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

Grouping populations

Be careful that Se/Sp still consistent

Random effects - code example

Otero-Abad paper

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(1,1)T(1-sp, )
  sp ~ dbeta(1,1)

  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]
  }

  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]
  }

  se <- 0.9
  sp <- 0.95

  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]
  }

  #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
}
```

Group vs Individual LR

Blocking at group level more efficient

Individual level is possible but not advisable

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
```

```
results
##
## JAGS model summary statistics from 20000 samples (chains = 2; adapt+burnin = 5000):
##
##                       Lower95   Median Upper95    Mean
## regression_precision 0.84305   1.9806  3.4163  2.0547
## intercept             4.5824   5.0308  5.5074  5.0311
## group_effect[1]            0        0       0       0
## group_effect[2]      -1.0008 -0.37192 0.30876 -0.3729
## deviance              40.182   42.729  48.604  43.424
## resid.sum.sq          8.7293   9.4284   12.23  9.8307
##
##                           SD    Mode     MCerr MC%ofSD
## regression_precision 0.68605   1.901 0.0053316     0.8
## intercept             0.2346  5.0333 0.0016551     0.7
## group_effect[1]            0       0        --      --
## group_effect[2]      0.33002 -0.3658 0.0023264     0.7
## deviance              2.6643  41.677  0.022246     0.8
## resid.sum.sq          1.2633  9.0354   0.01016     0.8
##
##                       SSeff     AC.10    psrf
## regression_precision 16557 0.0026784  1.0004
## intercept            20091 0.0091337 0.99998
## group_effect[1]         --        --      --
## group_effect[2]      20124 0.0061385 0.99995
## deviance             14343    0.0128  1.0001
```

Supported features:

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

We can also add (currently manually):

- Random slopes
- Spline terms
- Interval censoring

Example

Modify the code to add a single fixed effect across e.g. 3 populations

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.