Session 3

Multi-population Hui-Walter models

Matt Denwood

2021-06-29

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

Multi-population Hui-Walter models

Hui-Walter models with multiple populations

- Basically an extension of the single-population model
- Works best with multiple populations each with differing prevalence
  - 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])))
    ## snip ##

    prev[p] ~ dbeta(1, 1)
  }

  se[1] ~ dbeta(se_prior[1,1], se_prior[1,2])T(1-sp[1], )
  sp[1] ~ dbeta(sp_prior[1,1], sp_prior[1,2])
  se[2] ~ dbeta(se_prior[2,1], se_prior[2,2])T(1-sp[2], )
  sp[2] ~ dbeta(sp_prior[2,1], sp_prior[2,2])

  #data# Tally, TotalTests, Populations, se_prior, sp_prior
  #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])
    ## snip ##

      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(se_prior[1,1], se_prior[1,2])T(1-sp[1], )
  sp[1] ~ dbeta(sp_prior[1,1], sp_prior[1,2])
  se[2] ~ dbeta(se_prior[2,1], se_prior[2,2])T(1-sp[2], )
  sp[2] ~ dbeta(sp_prior[2,1], sp_prior[2,2])

  #data# Tally, TotalTests, Populations, se_prior, sp_prior
  #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?
  - If so then the assumptions might not hold!

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?
  - If so then the assumptions might not hold!
- The populations can be artificial (e.g. age groups) but must not be decided based on the diagnostic test results
  - It helps if the prevalence differs between the populations

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

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 often dangerous to assume that sensitivity is consistent!

Incorporating populations with known prevalence

Up to now prevalence has been a parameter, but it can also be (partially) observed:

```
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])))
    ## snip ##

    prev[p] ~ dbeta(1, 1)
  }

  ## snip ##

  #data# Tally, TotalTests, Populations, se_prior, sp_prior, prev
  #monitor# prev, prob, se, sp
  #inits# prev, se, sp
}
```

To fix the prevalence of population 1 we could do:

```
Populations <- 5
prev <- rep(NA, Populations)
prev[1] <- 0
prev
## [1]  0 NA NA NA NA
```

To fix the prevalence of population 1 we could do:

```
Populations <- 5
prev <- rep(NA, Populations)
prev[1] <- 0
prev
## [1]  0 NA NA NA NA
```

But you also need to account for this in the initial values:

```
prev <- list(chain1=c(NA, 0.2, 0.4, 0.6, 0.8), chain2=c(NA, 0.8, 0.6,
↪  0.4, 0.2))
```

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:

```
model_string <- "
model{
  Positives ~ dbinom(prevalence, TotalTests)
  prevalence ~ dbeta(1, 1)

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

- 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(model_string, 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