EP5
Full Bayesian Significance Test
the sharp hypothesis of Hardy Weinberg Equilibrium

André Vinícius Rocha Pires
10737290

June 2019

> Well, I stand up next to a
> mountain... and I chop it down
> with the edge of my hand
>
> *Jimmi Hendrix - Voodoo Child*

1 Introduction

This is the final report on our classes about Monte Carlos Methods in numerical integration. Here, we will make our last experience, the use of Markov Chain Monte Carlo(MCMC) on a statistic test called "Full Bayesian Significance Test"(FBST), a Bayesian measure of evidence for precise hypotheses. Our hypothesis will be the Hardy Weinberg Equilibrium(HWE), which will be tested for some sets of data.

Being that we have already wrote about MCMC integration on the latter report, this time only the details of the method's new use will be discussed.

2 What is the Hardy-Weinberg Equilibrium

The idea of the HWE is very simple. It says that, under some conditions, the allele frequency of a Mendelian population will stay the same through the generations. Our work here is to verify if this equilibrium was achieved for a case with two alleles, which combined gives three genotypes.

Our sharp hypothesis, then, is: given that the probabilities of each genotype are θ_1, θ_2, and θ_3, and $\theta_1+\theta_2+\theta_3 = 1$, the equilibrium occurs if $\theta_3 = (1-\sqrt{\theta_1})^2$.

Besides this, we have the likelihood function:

$$f(\theta_1, \theta_2, \theta_3 \mid x_1, x_2, x_3) \propto \theta_1^{x_1}\theta_2^{x_2}\theta_3^{x_3}$$

Where x_1, x_2, x_3 are the occurrences of their respectives thetas.

3 Code Documentation

3.1 Function h

- **Description**
  The math formula for getting the maximum likelihood for the hipothesis.
- **Math**
  If $\theta_3 = (1-\sqrt{\theta_1})^2$ and $\theta_1+\theta_2+\theta_3 = 1$, we can make $\theta_2 = 1-\theta_1-(1-\sqrt{\theta_1})^2$. This way, we calculate the maximum for a single variable function, which is easier.
- **Usage**
  h(t, xzes)
- **Arguments**
  - t: the single variable to find the maximum
  - xzes: the vector of xzes

3.2 Function s

- **Description**
  The original likelihood function, and the stationary function for the MCMC.
- **Math**
  $s(\theta_1, \theta_2, \theta_3 \mid x_1, x_2, x_3) \propto \theta_1^{x_1}\theta_2^{x_2}\theta_3^{x_3}$
- **Usage**
  s(theta, xzes)
- **Arguments**
  - theta: the vector of thetas
  - xzes: the vector of xzes

3.3 Function p

- **Description**
  The proposal distribution for the kernel of our MCMC.
- **Math**
  A bivariate normal distribution.
- **Usage**
  p(previous, sigma)
- **Arguments**
  - previous: the expectance

  - sigma: the covariance matrix

- **Details**
  The new sample is only accepted if it respects the restrictions:

  1. $0 \leq \theta_1, \theta_2 \leq 1$
  2. $\theta_1 + \theta_2 \leq 1$

  Having a pair, we can get the third element of the sample:

  3. $\theta_3 = 1 - \theta_1 - \theta_2$

3.4 Function *step*

- **Description**
  This is the function which implements the *Metropolis Algorithm*. The details were given in the last report.

- **Math**
  The basis for this Algorithm are the Markov Chains and the Ergodic Theory.

- **Usage**
  step(previous, s, xzes, p, sigma, mode = "barker")

- **Arguments**

  - previous: the previous vector given by the Markov Chain
  - s: the potential stationary distribution function
  - xzes: the parameters for the function s
  - p: the proposal distribution function
  - sigma: the covariance matrix for the function p
  - mode: the algorithm mode. Default "barker"

- **Details**
  Differently from the original function, this one returns, together with the vector given by the Markov Chain, the value of the s function for that vector. This value is necessary when computing the e-value, and storing it on memory saves the work to calculate it again.

3.5 Function *run*

- **Description**
  This function produces a Markov Chain with a minimum autocorrelation. It receives a x_0 as input and generates the sequence applying iteractively the *step* function. To reduce the autocorrelation, it throws out a defined number of steps for each iteration.

- **Math**
  The basis for this Algorithm are the Markov Chains and the Ergodic Theory. The details are beyond the scope of this report.

- **Usage**
  run(theta, s, xzes, p, sigma, nsteps)

- **Arguments**
  - theta: the initial vector
  - s: the potential stationary distribution function
  - xzes: the parameters for the function s
  - p: the proposal distribution function
  - sigma: the covariance matrix for the function p
  - nsteps: the number of steps (samples)
  - burnin: the number of steps to throw away for each iteration. Default 4.

- **Details**
  It's important to note that the autocorrelation reduction comes with a cost of performance, once the number of steps is multiplied and few of them are used. On the other hand, this reduces the variance of the integration results. To define a good number to "burnin", we produced manually some samples, on which we applied the R function "acf", and choose a value considering both performance and precision. The diagonal covariance matrix also was tuned producing some samples manually and computing it's autocorrelation, but for the matrix there is no performance trade-off.

3.6 Function e_value

- **Description**
  This function produces one estimate for the e-value, given the parameters to evaluate. To do this, it uses MCMC to compute the integral.

- **Math**
  The estimation of the density to the interval is given by the proportion of samples in the interval by the total of samples. The basis for this is:

$$P(X_n = x | X_0 = x_0) \xrightarrow[n \to \infty]{} S(x)$$

- **Usage**
  e_value(s, xzes, p, sigma, limit, nsteps = 1000)

- **Arguments**

- s: the potential stationary distribution function
- xzes: the parameters for the function s
- p: the proposal distribution function
- sigma: the covariance matrix for the function p
- limit: the maximum value of the hypothesis(function h)
- nsteps: the number of steps (samples). Defaut 1000.

- **Details**
  We see as important to explain how the function knows how to get only the steps which satisfies the definitions of the FBST. Once that is hard to define the wanted interval for the sampled thetas, we instead used the image of the function. That's why we stored the values computed when producing the steps. The simple solution comes from getting the sampled thetas that results in a value greater than the maximum value of the hypothesis.

3.7 Function *e_value_estimate*

- **Description**
  This function returns the mean and variance of a series of estimates obtained through the function e_value. The number of estimates is passed to this function as an argument.

- **Math**
  The basis to this function is the Central Limit Theorem.

- **Usage**
  e_value_estimate(s, xzes, p, sigma, limit, nruns = 8000)

- **Arguments**

  - s: the potential stationary distribution function
  - xzes: the parameters for the function s
  - p: the proposal distribution function
  - sigma: the covariance matrix for the function p
  - limit: the maximum value of the hypothesis(function h)
  - nruns: number of estimates. Default = 8000

3.8 Function *e_value_precision*

- **Description**
  The problem of defining the the best number of iterations was solved with a R function that recursively executes the *e_value_estimate* function, until it reaches a mathematical precision criteria given by the arguments.

To do this, is implemented a loop with a stopping criteria that depends on an error tax. This error tax is calculated empirically, with the data produced by the function itself. The returned data, besides the resulting integral, shows the efficiency of the applied method through the number of iterations, the variance reduction and the estimated error tax:

  - n : the minimum necessary number of iterations for a given error tolerance
  - variance : the variance achieved
  - Ev : the e-value calculated
  - error : the estimated error tax

- **Math**
  The basis to this function is the Central Limit Theorem.

- **Usage**
  e_value_precision(s, xzes, p, sigma, limit, CI = 0.95, errtolerance = 0.01)

- **Arguments**

  - s: the potential stationary distribution function
  - xzes: the parameters for the function s
  - p: the proposal distribution function
  - sigma: the covariance matrix for the function p
  - limit: the maximum value of the hypothesis(function h)
  - CI: the confidence interval used to define the error. Default = 0.95
  - errtolerance: the wanted precision. Default = 0.01

- **Details**
  The error convergence speed is inverselly proportional to the magnitude of the e-value. Smaller numbers need smaller errors. There is an implementation of a progress bar which shows the evolution of the precision, and the n is incremented in a thousand on each iteration.