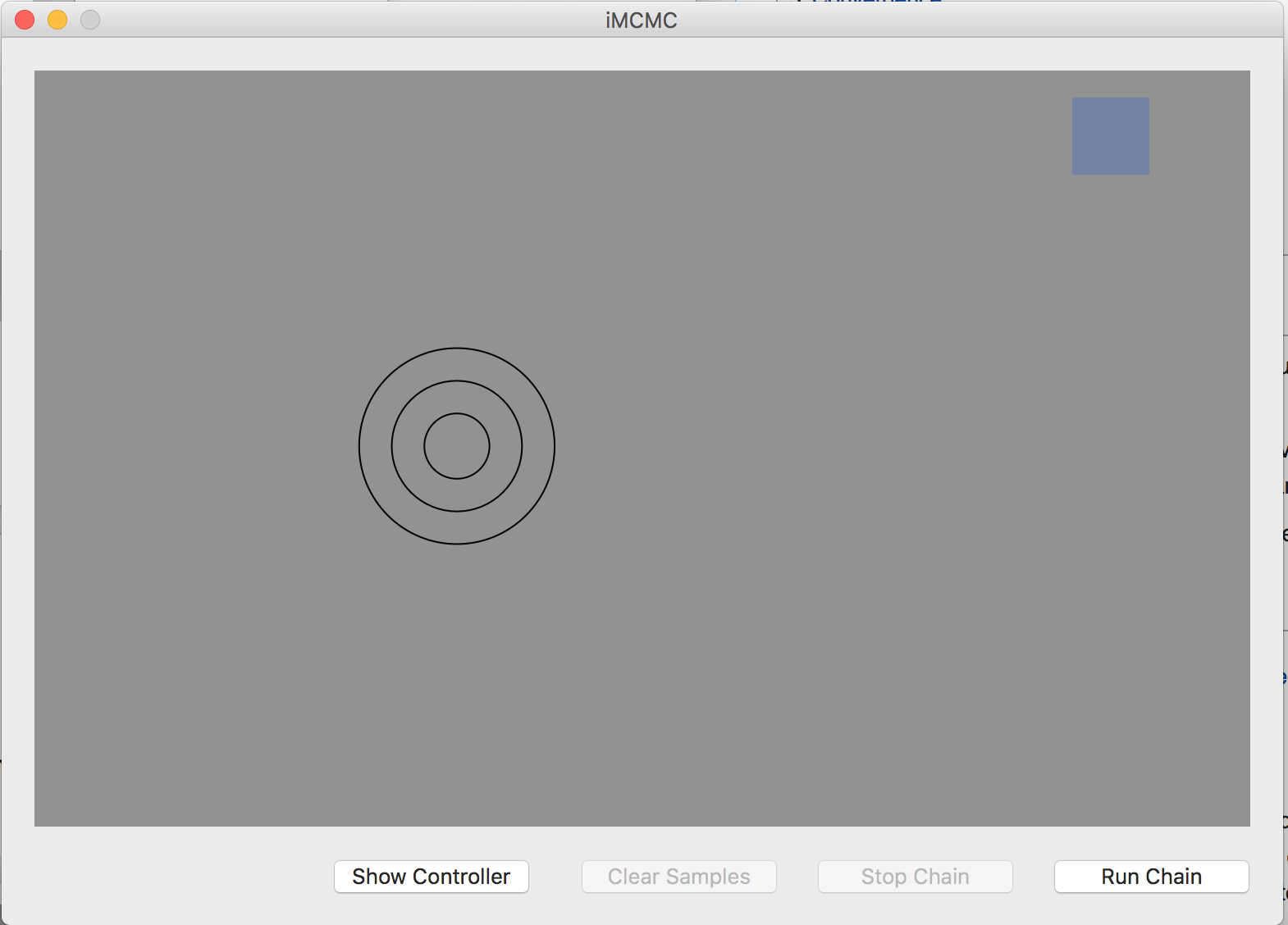
1. Start at some tree
2. Randomly draw a neighboring tree
3. Compute probabilities for those trees
4. Accept or reject the change, depending on the ratio of those probabilities
5. Return to step 2

To demonstrate the process more generally, we will use iMCMC, a package by John Heulsenbeck, one of the authors of MrBayes (the package for Bayesian phylogenetic inference we will use later). Its pretty cool, and I think really helps to visualize and understand what MCMC is doing. In this example, we will be dealing with a much simpler parameter space, with two axes: “x” and “y”. We will draw ‘peaks’ representing increased probabilities in this space, and observe how the MCMC sampling behaves under different scenarios.

First, we will simulate a simple MCMC chain, with a single ‘peak’ in parameter space. Open iMCMC (Applications -> McmcApp) and you should see a gray screen with a blue box. Click “Show Controller” to view the parameters for running iMCMC. We will leave the parameters box open for the entire time. To simulate a single chain, set parameters to something like this:



Create a single peak in ‘parameter space’ (by clicking and dragging with your mouse in the gray space), something like this:



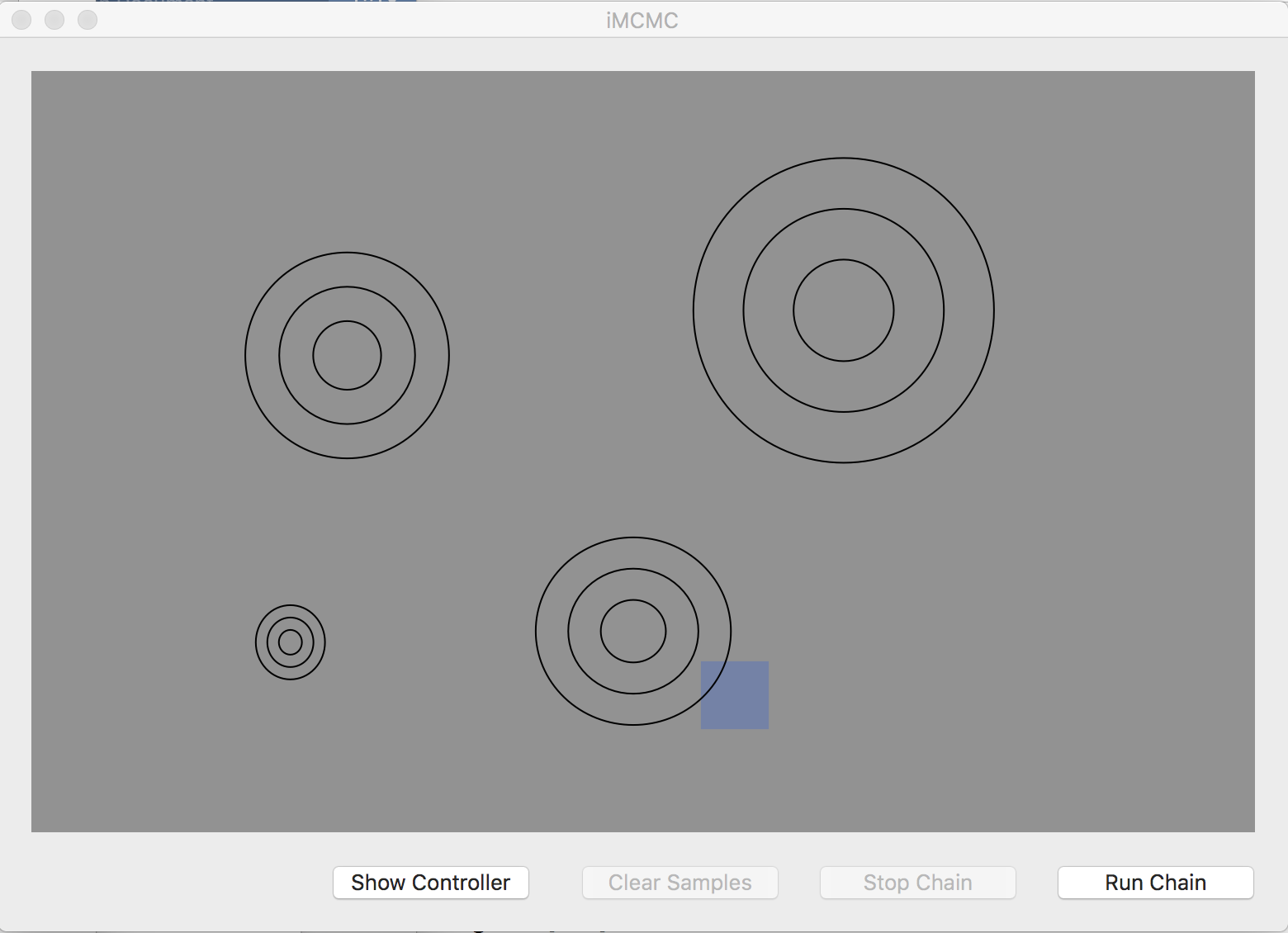
For the purposes of demonstration, please place your ‘peak’ so that it is not overlapping with the blue box (representing your starting point). This represents increased probabilities for our two variables “x” and “y” (the axes).

Click “Run Chain”. Let the chain ‘explore’ proposed parameter values for a while (you can increase the speed in the controller window if you want). Once you think your chain has adequately samples space, click “Stop Chain”, and **answer Question 1 in your homework document**.

Next, use the “Burn-in” scrollbar in the Controller panel to remove the exploratory sampling your MCMC chain did to find the peak. **Answer question 2**.

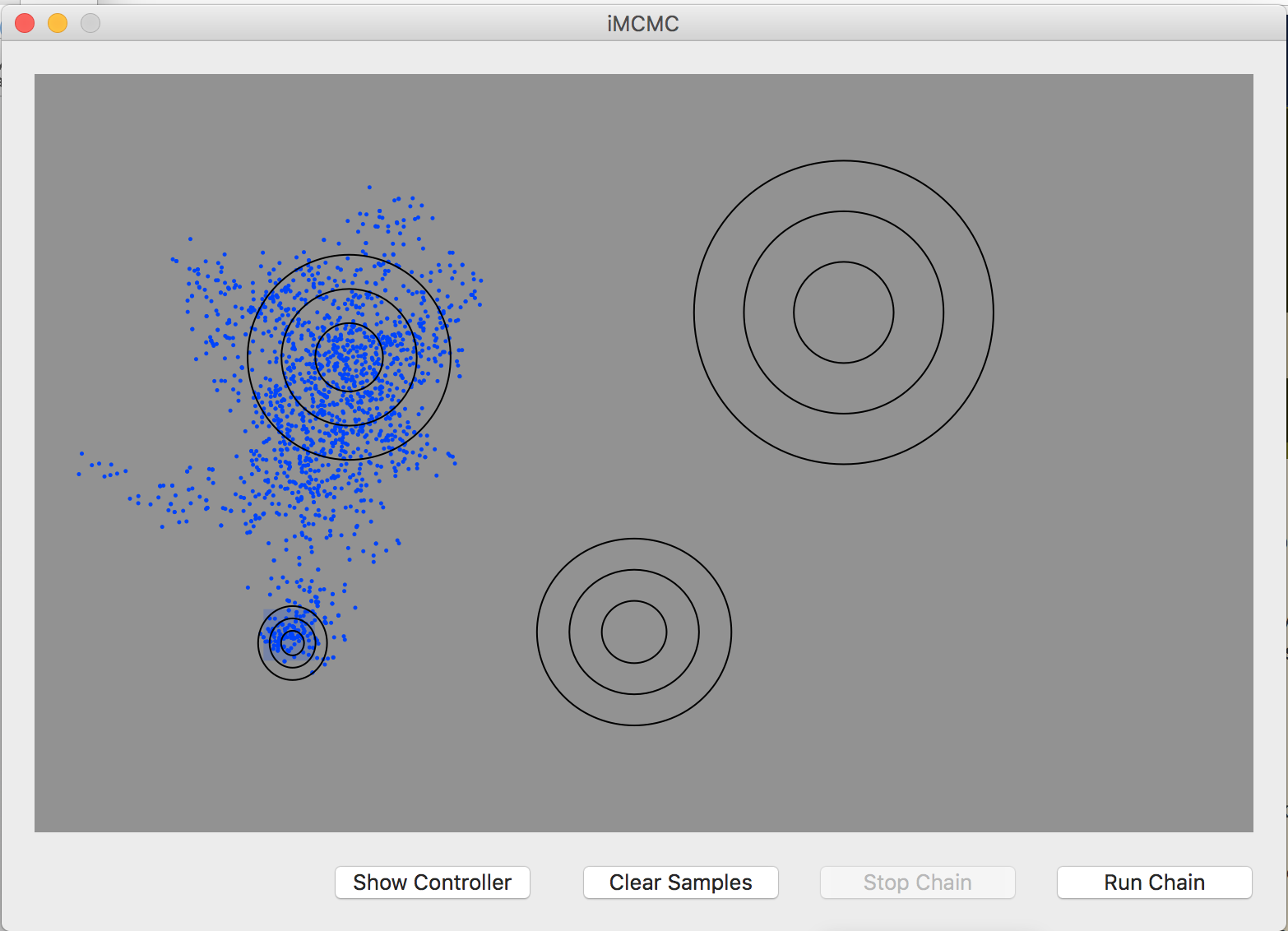
We will not investigate the role of the Proposal Size Window, which represents how large of a ‘jump’ your MCMC chain can make with each new parameter proposal. I want you to select a few different values (ranging from “Small” to “Large”) for the Proposal Size Window. You will see the size of the blue box in your iMCMC panel change. Before each trial, make sure to click “Clear Samples”. **After trying a few different values, report your findings in your homework document under question 3.**

We will next try a more complex parameter space. Click “Clear Samples”, and add a few more peaks to make a probability landscape with ‘hills’ and ‘valleys’, like so:



Return your chain back to initial settings (Burn-in=0%, medium-small proposal size window). Click “Run Chain”, and **answer question 4 in your homework document**.

You likely found that with a more complex probability landscape, your chain spent most of its time sampling one or a few separate peaks:



To help our chain to cross the ‘valleys’ between areas of high probabilities for *x* and *y*, we will use a method called Metropoulis-Coupled MCMC (MC3). Here, we introduce additional independent chains, which explore parameter space separately from our primary chain. These extra chains are referred to as “heated”, and they act as scouts by crossing valleys to find other local peaks. They do this by treating parameter space as being more flattened, which enables them to accept ‘worse’ changes than the main chain (here referred to as “cold”). At predefined number of iterations (say, every 100 or 1000 iterations), the heated chain is compared to the cold chain. If it has found a better optimum, the two chains may swap states. Since we are only sampling values (e.g. trees, or in this case, values of *x* and *y*), this helps us better characterize the probability distribution, and leads to better mixing.

Add additional chains by changing the Number of Chains in the Controller window to “two”. You can use the “Show” dropdown box to view the states of these chains. Change the “Show” dropdown to “All Chains”, and run the simulation. **Answer question 5**.

To better illustrate this, you can use the “Remove Peak” command in the Peaks box to remove these peaks, and try creating some which are further apart. Try your best to understand this, and when you are ready, move on to complete Part II.