Session 7

Incorporating imperfect sensitivity and specificity into more complex models

Matt Denwood

2021-07-01

Recap

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

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 imperfect sensitivity and specificity into more complex models

Logistic regression in JAGS

```
model{

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

  intercept ~ dnorm(0, 10^-6)
  beta1 ~ dnorm(0, 10^-6)
  beta2 ~ dnorm(0, 10^-6)

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

```
model{

  for(i in 1:N){
    Outcome[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, 10^-6)
  beta1 ~ dnorm(0, 10^-6)
  beta2 ~ dnorm(0, 10^-6)

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

```
model{

  for(i in 1:N){
    Outcome[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, 10^-6)
  beta1 ~ dnorm(0, 10^-6)
  beta2 ~ dnorm(0, 10^-6)

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

```
model{

  for(i in 1:N){
    Outcome[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]
  }

  se[1] ~ dbeta(148.43, 16.49)T(1-sp[1], )
  sp[1] ~ dbeta(240.03, 12.63)
  se[2] ~ dbeta(183.59, 9.98)T(1-sp[2], )
  sp[2] ~ dbeta(199.22, 0.53)

  intercept ~ dnorm(0, 10^-6)
  beta1 ~ dnorm(0, 10^-6)
  beta2 ~ dnorm(0, 10^-6)

  #data# N, Outcome, Category, Covariate, Test
  #monitor# intercept, beta1, beta2, se, sp
  #inits# intercept, beta1, beta2, se, sp
}
```

Other types of GL(M)M

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 0.82427   1.9869   3.4119   2.0603
## intercept             4.5606   5.0316   5.4943   5.0307
## group_effect[1]            0        0        0        0
## group_effect[2]      -1.0258 -0.36828 0.28205 -0.36871
## deviance              40.181   42.727   48.687   43.436
## resid.sum.sq          8.7293   9.4175   12.264   9.8313
##
##                           SD Mode      MCerr MC%ofSD SSeff
## regression_precision 0.68727   -- 0.0053809     0.8 16313
## intercept            0.23498   -- 0.0016616     0.7 20000
## group_effect[1]            0    0        --      --    --
## group_effect[2]      0.33082   -- 0.0022963     0.7 20756
## deviance              2.6983   --  0.021922     0.8 15150
## resid.sum.sq          1.2877   --  0.010336     0.8 15522
##
##                           AC.10    psrf
## regression_precision  0.0082247  1.0001
## intercept             0.0063624  1.0001
## group_effect[1]              --      --
## group_effect[2]      -0.0071836 0.99998
## deviance             -0.0069055  1.0003
```

Supported features:

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

We can also add (currently manually):

- Random slopes
- Spline terms
- Interval censoring

What about other models?

MCMC is highly flexible!

What about other models?

MCMC is highly flexible!

We can have:

- Hidden Markov models
- State Space models
- Other types of latent class model

What about other models?

MCMC is highly flexible!

We can have:

- Hidden Markov models
- State Space models
- Other types of latent class model

But does your data match your ambitions?

- All models can be specified
- Relatively few are identifiable

Before you go...

- Feedback on the course would be extremely welcome!
    - I will send an email later today with a survey link

Before you go...

- Feedback on the course would be extremely welcome!
  - I will send an email later today with a survey link
- Remember to keep an eye on the COST action website:
  - http://harmony-net.eu
  - Physical training schools are being run in September and accepting sign-ups now!

Practical session 7

Points to consider

1. When is there a benefit of adding imperfect test information?
2. When is there no real benefit?