Series 6

1. **Hamiltonian MC using RStan**

   The goal of this exercise is to use the R package RStan to simulate from the posterior of a Bayesian model. Assume that X is distributed as

   $$X \sim N(\theta, \sigma^2).$$

   For the parameters θ and σ^2, use the following priors

   $$\theta \sim N(\xi, \kappa^2), \quad 1/\sigma^2 \sim Gamma(\gamma, \lambda).$$

   a) Simulate data according to

   $$X \sim N(-2, 3^2),$$

   and use the R package RStan to sample from the posterior distribution. For the hyper-parameters, use

   $$\xi = 0, \kappa = 10000, \gamma = 1, \lambda = 1.$$

   b) Inspect trace plots, autocorrelation as well as the bivariate posterior distribution.

   Hints:

   - Install RStan by following the instructions on https://github.com/stan-dev/rstan/wiki/RStan-Getting-Started
   - Create a .stan file to define your model. Include a "transformed parameters" block in order to define the inverse gamma prior on the variance σ^2.
   - See the R example from the lecture for an example of the use of RStan. For more information, see the stan "user's guide and reference manual" available on http://mc-stan.org/users/documentation/

2. **Own implementation of Hamiltonian MC**

   The goal of this exercise is to implement a Hamiltonian Monte Carlo algorithm. We use the schools data and the same model as shown in the RStan example during the lecture. I.e, we use the following model

   $$y_j \sim N(\theta_j, \sigma_j^2)$$

   and

   $$\theta_j \sim N(\mu, \tau^2),$$

where y_j are the observed school effects and σ_j^2 are the known standard error estimates. The posterior can be written as

$$p(\theta, \mu, \tau) \propto \prod_{j=1}^{J} p(y_j|\theta_j, \sigma_j^2) p(\theta_j|\mu, \tau^2),$$

where we have assumed that $p(\mu, \tau) \propto 1$ and where both $p()$'s are normal densities.

a) Construct a HMC algorithm using the leap-frog method in order to sample from the posterior.

b) Try different values of T, m_i, and the step size ε in order that the algorithm mixes well (in particular, low autocorrelation of the parameter samples). Inspect trace plots, autocorrelation as well as the scatter plots of the posterior distribution.

Hints:

- You can use the following code skeleton.

```
##Read data
schools <- read.csv("schools.csv", header=TRUE)
J <- nrow(schools)
y <- schools$estimate
sigma <- schools$sd

##Log posterior density
log_post <- function(th, y, sigma){
  J <- length(th) - 2
  theta <- th[1:J]
  mu <- th[J+1]
  tau <- th[J+2]
  if (is.nan(tau) | tau<=0)
    return(-Inf)
  else{
    log_prior <- XXXXX
    log_likelihood <- XXXXX
    return(XXXXX)
  }
}
##Gradient of log posterior density
grad_log_post <- function(param, y, sigma){
  J <- length(param) - 2
  theta <- param[1:J]
  mu <- param[J+1]
  tau <- param[J+2]
```

```
  if (tau<=0)
    return(c(rep(0,J),0,0))
  else {##calculate gradient
    d_theta <- XXXXX
    d_mu <- XXXXX
    d_tau <- XXXXX
    return(XXXXX)
  }
}
##Function that does on simulation step and returns the next value of the chain
hmc_iteration <- function(param, y, sigma, epsilon, L, M) {
  M_inv <- 1/M
  d <- length(param)
  u <- rnorm(d, 0, sqrt(M))
  param_old <- param
  log_H_old <- XXXXX
  u <- u + 0.5*epsilon*XXXXX
  for (l in 1:L){
    param <- param + epsilon*XXXXX
    if(l==L) u <- u + 0.5*XXXXX else u <- u + epsilon*XXXXX
  }
  log_H_prop <- XXXXX
  r <- exp(XXXXX)
  if (is.nan(r)) r <- 0
  ##acceptance probability
  p_jump <- XXXXX
  XXXXX
  return(XXXXX)
}
```

- See also the R example from the lecture.