Comparing Two Population Means: Bayesian

2021-08-20

Table of Contents

# Introduction

This tutorial will focus on the use of Bayesian (MCMC sampling) estimation to explore differences between two populations. Bayesian estimation is supported within R via two main packages (MCMCpack and MCMCglmm). These packages provide relatively simple R-like interfaces (particularly MCMCpack) to Bayesian routines and are therefore reasonably good starting points for those transitioning between traditional and Bayesian approaches.

BUGS (*Bayesian inference Using Gibbs Sampling*) is an algorithm and supporting language (resembling R) dedicated to performing the Gibbs sampling implementation of Markov Chain Monte Carlo method. Dialects of the BUGS language are implemented within two main projects;

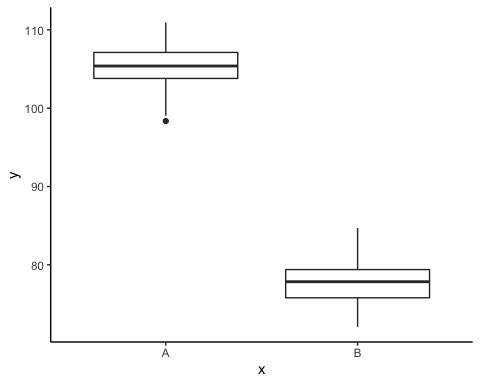
* WinBUGS/OpenBUGS - written in component pascal and therefore originally Windows only. A less platform specific version (OpenBUGS - Windows and Linux via some component pascal libraries) is now being developed as the WinBUGS successor. Unlike WinBUGS, OpenBUGS does not have a Graphical User Interface and was designed to be invoked from other applications (such as R). The major drawback of WinBUGS/OpenBUGS is that it is relatively slow.
* JAGS (Just Another Gibbs Sampler) - written in C++ and is therefore cross-platform and very fast. It can also be called from within R via various packages.

library(tidyverse)  
library(coda)  
library(broom)  
library(bayesplot)  
# Approach specific packages  
library(MCMCpack)  
library(R2jags)  
library(rstan)  
library(rstanarm)  
library(brms)

# Assign starter values  
set.seed(1)  
nA <- 60 # sample size from Population A  
nB <- 40 # sample size from Population B  
muA <- 105 # population mean of Population A  
muB <- 77.5 # population mean of Population B  
sigma <- 3 # standard deviation of both populations (equally varied)  
  
# Data generation  
yA <- rnorm(n = nA, mean = muA, sd = sigma) # Population A sample  
yB <- rnorm(n = nB, mean = muB, sd = sigma) #Population B sample  
y <- c(yA, yB)  
x <- factor(rep(c("A", "B"), c(nA, nB))) #categorical listing of the populations  
xn <- as.numeric(x) #numerical version of the population category for means parameterization.   
# Should not start at 0.  
data <- data.frame(y, x, xn) # dataset  
head(data)

## y x xn  
## 1 103.1206 A 1  
## 2 105.5509 A 1  
## 3 102.4931 A 1  
## 4 109.7858 A 1  
## 5 105.9885 A 1  
## 6 102.5386 A 1

ggplot(data = data, aes(x = x, y = y)) +   
 geom\_boxplot() +  
 theme\_classic()



# Model fitting

A t-test is essentially just a simple regression model in which the categorical predictor is represented by a binary variable in which one level is coded as 0 and the other 1.

For the model itself, the observed response () are assumed to be drawn from a normal distribution with a given mean () and standard deviation (). The expected values () are themselves determined by the linear predictor (). In this case, β0 represents the mean of the first treatment group and β represents the difference between the mean of the first group and the mean of the second group (the effect).

MCMC sampling requires priors on all parameters. We will employ weakly informative priors. Specifying ‘uninformative’ priors is always a bit of a balancing act. If the priors are too vague (wide) the MCMC sampler can wander off into nonscence areas of likelihood rather than concentrate around areas of highest likelihood (desired when wanting the outcomes to be largely driven by the data). On the other hand, if the priors are too strong, they may have an influence on the parameters. In such a simple model, this balance is very forgiving - it is for more complex models that prior choice becomes more important.

For this simple model, we will go with zero-centered Gaussian (normal) priors with relatively large standard deviations (1000) for both the intercept and the treatment effect and a wide half-cauchy (scale=25) for the standard deviation.