# VectorBiTE Training 2019
# Methods Workshop

## Introduction to Bayesian Computation and MCMC with JAGS


www.vectorbite.org

# Learning Objectives

1. Understand the basic principles underlying Bayesian modeling methodology
2. Introduce how to use Bayesian inference for real-world problems
3. Introduce computation tools to perform inference for simple models in R (how to turn the Bayesian crank)
4. Appreciate the need for sensitivity analysis, model checking and comparison, and the potential dangers of Bayesian methods.

# Numerical Methods

Most of the time we can't get a nice analytic form for a posterior distribution. If we go back to the full Bayes theorem:

$$\Pr(\theta|Y) = \frac{\mathcal{L}(\theta; Y)f(\theta)}{\Pr(Y)}$$

We are usually specifying the likelihood and the prior but we often don't know the normalizing constant in the denominator. Without this, the probabilities don't properly integrate to 1 and we can't make probability statements. We need a way to approximate the distribution. We'll use Monte Carlo methods.

# Stochastic Simulation/Computation

Stochastic simulation is a way to understand variability in a system and for calculating quantities that may be difficult or impossible to obtain directly.

Monte Carlo (MC) methods are "a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results." - Wikipedia

## How does it work?

Run a simulation/computer calculation (with some component that is "random") many many times in order to obtain the distribution of an unknown probabilistic quantity.

A basic algorithm:

1. Obtain random deviate(s) from a probability distribution
2. Make a calculation from your system
3. Record the result of the calculation to save it for later
4. Repeat many times

We typically have three reasons to use MC

1. Explore possible patterns/behaviors that a model can exhibit.
2. Create synthetic data to use in place of real data to test estimation procedures.
3. Understand and quantify uncertainty.

# MC for Bayesian Statistics

We use Monte Carlo (MC) methods to generate random deviates in the right ratios from the target posterior called "draws" or samples. We then use these draws to approximate our distribution and make inference statements (estimates, CIs, etc).

We can also use the draws to calculate the posterior distribution of **any function of our estimated parameters**. As the number of draws/samples gets large we can approximate these quantities arbitrarily high precision.

# The "plug-in principle"

Using MC to perform these calculations (and to propagate the uncertainty) rests on the idea of the plug-in principle:

A summary statistic or other feature of a distribution (e.g. expected value) can be approximated by the same summary/feature of an *empirical sample* from that distribution (e.g., sample mean).

The approximation becomes more accurate if the number of samples is very large.

**Example:** Numerical 92% CI of a normal distribution with mean $\mu$ and variance $\sigma^2$.

Imagine we want to find, for some unknown reason, the central 92% CI for a normal distribution. How can we calculate this without using a look-up table, or similar function?

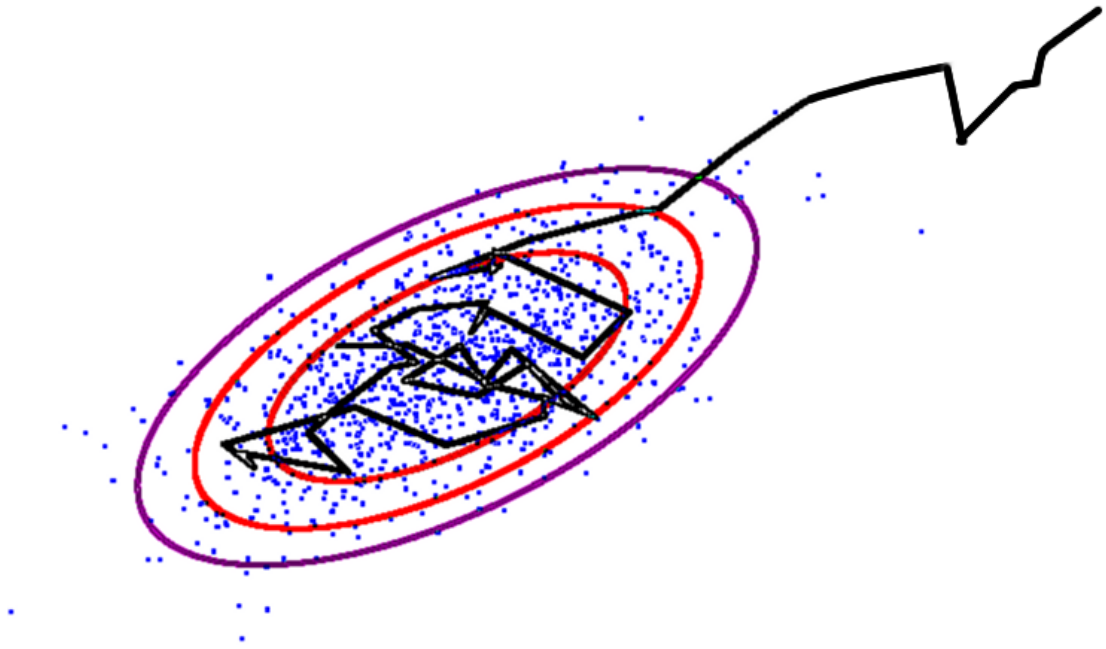
If we are able to generate samples from the desired distribution (which we'll take as given for now), we can use MC and the plug-in principle as follows

1. Generate many samples from the target distribution (say $N = 2000$, to get good estimates).
2. Find the $\alpha/2$ and $1 - \alpha/2$ empirical quantiles (here 4% and 96%). For example these can be approximated by the $N \times \left(\frac{\alpha}{2}, 1 - \frac{\alpha}{2}\right)$ order statistics.
3. You're done.

# Markov Chain MC (MCMC)

The most commonly used numerical algorithm for generating posterior samples is MCMC.

A Markov Chain is a randomly generated sequence of numbers where each draw depends on the one immediately preceding it $\rightarrow$ random walk.



Plot – Ian Murray (http://mlg.eng.cam.ac.uk/zoubin/tut06/mcmc.pdf)

# Gibbs Sampling

One specific algorithm that is commonly used is Gibbs Sampling.

Gibbs sampling leverages the *conditional* distributions of parameters to generate samples by proposing them one at a time. This is the algorithm implemented in the popular Bayesian packages BUGS, WinBUGS, and JAGS/`rjags`.

We will treat Gibbs sampling and other of the numerical methods as mostly "black boxes". We'll learn to diagnose output from these later on in the practical component.

# What do we do with Posterior Samples?

We can treat the draws much like we would data:

- Calculate posterior summaries (mean, median, mode, etc) just like we would a data sample
- Calculate precision of the summaries (e.g., sample variance)
- CIs via quantiles (order statistics of the data) or HPD intervals (using `CODA` package in R)

If the samples are parameters in a complex model, we can plug them all in, one at a time to get a range of possible predictions from the model (we'll see this in the practical bit, later on).

# How do we compare models?

The simplest way that we will use to compare models is via the Deviance Information Criterion (DIC). Like AIC and BIC, DIC seeks to judge a model on how well it fits, penalized by the complexity of the model.

$$DIC = D(\bar{\theta}) + 2p_D$$

where:

- Deviance: $D(\theta) = -2\log(\mathcal{L}(\theta; y)) + C$
- Penalty: $p_D = \bar{D} - D(\bar{\theta})$
- $D(\bar{\theta})$: deviance at the posterior mean of $\theta$
- $\bar{D}$: average deviance across the posterior samples.

$\rightarrow$ Already implemented in JAGS!

# Bayesian using JAGS

JAGS implements a version of Gibbs sampling in a fairly easy to use package.

That is, once you specify the appropriate *sampling distribution/likelihood* and any *priors* for the parameters, it will sequentially use MC to obtain samples from the posterior in the right ratios so that we can calculate whatever we want.

# Specifying models in JAGS

The trickiest and most important part of each analysis is properly specifying the model for all of the data that you want to fit. Before you begin to code, you need to decide:

- What is the relationship between your predictors and your response?
- What kind of probability distribution should you use to describe your response variable?
- Are there any constraints on your parameters or responses that you need to encode in your prior or likelihood, respectively?

# Example: Midge data?

NOTE: Your loop(s) need to go over every data point that you want use in the inference process!