Hands-on training session 3

Hui-Walter models with more than two diagnostic tests

Matt Denwood Giles Innocent Sonja Hartnack

2020-02-19

Introduction

Overview

Date/time:

- 20th February 2020
- 14.00 - 15.30

Teachers:

- Matt Denwood (presenter)
- Giles Innocent
- Sonja Hartnack

Recap

- JAGS / runjags is the wasy way to work with complex models
  - But we *still have to* check convergence and effective sample size!
- Estimating sensitivity and specificity is like pulling a rabbit out of a hat
  - Multiple populations helps **a lot**
  - Strong priors for one of the tests helps even more!

Recap

- JAGS / runjags is the wasy way to work with complex models
  - But we *still have to* check convergence and effective sample size!
- Estimating sensitivity and specificity is like pulling a rabbit out of a hat
  - Multiple populations helps **a lot**
  - Strong priors for one of the tests helps even more!
- But what if the tests are not independent of each other?

Session 3a: Hui-Walter models for multiple tests with conditional indepdendence

What exactly is our latent class?

- What do we mean by “conditionally independent?”

What exactly is our latent class?

- What do we mean by "conditionally independent?"
- Example: we have three antibody tests
  - The latent status is actually 'producing antibodies' not 'diseased'

What exactly is our latent class?

- What do we mean by "conditionally independent?"
- Example: we have three antibody tests
  - The latent status is actually 'producing antibodies' not 'diseased'
- Example: antibody vs egg count tests for liver fluke
  - Does the latent state include migrating juvenile fluke?

What exactly is our latent class?

- What do we mean by "conditionally independent?"
- Example: we have three antibody tests
  - The latent status is actually 'producing antibodies' not 'diseased'
- Example: antibody vs egg count tests for liver fluke
  - Does the latent state include migrating juvenile fluke?
- We're actually pulling **something** out of a hat, and deciding to call it a rabbit

Simulating data

Simulating data using an arbitrary number of independent tests is quite straightforward.

```r
# Parameter values to simulate:
N <- 200
se1 <- 0.8
sp1 <- 0.95
se2 <- 0.9
sp2 <- 0.99
se3 <- 0.95
sp3 <- 0.95

Populations <- 2
prevalence <- c(0.25,0.75)
Group <- sample(1:Populations, N, replace=TRUE)
```

```r
# Ensure replicable data:
set.seed(2020-02-18)

# Simulate the true latent state (which is unobserved in real life):
true <- rbinom(N, 1, prevalence[Group])
# Simulate test results for test 1:
test1 <- rbinom(N, 1, se1*true + (1-sp1)*(1-true))
# Simulate test results for test 2:
test2 <- rbinom(N, 1, se2*true + (1-sp2)*(1-true))
# Simulate test results for test 3:
test3 <- rbinom(N, 1, se3*true + (1-sp3)*(1-true))

simdata <- data.frame(Population=factor(Group), Test1=test1, Test2=test2, Test3=test3)
```

Model specification

- Like for two tests, except it is now a 2x2x2 table
  - If calculating this manually, take **extreme** care with multinomial tabulation
- Or use autohuiwalter
  - This will also deal gracefully with missing data in one or more test results

```
source("autohuiwalter.R")
auto_huiwalter(simdata[,c('Population','Test1','Test2','Test3')],
↪  outfile='auto3thw.bug')
```

```
    for(p in 1:Populations){
        Tally_RRR[1:8,p] ~ dmulti(prob_RRR[1:8,p], N_RRR[p])
        prob_RRR[1:8,p] <- se_prob[1:8,p] + sp_prob[1:8,p]
    }

    . . .

    for(p in 1:Populations){
        # Probability of observing Test1- Test2- Test3- from a
        ↪  true positive::
        se_prob[1,p] <- prev[p] * ((1-se[1])*(1-se[2])*(1-se[3])
        ↪  +covse12 +covse13 +covse23)
        # Probability of observing Test1- Test2- Test3- from a
        ↪  true negative::
        sp_prob[1,p] <- (1-prev[p]) * (sp[1]*sp[2]*sp[3] +covsp12
        ↪  +covsp13 +covsp23)

        . . .

        # Probability of observing Test1+ Test2+ Test3+ from a
        ↪  true positive::
        se_prob[8,p] <- prev[p] * (se[1]*se[2]*se[3] +covse12
        ↪  +covse13 +covse23)
```

Alternative model specification

We might want to explicitly model the latent state:

```
model{

  for(i in 1:N){
    truestatus[i] ~ dbern(prev[Population[i]])

    Status[i] ~ dcat(prob[i, ])
    prob[1:8,i] <- se_prob[1:8,i] + sp_prob[1:8,i]
```

```
1
2       se_prob[1,p] <- truestatus[i] *
        ↪  ((1-se[1])*(1-se[2])*(1-se[3]))
3           sp_prob[1,p] <- (1-truestatus[i]) * (sp[1]*sp[2]*sp[3])
4
5           se_prob[2,p] <- truestatus[i] *
            ↪  (se[1]*(1-se[2])*(1-se[3])
6           sp_prob[2,p] <- (1-truestatus[i]) *
            ↪  ((1-sp[1])*sp[2]*sp[3])
7
8           se_prob[3,p] <- truestatus[i] *
            ↪  ((1-se[1])*se[2]*(1-se[3]))
9           sp_prob[3,p] <- (1-truestatus[i]) *
            ↪  (sp[1]*(1-sp[2])*sp[3])
10
11          se_prob[4,p] <- truestatus[i] * (se[1]*se[2]*(1-se[3]))
12          sp_prob[4,p] <- (1-truestatus[i]) *
            ↪  ((1-sp[1])*(1-sp[2])*sp[3])
13          se_prob[5,p] <- truestatus[i] *
            ↪  ((1-se[1])*(1-se[2])*se[3])
14          sp_prob[5,p] <- (1-truestatus[i]) *
            ↪  (sp[1]*sp[2]*(1-sp[3]))
15
```

But this is inefficient

There is also no way to distinguish individuals within the same boxes

We could also use the estimated se/sp/prev to post-calculate these status probabilities

This is useful for post-hoc ROC

Exercise

Simulate data from 3 tests and analyse using the autohuiwalter function

Do the estimates of Se/Sp correspond to the simulation parameters?

Make some data missing for one or more tests and re-generate the model

- Can you see what has changed in the code?

Optional Exercise

Simulate data from 3 antibody tests with ab positive step [give code in solution]

Does the se/sp estimated by a model recover the parameters?

Why not?

```
library('tidyverse')
```

```
## - Attaching packages ---------- tidyverse 1.3.0 -
```

```
## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## - Conflicts ----------- tidyverse_conflicts() -
## x tidyr::extract() masks runjags::extract()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()
```

Session 3b: Hui-Walter models for multiple tests with conditional depdendence

Branching of processes leading to test results

Example: two antibody tests and one antigen test

Or three antibody tests where one has a different target to the other two

Simulating data

It helps to consider the data simulation as a biological process.

```r
# Parameter values to simulate:
N <- 200
se1 <- 0.8
se2 <- 0.9
se3 <- 0.95
sp1 <- 0.95
sp2 <- 0.99
sp3 <- 0.95

Populations <- 2
prevalence <- c(0.25,0.75)
Group <- rep(1:Populations, each=N)

# Ensure replicable data:
set.seed(2017-11-21)

# We will assume test 1 is dependent of the others, but tests 2&3
```

Verifying simulation results

Model specification

```
1           # Probability of observing ELISA1- ELISA2- WesternBlot-
            ↪  from a true positive::
2           se_prob[1,p] <- prev[p] * ((1-se[1])*(1-se[2])*(1-se[3])
    ↪   +covse12 +covse13 +covse23)
3           # Probability of observing ELISA1- ELISA2- WesternBlot-
            ↪  from a true negative::
4           sp_prob[1,p] <- (1-prev[p]) * (sp[1]*sp[2]*sp[3] +covsp12
    ↪   +covsp13 +covsp23)
5
6           # Probability of observing ELISA1+ ELISA2- WesternBlot-
            ↪  from a true positive::
7           se_prob[2,p] <- prev[p] * (se[1]*(1-se[2])*(1-se[3])
    ↪   -covse12 -covse13 +covse23)
8           # Probability of observing ELISA1+ ELISA2- WesternBlot-
            ↪  from a true negative::
9           sp_prob[2,p] <- (1-prev[p]) * ((1-sp[1])*sp[2]*sp[3]
    ↪   -covsp12 -covsp13 +covsp23)
10
11          ...
12
```

Generating the model

Extreme care needed

Use autohuiwalter with argument covon=TRUE

```
source('autohuiwalter.R')
auto_huiwalter(ind3tests, 'auto3tihw.bug', covon=TRUE)
```

```
## The model and data have been written to auto3tihw.bug in the current working directory
## You should check and alter priors before running the model
```

```
## Auto-generated Hui-Walter model created by script version 0.1 on 2020-02-19

model{

    ## Observation layer:

    # Complete observations (N=400):
    for(p in 1:Populations){
        Tally.RRR[1:8,p] ~ dmulti(prob.RRR[1:8,p], N.RRR[p])
```

Exercise

Simulate data with a dependence between 2 tests

Model assuming conditional independence biases the estimates

Model with conditional depdendence has bigger CI but unbiased

Session 3c: Model selection

Motivation

[Planning for this session to be a general discussion between all instructors and students, as I am not entirely sure what to recommend in terms of model selection - except that I dislike DIC!!!]

Background to DIC

[Some theory slides stolen from ABME course: ABME_Model selection.pptx]

DIC works fine for hierarchical normal models but not others

Other methods

Bayes factors work well if you can count them

WAIC works better for a wide range of models

```
* An approximation to LOO with general applicability
* Probably won't work for Hui-Walter though due to lack of
↪  independent data
* Could be useful if using the GLM version (untested!)
```

Models tend to be sensitive to priors

Simulating data and testing that your model recovers the parameters is a good idea

Calculating DIC

Add dic and ped to the monitors in runjags

But don't trust the results

Also bear in mind you can't parallelise

Calculating WAIC

Currently a pain

```
1  ## This is an example of extracting WAIC from runjags/jags objects
2  # Matt Denwood, 2019-11-11
3  # Note that this will all get much easier with the release of JAGS 5 and the next verison of runjags!!
4
5  ## A function to return the WAIC
6  # Also returns the effective number of parameters (p_waic), elpd and lpd as described by:
7  # www.stat.columbia.edu/~gelman/research/unpublished/waic_stan.pdf
8  # Note:     mean_lik is the log of the (exponentiated) likelihoods
9  #           var_log_lik is the variance of the log likelihoods
10 #           these need separate monitors in JAGS
11 get_waic <- function(mean_lik, var_log_lik){
12
13     stopifnot(length(mean_lik)==length(var_log_lik))
```

Future Updates

Model criticism will get better in JAGS 5, and the next update of runjags

Installing development version of runjags:

```
# Put on drat server and supply code here
```

WAIC is also calculable from Stan models (easily?)

Discussion and free practical time

What would be useful to add to the autohuiwalter function?

- Modify so it allows Se/Sp priors to be defined as matrices?
- And correlations on/off as matrices?