Session 3

Multi-population Hui-Walter models

Matt Denwood

2021-06-29

Session 3: Multi-population Hui-Walter models

Recap

- Fitting models using MCMC is easy with JAGS / runjags
- But we must **never forget** to check convergence and effective sample size!
- More complex models become easy to implement
  - For example imperfect diagnostic tests, and Hui-Walter models
  - But remember to be realistic about what is possible with your data
  - Also carefully consider the influence of your priors

Hui-Walter models with multiple populations

- Basically an extension of the single-population model
- Works best with multiple populations each with differing prevalences
    - Including an unexposed population works well
    - BUT be wary of assumptions regarding constant sensitivity/specificity across populations with very different types of infections

Independent intercepts for populations

```
model{
  for(p in 1:Populations){
    Tally[1:4, p] ~ dmulti(prob[1:4, p], TotalTests[p])
    # Test1- Test2- Pop1
      prob[1, p] <- (prev[p] * ((1-se[1])*(1-se[2]))) + ((1-prev[p]) *
↪  ((sp[1])*(sp[2])))
    ## etc ##
    prev[p] ~ dbeta(1, 1)
  }

  se[1] ~ dbeta(HPSe[1,1], HPSe[1,2])T(1-sp[1], )
  sp[1] ~ dbeta(HPSp[1,1], HPSp[1,2])
  se[2] ~ dbeta(HPSe[2,1], HPSe[2,2])T(1-sp[2], )
  sp[2] ~ dbeta(HPSp[2,1], HPSp[2,2])

  #data# Tally, TotalTests, Populations, HPSe, HPSp
  #monitor# prev, prob, se, sp
  #inits# prev, se, sp
}
```

Random intercepts for a larger number of populations

```
model{
  for(p in 1:Populations){
    Tally[1:4, p] ~ dmulti(prob[1:4, p], TotalTests[p])
    # Test1- Test2- Pop1
      prob[1, p] <- (prev[p] * ((1-se[1])*(1-se[2]))) + ((1-prev[p]) *
↪  ((sp[1])*(sp[2])))
    ## etc ##

      logit(prev[p]) <- intercept + population_effect[p]
      population_effect[p] ~ dnorm(0, tau)
  }

  tau ~ dgamma(0.01, 0.01)
  intercept ~ dnorm(0, 0.33)

  se[1] ~ dbeta(HPSe[1,1], HPSe[1,2])T(1-sp[1], )
  sp[1] ~ dbeta(HPSp[1,1], HPSp[1,2])
  se[2] ~ dbeta(HPSe[2,1], HPSe[2,2])T(1-sp[2], )
  sp[2] ~ dbeta(HPSp[2,1], HPSp[2,2])

  #data# Tally, TotalTests, Populations, HPSe, HPSp
  #monitor# prev, prob, se, sp
  #inits# prev, se, sp
}
```

Multiple populations: assumptions

- We typically assume that the sensitivity and specificity *must* be consistent between populations
    - Do you have an endemic and epidemic population? Or vaccinated and unvaccinated?
- It helps if the prevalence differs between the populations
- The populations can be artificial (e.g. age groups) but must not be decided based on the diagnostic test results

Multiple populations: special cases

- A small disease-free group is extremely helpful
  - Contains strong data regarding specificity
  - As long as specificity can be assumed to be the same in the other populations
- A small experimentally infected group MAY be helpful but it is sometimes difficult to assume that sensitivity is consistent

Incorporating additional data

TODO model formulation with special groups of known disease status (Or is it easier just to specify the prevalence as 0 or 1 in data??)

Other runjags options

There are a large number of other options to runjags. Some highlights:

- The method can be parallel or background or bgparallel
- You can use extend.jags to continue running an existing model (e.g. to increase the sample size)
- You can use coda::as.mcmc.list to extract the underlying MCMC chains
- Use the summary() method to extract summary statistics
  - See `?summary.runjags` and `?runjagsclass` for more information

Using embedded character strings

- For simple models we might not want to bother with an external text file. Then we can do:

```
mt <- "
model{
  Positives ~ dbinom(prevalence, TotalTests)
  prevalence ~ dbeta(2, 2)

  #data# Positives, TotalTests
  #monitor# prevalence
  #inits# prevalence
}
"
Positives <- 7
TotalTests <- 10
prevalence <- list(chain1=0.01, chain2=0.99)
results <- run.jags(mt, n.chains=2)
## Loading required namespace: rjags
```

- But I would advise that you stick to using a separate text file!

Setting the RNG seed

- If we want to get numerically replicable results we need to add `.RNG.name` and `.RNG.seed` to the initial values, and an additional `#modules#` lecuyer hook to our basicjags.bug file:

```
model{
  Positives ~ dbinom(prevalence, TotalTests)
  prevalence ~ dbeta(2, 2)

  #data# Positives, TotalTests
  #monitor# prevalence
  #inits# prevalence, .RNG.name, .RNG.seed
  #modules# lecuyer
}
```

```
.RNG.name <- "lecuyer::RngStream"
.RNG.seed <- list(chain1=1, chain2=2)
results <- run.jags('basicjags.bug', n.chains=2)
```

- Every time this model is run the results will now be identical

Practical session 3

Points to consider

1. What are the benefits of including multiple populations?
2. How can we define/obtain these populations?
3. What happens if our fundamental assumptions about consistent Se/Sp are broken?

Summary

- Multiple populations helps to estimate Se and Sp
  - Particularly if the prevalences differ
  - A large number of populations with small N may be better as a random effect
- Populations may be artificial
  - But cannot be based on the result of either test
- But if Se / Sp are inconsistent then we will get misleading results
  - In practice, groups with widely varying prevalence rarely have consistent Se / Sp
  - It is possible to allow Se / Sp to differ between populations, but then there is no benefit of combining the data