Lab 7: Introduction to Hamiltonian Monte Carlo

NOT GRADED; SIMPLY ATTEMPT ON YOUR OWN

Getting started

You will need the following R packages. If you do not already have them installed, please do so first using the `install.packages` function.

```
require(tidyverse)
require(rstanarm)
require(magrittr)
require(rstan)
require(bayesplot)
require(loo)
require(readxl)
require(plyr)
```

For this lab, you will need one stan file. Download here:

- https://omelikechi.github.io/sta402spring26/labs/lab07/lab-07-hmc_norm_example.stan.

Download and make sure to save the file in the same folder as the R script or R markdown file you are working from.

Overview

It is finally time to get a (very!) basic introduction to Hamiltonian Monte Carlo. As we have seen so far, the benefit of Gibbs sampling is that it allows us to sample from distributions whose form we may not be able to write down, as long as we can write down the full conditionals. The drawback is that, unlike samples directly from the posterior, samples from the Gibbs sampler *eventually* behave as if they were drawn from the posterior distribution: now as we have seen in some examples and exercises already, we have to be concerned about exactly how and when this convergence occurs.

The goal of this lab is to use that intuition already developed from writing your own Gibbs samplers, to try to understand the advanced methods `Stan` is running under the hood to generate its samples.

Gibbs sampling

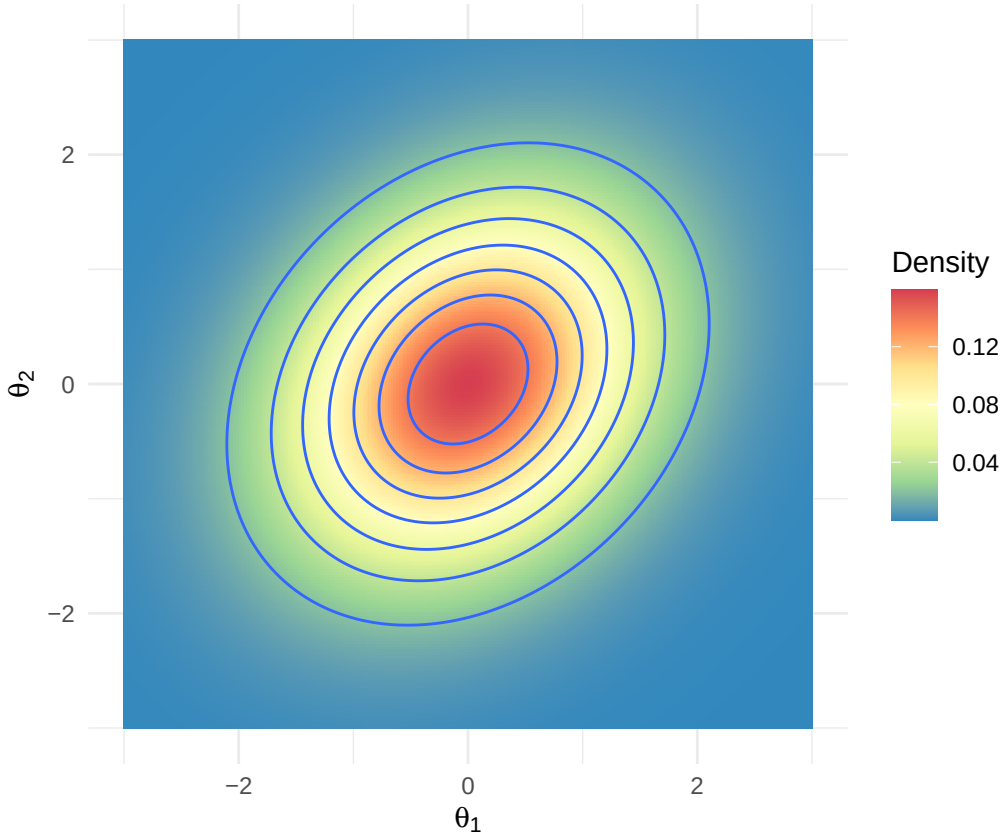
Suppose our target density is

$$p(\theta_1, \theta_2|X),$$

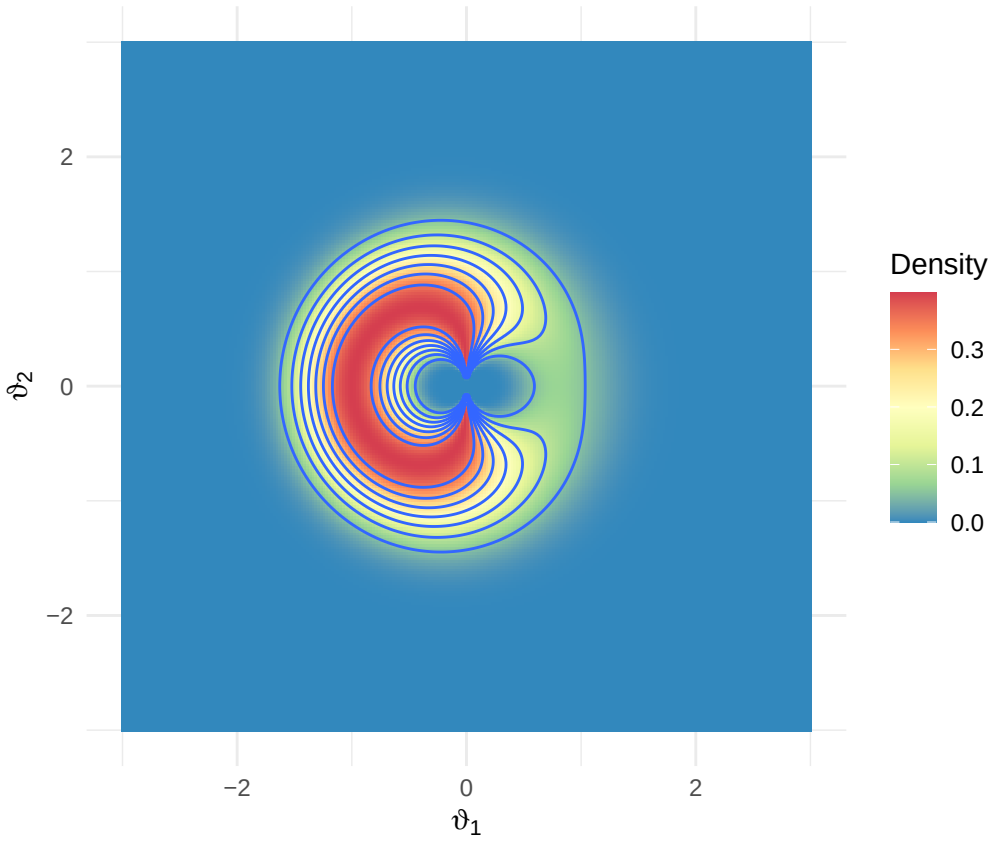
for two univariate parameters θ_1 and θ_2. As we have already seen in class, MCMC methods like the Gibbs sampling, will move iteratively through the plane, transitioning from point to point and producing a set of S samples

$$[\theta_1^{(1)}, \theta_2^{(1)}], \ldots, [\theta_1^{(S)}, \theta_2^{(S)}].$$

When the target density is nicely behaved, the exploration of a Gibbs sampler is also "nice." Regions of high density are visited with higher frequency than regions of lower density. All regions of the plane are both in theory *and* in practice accessible within a finite (and relatively small) number of transition steps. An example of a nice density on $\mathbb{R}^2$ is a bivariate normal with a modest degree of dependence between its two components θ_1, θ_2:



What's an example of a "not nice" density for moving through the plane with Gibbs transition steps? Let's look at an example of a bivariate density that would likely give the Gibbs sampler some trouble:



The problem here is that there are regions with very high density (the red peaks in the middle) right next to regions of very low density. It will take many transition steps for the Markov Chain to reach the peaks, and will also take many transitions to get out of the neighborhood of the peaked regions once it moves near it.

While the asymptotic properties of the Markov chain still hold, the sticking behavior makes it difficult to get accurate posterior summaries without taking an infinite number of samples and waiting an infinitely long time for them. As with many other problems in statistics, this problem often gets worse as the number of dimensions in the parameter space increases.

Gibbs failure modes

Even less extreme examples than the one shown above will cause Gibbs samplers to explore the sample space more slowly than we'd like. Let's look at a simpler case and write the code for our own Gibbs sampler. We will then compare it to Stan's behavior on the same problem and discuss the benefits of using HMC.

Consider data generated from a bivariate normal distribution with mean parameter $\boldsymbol{\theta} = (\theta_1, \theta_2)$ and known covariance matrix Σ and suppose we place independent normal priors on θ_1, θ_2:

$$\begin{aligned} \boldsymbol{X}_1, \ldots, \boldsymbol{X}_n &\sim N_2\left(\boldsymbol{\theta}, \Sigma\right); \\ \theta_j &\sim N(0,1) \qquad j = 1, 2. \end{aligned}$$

where each $\boldsymbol{X}_i = (X_{i1}, X_{i2})^T$. Suppose that the covariance matrix Σ is known and has the form

$$\Sigma = \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix},$$

with known correlation parameter ρ.

So far with the multivariate normal, we have derived the multivariate posterior density $p(\boldsymbol{\theta}|\Sigma, \boldsymbol{X})$. However, for the purposes of this exercise, we will sample from the following univariate full conditional densities instead.

$$p(\theta_1|\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n, \rho, \theta_2)p(\theta_2|\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n, \rho, \theta_1).$$

This way, we can explore a toy example of where Gibbs sampling faces issues, which provides an analogy to situations one might encounter in practice when direct sampling methods are not available.

Before building the Gibbs sampler to make inferences on θ_1, θ_2, first answer these questions:

1. What is the conditional density $p(\theta_1|\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n, \rho, \theta_2)$?
2. What is the conditional density $p(\theta_2|\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n, \rho, \theta_1)$?

While you have derived similar conditional normal distributions a few times now, **doing it yet again is very good practice**!

Now that you have derived the full condition densities, write some code to implement a Gibbs sampler for this model.

---

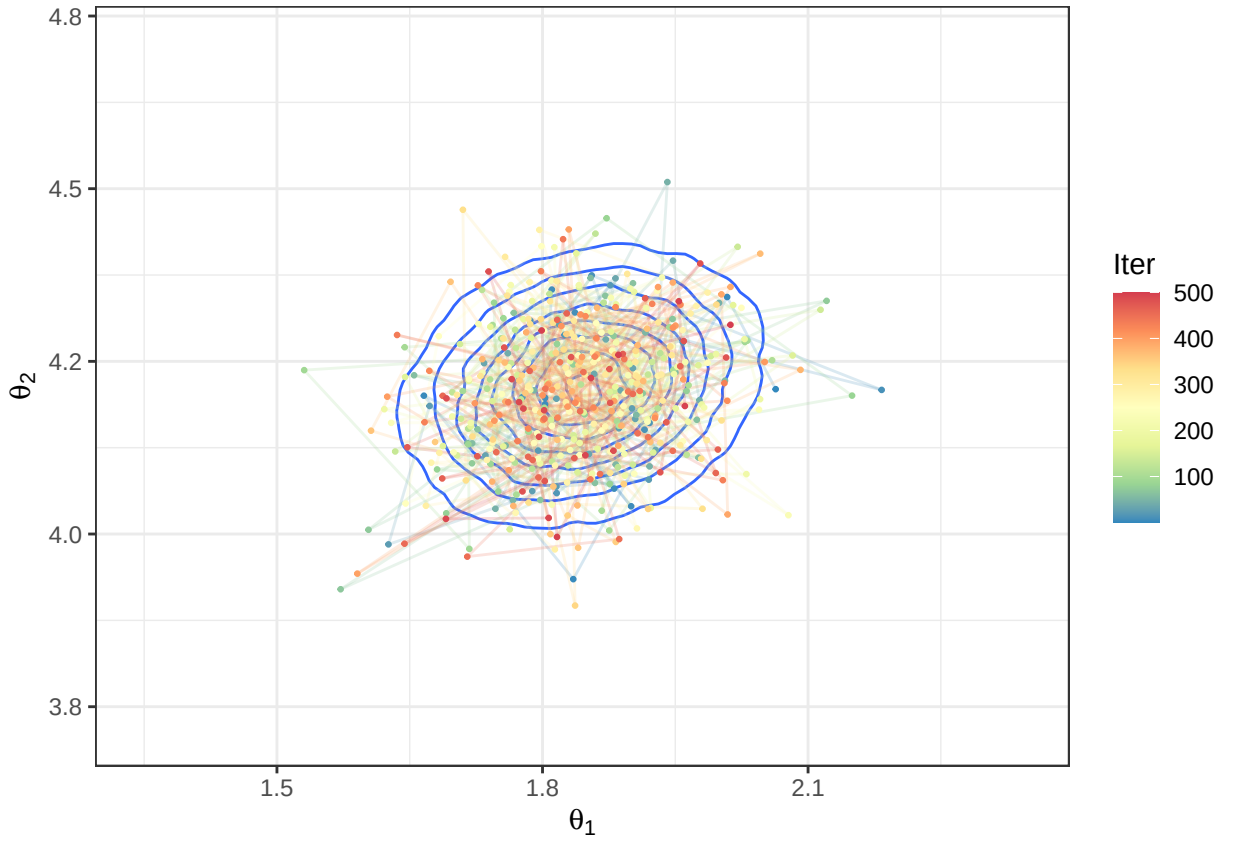
3. Specifically, write a function called `normal_gibbs_sampler` that takes as arguments: **(1)** the number of samples desired (`S`), **(2)** an $n \times 2$ matrix of data values (`X`), and **(3)** the given correlation parameter (`rho`).

   Have this function return an $S \times 2$ matrix of samples containing $[\theta_1^{(1)}, \theta_2^{(1)}], \ldots, [\theta_1^{(S)}, \theta_2^{(S)}]$, your realizations from the joint posterior $p(\boldsymbol{\theta}|\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n, \rho)$.

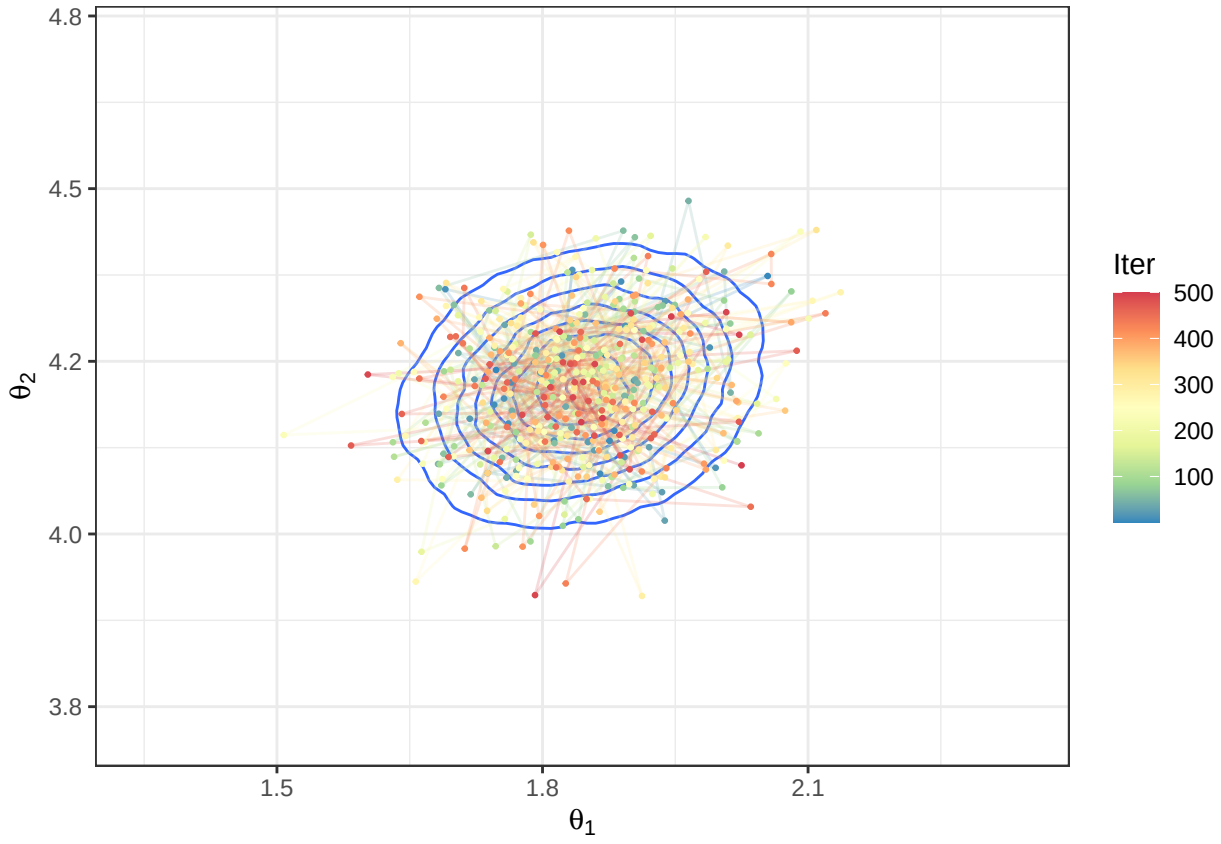
With the Gibbs sampling code in hand, let's generate samples from the posterior distribution of θ_1, θ_2 with $\rho = 0.2$. We'll do the same using `Stan`.

```
n <- 100
rho <- 0.2
X <- MASS::mvrnorm(n = n, mu = c(2, 4), Sigma = matrix(c(1, rho, rho, 1), nrow = 2))
Sigma_post <- matrix(((1-rho^2)/((n+1-rho^2)^2 - (n^2)*(rho^2)))*c(n+1-rho^2, n*rho, n*rho, n+1-rho^2),
mu_post <- n*Sigma_post%*%matrix(c(1/(1-rho^2), -rho/(1-rho^2),
                                                    -rho/(1-rho^2), 1/(1-rho^2)),
                                                    nrow = 2)%*%colMeans(X)
norm_gibbs_samps <- normal_gibbs_sampler(600, X, rho)
#
true_post <- MASS::mvrnorm(n = 100000,
                           mu = mu_post,
                           Sigma = Sigma_post)
data.frame(norm_gibbs_samps) %>%
  magrittr::set_colnames(c("theta_1", "theta_2")) %>%
  dplyr::mutate(iter = 1:n()) %>%
  dplyr::filter(iter > 100) %>%
  dplyr::mutate(iter = 1:n()) %>%
  ggplot2::ggplot() +
  geom_density2d(data = data.frame(true_post) %>%
                        magrittr::set_colnames(c("true_1", "true_2")),
                 aes(x = true_1, y = true_2)) +
  geom_path(aes(x = theta_1, y = theta_2, colour = iter), alpha = 0.2, size = 0.5) +
```

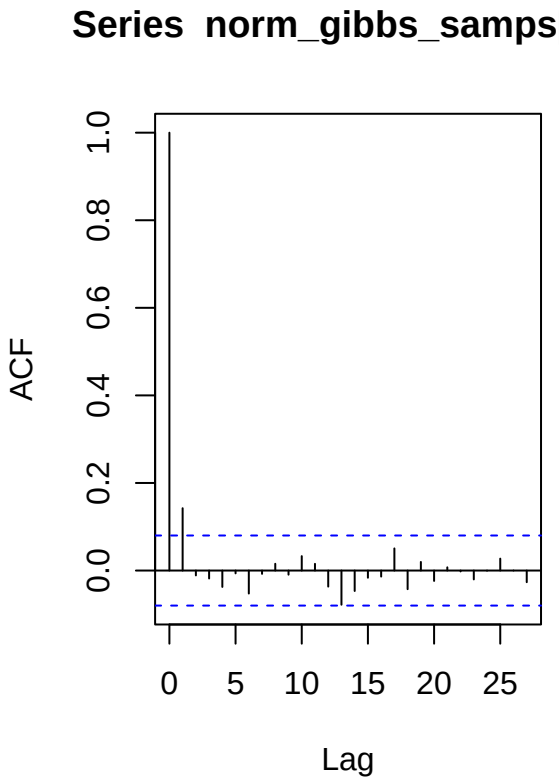
```
  geom_point(aes(x = theta_1, y = theta_2, colour = iter), size = 0.5) +
  scale_color_distiller(palette = "Spectral", name = "Iter") +
  labs(x = expression(theta[1]), y = expression(theta[2])) +
  xlim(c(mu_post[1] - 0.5, mu_post[1] + 0.5)) +
  ylim(c(mu_post[2] - 0.5, mu_post[2] + 0.5))
```

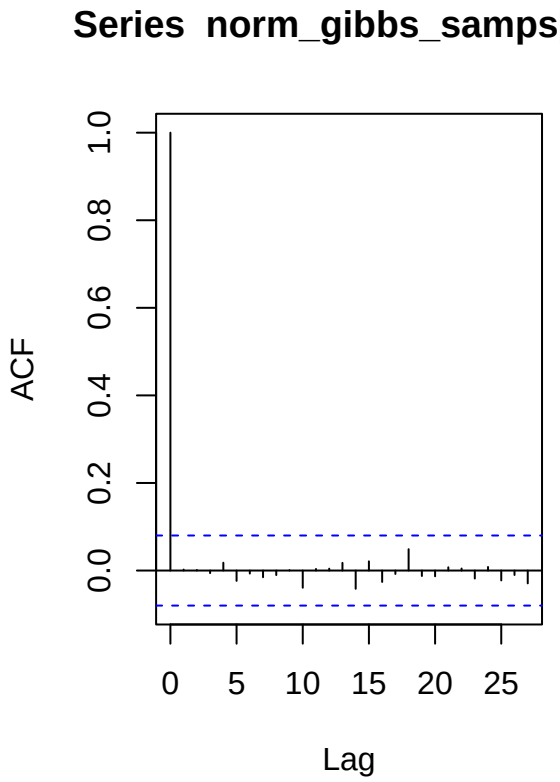



```
#
stan_res <- rstan::stan("lab-07-hmc_norm_example.stan", data = list(X = X,
                                                                    N = nrow(X),
                                                                    Sigma = matrix(c(1, rho, rho, 1), nrow = 2
                        chains = 1, iter = 600, warmup = 100, verbose = F, refresh = 0) %>%
            rstan::extract()
#
data.frame(stan_res$theta) %>%
  magrittr::set_colnames(c("theta_1", "theta_2")) %>%
  dplyr::mutate(iter = 1:n()) %>%
  ggplot2::ggplot() +
  geom_density2d(data = data.frame(true_post) %>%
                        magrittr::set_colnames(c("true_1", "true_2")),
                 aes(x = true_1, y = true_2)) +
  geom_path(aes(x = theta_1, y = theta_2, colour = iter), alpha = 0.2, size = 0.5) +
  geom_point(aes(x = theta_1, y = theta_2, colour = iter), size = 0.5) +
  scale_color_distiller(palette = "Spectral", name = "Iter") +
  labs(x = expression(theta[1]), y = expression(theta[2])) +
  xlim(c(mu_post[1] - 0.5, mu_post[1] + 0.5)) +
  ylim(c(mu_post[2] - 0.5, mu_post[2] + 0.5))
```

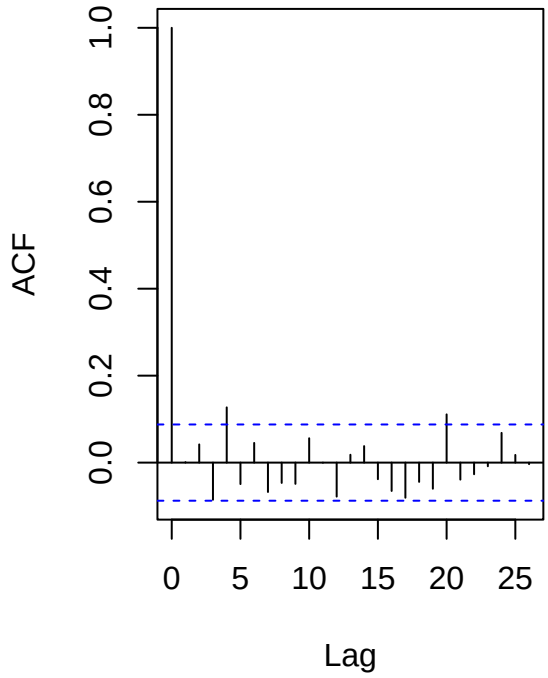


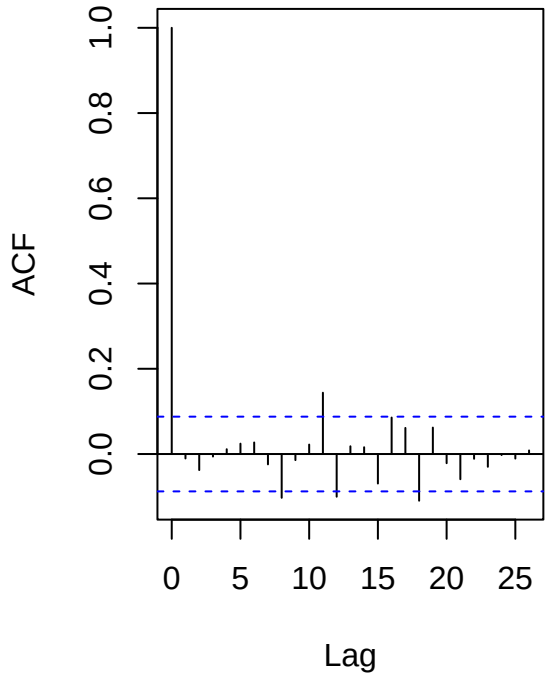

```
#
par(mfrow = c(1,2))
acf(norm_gibbs_samps[,1])
acf(norm_gibbs_samps[,2])
```





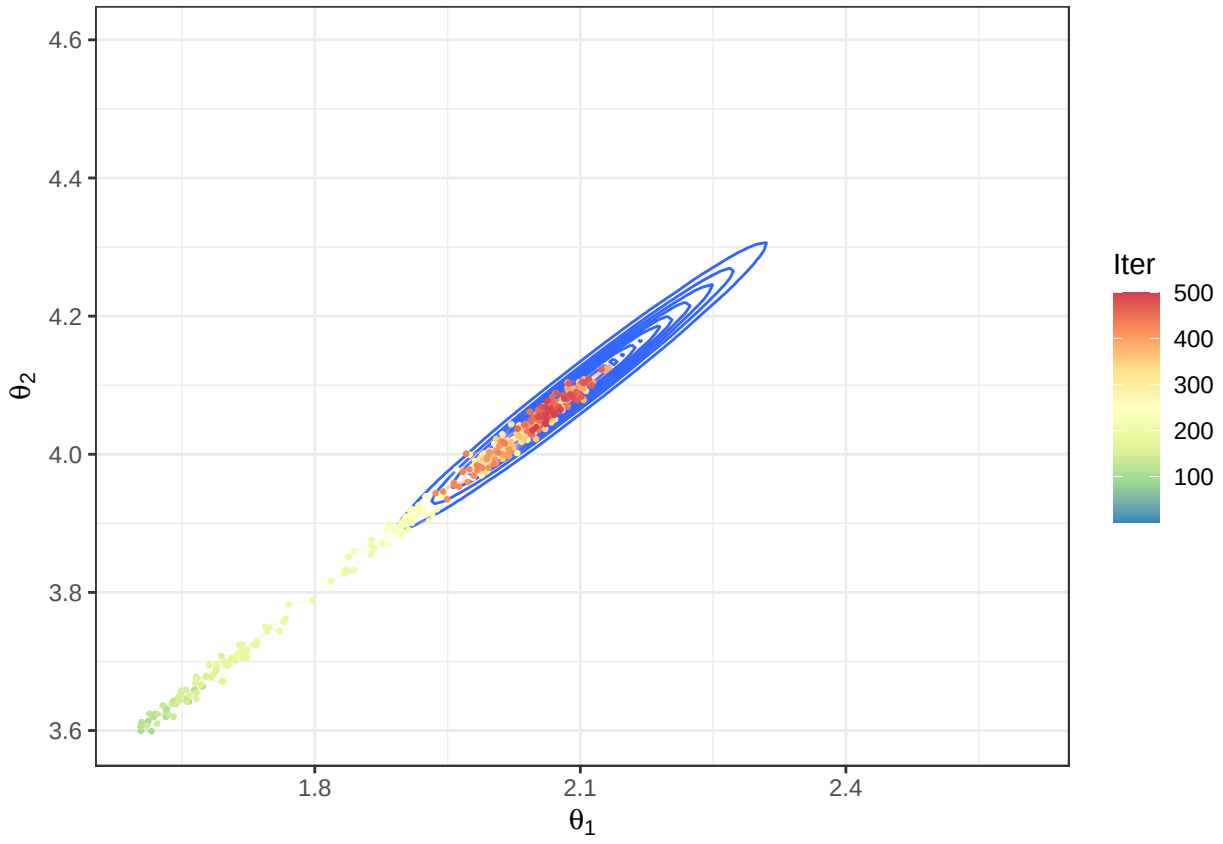

```
#
par(mfrow = c(1,2))
acf(stan_res$theta[,1])
acf(stan_res$theta[,2])
```



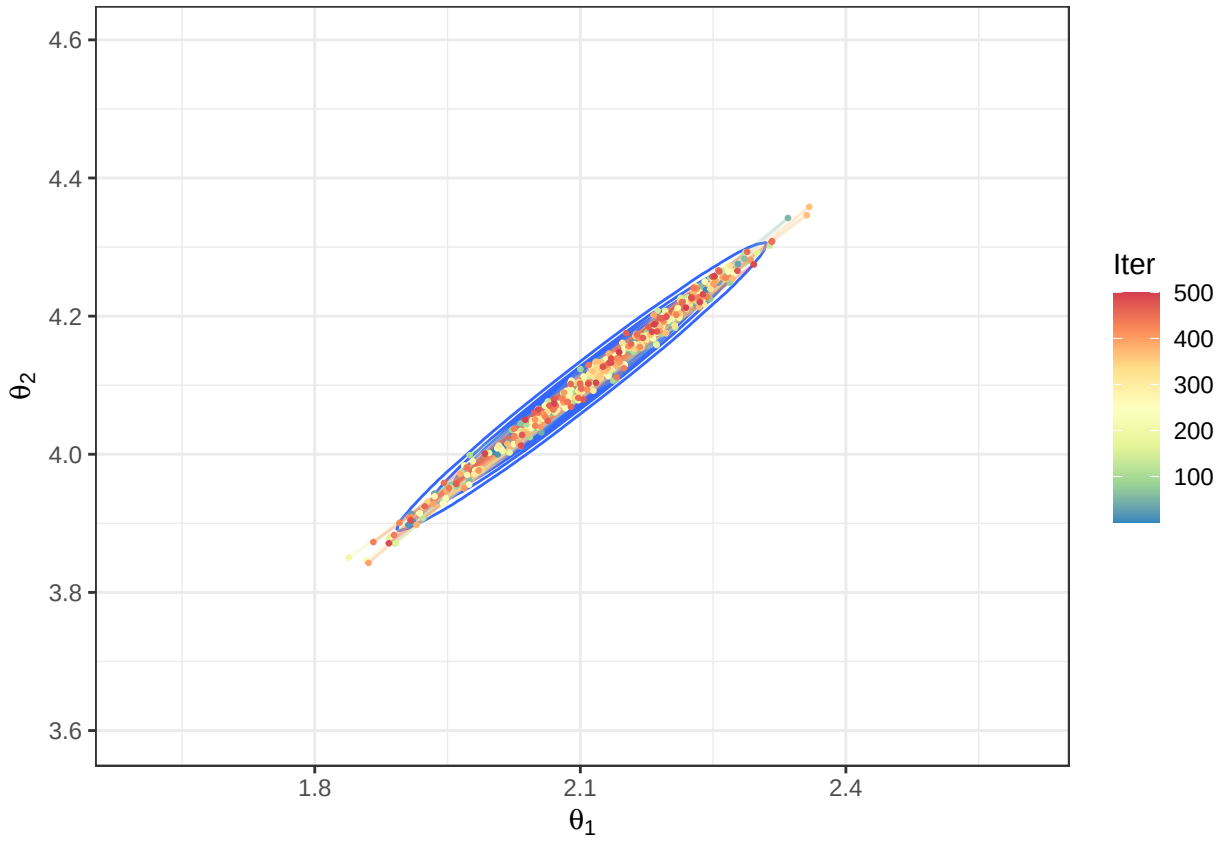


The Gibbs sampling results and the HMC results look pretty similar! What happens when $\rho = 0.995$?

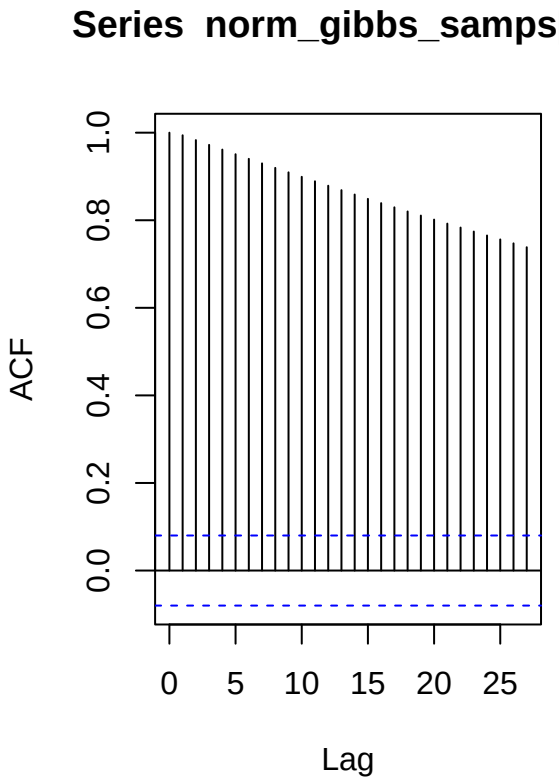
```
n <- 100
rho <- 0.995
X <- MASS::mvrnorm(n = n, mu = c(2, 4), Sigma = matrix(c(1, rho, rho, 1), nrow = 2))
Sigma_post <- matrix(((1-rho^2)/((n+1-rho^2)^2 - (n^2)*(rho^2)))*c(n+1-rho^2, n*rho, n*rho, n+1-rho^2),
mu_post <- n*Sigma_post%*%matrix(c(1/(1-rho^2), -rho/(1-rho^2),
                                                -rho/(1-rho^2), 1/(1-rho^2)),
                                                nrow = 2)%*%colMeans(X)
norm_gibbs_samps <- normal_gibbs_sampler(600, X, rho)
#
true_post <- MASS::mvrnorm(n = 100000,
                           mu = n*Sigma_post%*%(matrix(c(1/(1-rho^2), -rho/(1-rho^2),
                                                -rho/(1-rho^2), 1/(1-rho^2)),
                                                nrow = 2)%*%colMeans(X)),
                           Sigma = Sigma_post)
#
data.frame(norm_gibbs_samps) %>%
  magrittr::set_colnames(c("theta_1", "theta_2")) %>%
  dplyr::mutate(iter = 1:n()) %>%
  dplyr::filter(iter > 100) %>%
  dplyr::mutate(iter = 1:n()) %>%
  ggplot2::ggplot() +
  geom_density2d(data = data.frame(true_post) %>%
                        magrittr::set_colnames(c("true_1", "true_2")),
                 aes(x = true_1, y = true_2)) +
  geom_path(aes(x = theta_1, y = theta_2, colour = iter), alpha = 0.2, size = 0.5) +
  geom_point(aes(x = theta_1, y = theta_2, colour = iter), size = 0.5) +
  scale_color_distiller(palette = "Spectral", name = "Iter") +
  labs(x = expression(theta[1]), y = expression(theta[2])) +
  xlim(c(mu_post[1] - 0.5, mu_post[1] + 0.5)) +
  ylim(c(mu_post[2] - 0.5, mu_post[2] + 0.5))
```

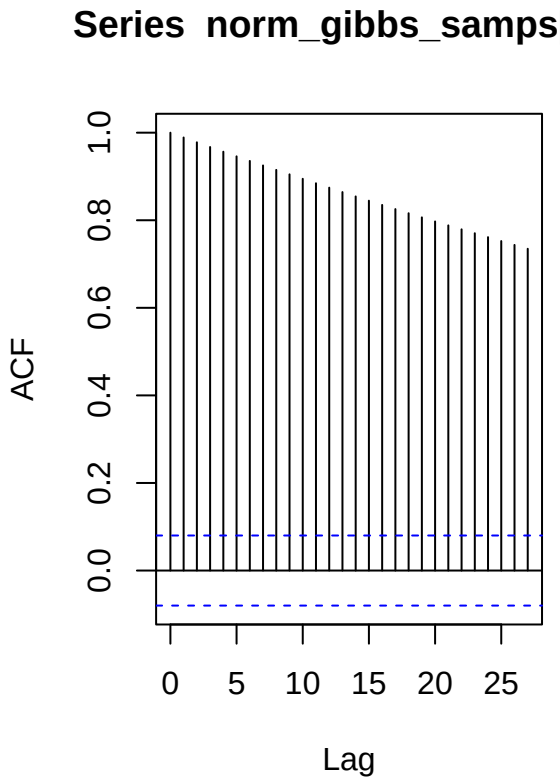



```
#
stan_res <- rstan::stan("lab-07-hmc_norm_example.stan", data = list(X = X,
                                                                    N = nrow(X),
                                                                    Sigma = matrix(c(1, rho, rho, 1), nrow = 2
                        chains = 1, iter = 600, warmup = 100, verbose = F, refresh = 0) %>%
            rstan::extract()
data.frame(stan_res$theta) %>%
  magrittr::set_colnames(c("theta_1", "theta_2")) %>%
  dplyr::mutate(iter = 1:n()) %>%
  ggplot2::ggplot() +
  geom_density2d(data = data.frame(true_post) %>%
                        magrittr::set_colnames(c("true_1", "true_2")),
                 aes(x = true_1, y = true_2)) +
  geom_path(aes(x = theta_1, y = theta_2, colour = iter), alpha = 0.2, size = 0.5) +
  geom_point(aes(x = theta_1, y = theta_2, colour = iter), size = 0.5) +
  scale_color_distiller(palette = "Spectral", name = "Iter") +
  labs(x = expression(theta[1]), y = expression(theta[2])) +
  xlim(c(mu_post[1] - 0.5, mu_post[1] + 0.5)) +
  ylim(c(mu_post[2] - 0.5, mu_post[2] + 0.5))
```

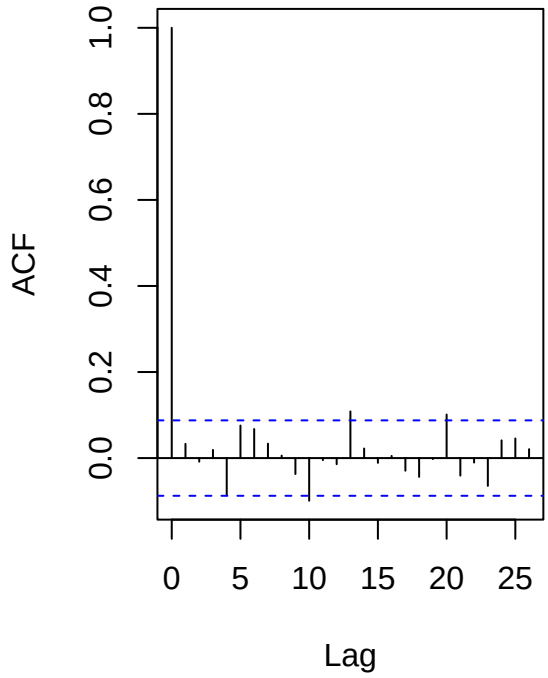


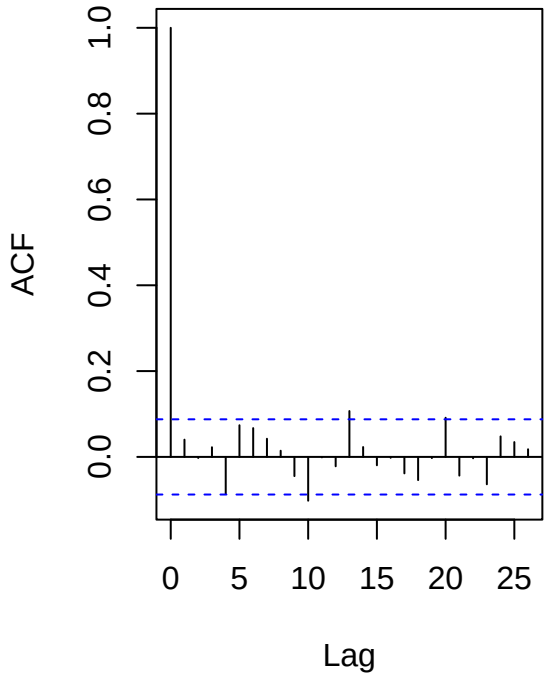

```
#
par(mfrow = c(1,2))
acf(norm_gibbs_samps[,1])
acf(norm_gibbs_samps[,2])
```






```
#
par(mfrow = c(1,2))
acf(stan_res$theta[,1])
acf(stan_res$theta[,2])
```





Please answer these questions:

4. How do the results of the Gibbs sampler differ from those obtained from HMC?
5. Why do the samples from the Gibbs sampler exhibit this behavior?

Hamiltonian Monte Carlo (HMC)

To learn the ins and outs of HMC, please read Michael Betancourt's A Conceptual Introduction to Hamiltonian Monte Carlo. For now, we will limit ourselves to a few key details:

1. HMC encourages better Markov Transitions

Hamiltonian Monte Carlo methods are like other MCMC methods in that they create Markov Chains that converge to the target distribution. The difference lies in how transitions from state to state are chosen. HMC creates transitions that *efficiently explore the parameter space* by using concepts from Hamiltonian mechanics.

2. Hamilton's equations and "phase space"

In Hamiltonian mechanics, a physical system is completely specified by positions ($\mathbf{q}$) and momenta ($\mathbf{p}$). A space defined by these coortinates is called "phase space." If the parameters of interest in a typical MCMC method are denoted as $q_1, \ldots, q_K$, then HMC introduces auxiliary "momentum" parameters $p_1, \ldots, p_K$ such that the algorithm produces draws from the joint density:

$$\pi(\mathbf{q}, \mathbf{p}) = \pi(\mathbf{p}|\mathbf{q})\pi(\mathbf{q})$$

Note that if we marginalize over the p_k's, we recover the marginal distribution of the q_k's. Therefore, if we create a Markov Chain that converges to $\pi(\mathbf{q}, \mathbf{p})$, we have immediate access to samples from $\pi(\mathbf{q})$, which is our target distribution.

At each iteration of the sampling algorithm, HMC implementations make draws from some distribution $\pi(\mathbf{p}|\mathbf{q})$ (often it is actually independent of $\mathbf{q}$; the choice of momentum distribution is important but not discussed here) and then *evolve the system* $(\mathbf{p}, \mathbf{q})$ to obtain the next sample of $\mathbf{q}$. What does that mean?

Hamilton's equations describe the time evolution of the system in terms of the **Hamiltonian**, $\mathcal{H}$, which usually corresponds to the total energy of the system:

$$\frac{d\mathbf{p}}{dt} = -\frac{\partial \mathcal{H}}{\partial \mathbf{q}} = -\frac{\partial K}{\partial \mathbf{q}} - \frac{\partial V}{\partial \mathbf{q}} \tag{1}$$

$$\frac{d\mathbf{q}}{dt} = +\frac{\partial \mathcal{H}}{\partial \mathbf{p}} = +\frac{\partial K}{\partial \mathbf{p}} \tag{2}$$

$$\tag{3}$$

$$\mathcal{H}(\mathbf{p}, \mathbf{q}) = K(\mathbf{p}, \mathbf{q}) + V(\mathbf{q}) \tag{4}$$

Here $K(\mathbf{p}, \mathbf{q})$ represents the **kinetic energy** of the system and $V(\mathbf{q})$ represents the **potential energy** of the system. HMC samplers set the kinetic energy component equal to the negative logarithm of the momentum distribution, and set the potential energy component equal to the negative logarithm of distribution over the target parameters.

To "evolve the system" is to move $(\mathbf{p}, \mathbf{q})$ forward in "time," i.e. to change the values of $(\mathbf{p}, \mathbf{q})$ according to Hamilton's differential equations. If one stares long enough, one can see that the first equation corresponds to saying:

> "The differential change in momentum parameters over time is governed in part by the differential information of the density over the target parameters."

In some sense, moving $(\mathbf{p}, \mathbf{q})$ forward in time according to Hamilton's equations changes the parameters in a way that is "guided" by the gradient of the target distribution.

3. So what does Stan do?

Stan uses HMC sampling. So under the hood, it is

- Sampling parameters $\mathbf{p}_t, \mathbf{q}_t$.
- Evolving $\mathbf{p}_t, \mathbf{q}_t$ forward in time according to Hamilton's equations to obtain $\mathbf{p}_{t+1}, \mathbf{q}_{t+1}$.
- Repeating.

There are *many* more details to HMC, but this is the gist.

Acknowledgement

This lab was created by Jordan Bryan and Becky Tang.