Session 7

Incorporating imperfect sensitivity and specificity into more complex models

Matt Denwood

2022-06-10

Recap

NOTE: THIS MATERIAL IS NOT YET FINALISED, PLEASE CHECK BACK SOON!

Models for diagnostic test evaluation require:

- At least 2 tests
- At least 2 populations, but preferably 3 or more
- Quite a lot of data

Recap

NOTE: THIS MATERIAL IS NOT YET FINALISED, PLEASE CHECK BACK SOON!

Models for diagnostic test evaluation require:

- At least 2 tests
- At least 2 populations, but preferably 3 or more
- Quite a lot of data

Fitting the models is technically quite straightforward

The real difficulty lies in the interpretation

- What exactly is the latent class?

Incorporating coefficients: prevalence

Modelling variation in infection probability

- Individuals may be at higher/lower risk of being infected due to known characteristics e.g.:
  - Age
  - Sex
  - History
  - Presence of co-infections
  - Whatever

Modelling variation in infection probability

- Individuals may be at higher/lower risk of being infected due to known characteristics e.g.:
  - Age
  - Sex
  - History
  - Presence of co-infections
  - Whatever
- There are three ways to deal with this:
  1. Ignore it
  2. Group “populations” by these characteristics
  3. Embed a (preferably simple!) generalised linear model within your LCM

Logistic regression in JAGS

```
model{

  for(i in 1:N){
    Observation[i] ~ dbern(prob[i])
    logit(prob[i]) <- intercept + beta1[Category[i]] + beta2*Covariate[i]
  }

  intercept ~ dnorm(0, 0.01)
  beta1[1] <- 0
  for(c in 2:NC){
    beta1[c] ~ dnorm(0, 0.01)
  }
  beta2 ~ dnorm(0, 0.01)

  #data# N, Observation, NC, Category, Covariate
  #monitor# intercept, beta1, beta2
  #inits# intercept, beta1, beta2
}
```

```
model{

  for(i in 1:N){
    Observation[i] ~ dbern(obs_prob[i])
    obs_prob[i] <- prob[i]*se + (1-prob[i])*(1-sp)
    logit(prob[i]) <- intercept + beta1[Category[i]] + beta2*Covariate[i]
  }

  se ~ dbeta(148.43, 16.49)T(1-sp, )
  sp ~ dbeta(240.03, 12.63)

  intercept ~ dnorm(0, 0.01)
  beta1[1] <- 0
  for(c in 2:NC){
    beta1[c] ~ dnorm(0, 0.01)
  }
  beta2 ~ dnorm(0, 0.01)

  #data# N, Observation, NC, Category, Covariate
  #monitor# intercept, beta1, beta2, se, sp
  #inits# intercept, beta1, beta2, se, sp
}
```

```
model{

  for(i in 1:N){
    Observation[i] ~ dbern(obs_prob[i])
    obs_prob[i] <- prob[i]*se + (1-prob[i])*(1-sp)
    logit(prob[i]) <- intercept + beta1[Category[i]] + beta2*Covariate[i]
  }

  #data# se, sp

  intercept ~ dnorm(0, 0.01)
  beta1[1] <- 0
  for(c in 2:NC){
    beta1[c] ~ dnorm(0, 0.01)
  }
  beta2 ~ dnorm(0, 0.01)

  #data# N, Observation, NC, Category, Covariate
  #monitor# intercept, beta1, beta2
  #inits# intercept, beta1, beta2
}
```

```
model{

  for(i in 1:N){
    Observation[i] ~ dbern(obs_prob[i])
    obs_prob[i] <- prob[i]*se[Test[i]] + (1-prob[i])*(1-sp[Test[i]])
    logit(prob[i]) <- intercept + beta1[Category[i]] + beta2*Covariate[i]
  }

  #data# se, sp

  intercept ~ dnorm(0, 0.01)
  beta1[1] <- 0
  for(c in 2:NC){
    beta1[c] ~ dnorm(0, 0.01)
  }
  beta2 ~ dnorm(0, 0.01)

  #data# N, Observation, NC, Category, Covariate, Test
  #monitor# intercept, beta1, beta2
  #inits# intercept, beta1, beta2
}
```

```
model{

  for(i in 1:N){
    Observations[i,1:4] ~ dmulti(obs_probs[i,1:4], 1)

    obs_probs[i,1] <- (prob[i] * ((1-se[1])*(1-se[2]))) + ((1-prob[i]) * ((sp[1])*(sp[2])))
    obs_probs[i,2] <- (prob[i] * ((se[1])*(1-se[2]))) + ((1-prob[i]) * ((1-sp[1])*(sp[2])))
    obs_probs[i,3] <- (prob[i] * ((1-se[1])*(se[2]))) + ((1-prob[i]) * ((sp[1])*(1-sp[2])))
    obs_probs[i,4] <- (prob[i] * ((se[1])*(se[2]))) + ((1-prob[i]) * ((1-sp[1])*(1-sp[2])))

    logit(prob[i]) <- intercept + beta1[Category[i]] + beta2*Covariate[i]
  }

  #snip#

}
```

```
model{

  for(i in 1:G){
    Observations[i,1:4] ~ dmulti(obs_probs[i,1:4], Total[i])

    obs_probs[i,1] <- (prob[i] * ((1-se[1])*(1-se[2]))) + ((1-prob[i]) * ((sp[1])*(sp[2])))
    obs_probs[i,2] <- (prob[i] * ((se[1])*(1-se[2]))) + ((1-prob[i]) * ((1-sp[1])*(sp[2])))
    obs_probs[i,3] <- (prob[i] * ((1-se[1])*(se[2]))) + ((1-prob[i]) * ((sp[1])*(1-sp[2])))
    obs_probs[i,4] <- (prob[i] * ((se[1])*(se[2]))) + ((1-prob[i]) * ((1-sp[1])*(1-sp[2])))

    logit(prob[i]) <- intercept + beta1[Category[i]] + beta2*RoundedCovariate[i]
  }

  #snip#

}
```

Embedding a LR within a LCM

- Blocking at group level is much more efficient than looping through all individuals
- Autocorrelation may be problematic - if so try to use different contrast schemes eg:

```
sex_effect ~ dnorm(0, 0.01)
beta1[1] <- -sex_effect/2
beta1[2] <- sex_effect/2
```

- Random effects are kind of like fixed effects:

```
#snip#
  logit(prob[i]) <- intercept + beta1[Category[i]] + beta3[Group[i]]
#snip#

for(r in 1:NR){
  beta3[r] ~ dnorm(0, tau)
}
tau ~ dgamma(0.01, 0.01)

#inits# tau
#monitor# tau, beta3
```

Generating code for a LR

You can use template.jags as inspiration:

```
template.jags(weight ~ group, family="gaussian", data=data, file="linear_model.txt")
## Your model template was created at "linear_model.txt" - it is highly advisable to examine
↪  the model syntax to be sure it is as intended
## You can then run the model using run.jags("linear_model.txt")
results <- run.jags("linear_model.txt")
## Loading required namespace: rjags
## module glm loaded
## module dic loaded
```

Supported features:

- Gaussian, binomial, Poisson, negative binomial, ZIB, ZIP, ZINB
- Random intercepts

We can also add (currently manually):

- Random slopes
- Spline terms
- Interval censoring

Grouping populations

- This is the easier option as we can use template_huiwalter!
    - See Otero-Abad 2017 for a simple example
- If you have a lot of populations you could use a simple random effect:

```
# prev[p] ~ dbeta(1, 1)
logit(prev[p]) <- intercept + raneff[i]
raneff[i] ~ dnorm(0, tau)
```

Grouping populations

- This is the easier option as we can use template_huiwalter!
  - See Otero-Abad 2017 for a simple example
- If you have a lot of populations you could use a simple random effect:

```
# prev[p] ~ dbeta(1, 1)
logit(prev[p]) <- intercept + raneff[i]
raneff[i] ~ dnorm(0, tau)
```

- Be careful that Se/Sp is still consistent across populations!

Do nothing?

What is the goal of your analysis?

- Estimating risk factors for disease?
- Estimating true prevalence?
- Estimating Se/Sp?

Do nothing?

What is the goal of your analysis?

- Estimating risk factors for disease?
- Estimating true prevalence?
- Estimating Se/Sp?

Inclusion of risk factors for disease is NOT necessary to estimate Se/Sp!

Incorporating coefficients: sensitivity / specificity

What if diagnostic tests are not consistent across populations?

This time we can't just ignore it!

Solutions:

- Remove that population (and clearly state this in the paper..!)
- Allow the relevant parameter to vary between populations
- Use a very simple GLM on the relevant parameter(s)

Varying between populations

Covid paper

Embedded GLM

Be careful with centering and contrasts

Martinez paper

General points

If you are interested in covariates on prevalence (rather than the se/sp directly) then use a different approach

Inconsistent Se/Sp may happen in e.g. laboratory vs field settings, different sample types, etc

Theoretically it is possible to incorporate this into the model, but if all populations have their own se/sp then the model collapses!

Be VERY careful when prevalence and se/sp have the same covariate

- Probably best to balance populations by these covariates and then only include them as se/sp covariates?

Practical session 7

Points to consider

1. What is the optimal number of populations?
2. What happens to identifiability when you deviate “too far” from the standard Hui-Walter model?

Summary

- Adding populations (or equivalently, covariates on prevalence) adds parameters but may add information
    - But it is not always worthwile!
- Using covariates on sensitivity and specificity is tricky. . .
- Some further reading: Martinez et al, Stærk-Østergaard et al.