

Hands-on training session 3

Hui-Walter models with more than two diagnostic tests

Matt Denwood Giles Innocent Sonja Hartnack

2020-02-17

Introduction

Overview

Date/time:

- 20th February 2020
- 14.00 - 15.30

Teachers:

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

Recap

Important points from sessions 1 and 2

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

What exactly is our latent class?

What do we mean by “conditionally independent?”

Example: three antibody tests

The latent status is actually ‘producing antibodies’ not ‘diseased’

We’re actually pulling **something** out of a hat, and deciding to call it a rabbit

Model specification

If doing this manually, take **extreme** care with multinomial tabulation

Or use `autohwiwalter`!

- This will also deal gracefully with missing data in one or more test results

Simulating data

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

```
1  # Parameter values to simulate:
2  N <- 200
3  se1 <- 0.8
4  se2 <- 0.9
5  se3 <- 0.95
6  sp1 <- 0.95
7  sp2 <- 0.99
8  sp3 <- 0.95
9
10 Populations <- 2
11 prevalence <- c(0.25,0.75)
12 Group <- rep(1:Populations, each=N)
13
14 # Ensure replicable data:
15 set.seed(2017-11-21)
16
17 # Simulate the true latent state (which is unobserved in real
```


Exercise

Simulate data from 3 tests and analyse using the `autohwiwalter` function

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

Optional Exercise

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

- Can you see what has changed in the code?

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

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

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
```

Simulating data

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

```
1  # Parameter values to simulate:
2  N <- 200
3  se1 <- 0.8
4  se2 <- 0.9
5  se3 <- 0.95
6  sp1 <- 0.95
7  sp2 <- 0.99
8  sp3 <- 0.95
9
10 Populations <- 2
11 prevalence <- c(0.25,0.75)
12 Group <- rep(1:Populations, each=N)
13
14 # Ensure replicable data:
15 set.seed(2017-11-21)
16
17 # We will assume test 1 is dependent of the others, but tests
   ↪ 2013
```

Verifying simulation results

```
1 library('tidyverse')

1 ## - Attaching packages ----- tidyverse 1.3.0 -
1 ## v ggplot2 3.2.1      v purrr 0.3.3
2 ## v tibble 2.1.3       v dplyr 0.8.3
3 ## v tidyr 1.0.2        v stringr 1.4.0
4 ## v readr 1.3.1        v forcats 0.4.0

1 ## - Conflicts ----- tidyverse_conflicts() -
2 ## x tidyr::extract() masks runjags::extract()
3 ## x dplyr::filter() masks stats::filter()
4 ## x dplyr::lag()      masks stats::lag()

1 ## Parameters
2
3 # True prevalence:
4 prev <- 0.2
5
6 # Probability of antibody response conditional on disease
  ↳ status:
```

Generating the model

Extreme care needed

Use `autohuiwalter` with argument `covon=TRUE`

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

1 ## The model and data have been written to ind3tests.bug in the
   ↪ current working directory
2 ## You should check and alter priors before running the model

1 ## Auto-generated Hui-Walter model created by script version 0.1
   ↪ on 2020-02-17

2
3 model{
4
5     ## Observation layer:
6
7     # Complete observations (N=400):
8     for(p in 1:Populations){
9         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 dependence has bigger CI but unbiased

Session 3c: Model selection

[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

- 1 * An approximation to LOO with general applicability
- 2 * Probably won't work for Hui-Walter though due to lack of
↳ independent data
- 3 * 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 version 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:

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

WAIC is also calculable from Stan models (easily?)

What would be useful to add to the `autohwiwaller` function?

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