# Session 5

How to interpret the latent class

Matt Denwood

2021-06-30

# Session 5: How to interpret the latent class

# Recap

- Adding more populations and more tests to a Hui-Walter model is technically easy
  - Particualrly if using template_huiwalter
- Verifying that the assumptions you are making are correct is harder
  - The sensitivity and specificity must be consistent across populations
  - Pairwise correlation between tests should be accounted for (with $>$2 tests)

# What exactly is our latent class?

Homework (reminder): think about what exactly the latent class is in these situations:

1. An antigen plus antibody test

# What exactly is our latent class?

Homework (reminder): think about what exactly the latent class is in these situations:

1. An antigen plus antibody test

- The latent status is probably close to the true disease status

# What exactly is our latent class?

Homework (reminder): think about what exactly the latent class is in these situations:

1. An antigen plus antibody test

- The latent status is probably close to the true disease status

1. Two antibody tests

# What exactly is our latent class?

Homework (reminder): think about what exactly the latent class is in these situations:

1. An antigen plus antibody test

- The latent status is probably close to the true disease status

1. Two antibody tests

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

# What exactly is our latent class?

Homework (reminder): think about what exactly the latent class is in these situations:

1. An antigen plus antibody test

- The latent status is probably close to the true disease status

1. Two antibody tests

- The latent status is actually 'producing antibodies' not 'diseased'
- What do we mean by "conditionally independent" (revisited) ?
    - Independent of each other conditional on the latent state
    - But the latent state is NOT always *disease*

# What exactly is our latent class?

Homework (reminder): think about what exactly the latent class is in these situations:

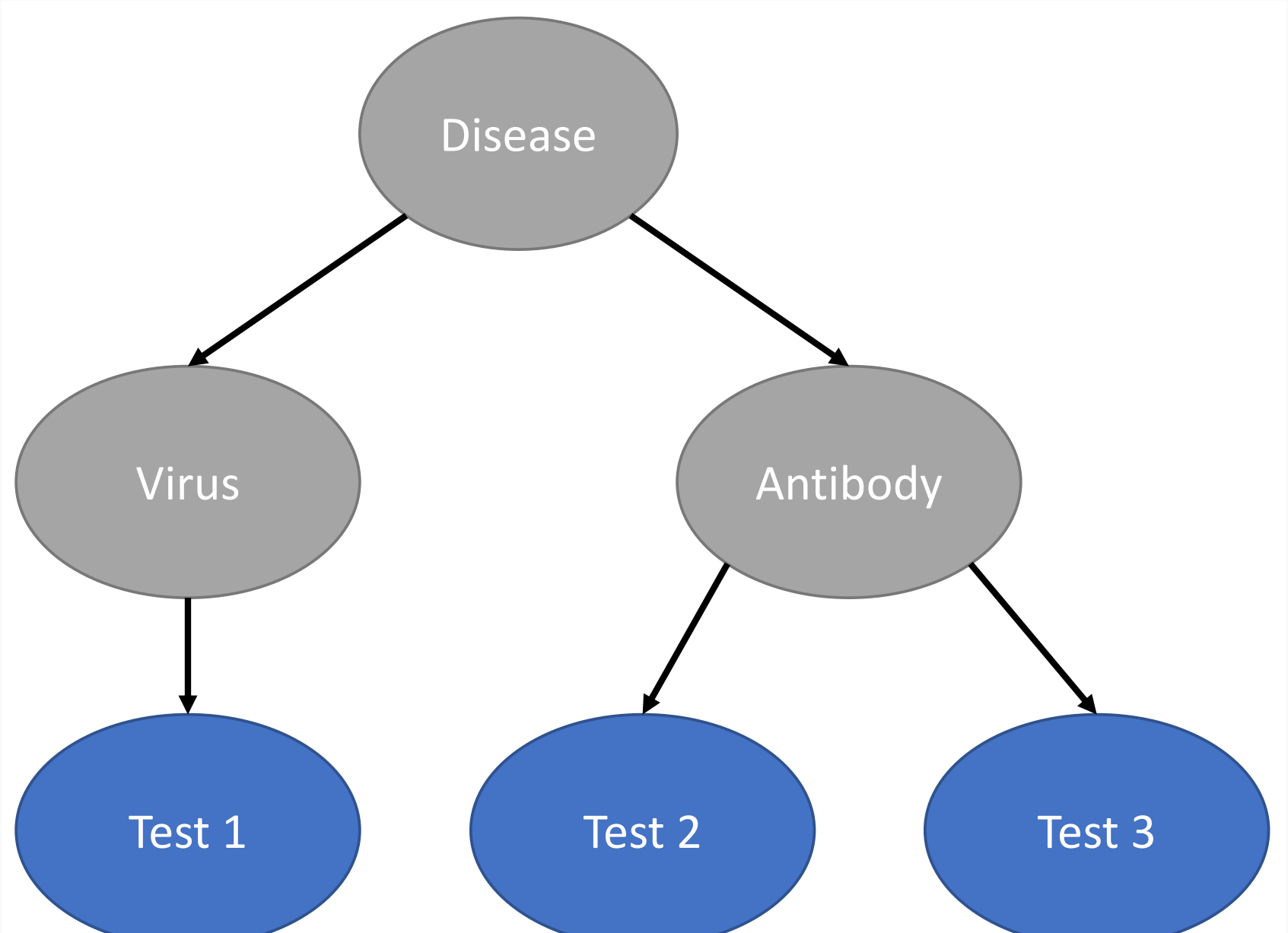
1. An antigen plus antibody test

- The latent status is probably close to the true disease status

1. Two antibody tests

- The latent status is actually 'producing antibodies' not 'diseased'
- What do we mean by "conditionally independent" (revisited) ?
    - Independent of each other conditional on the latent state
    - But the latent state is NOT always *disease*
- i.e. we're pulling **something** out of a hat, and deciding to call it a rabbit

# A hierarchy of latent states



# Branching of processes leading to test results

- Sometimes we have multiple tests that are detecting a similar thing
  - For example: two antibody tests and one antigen test
  - The antibody tests will be correlated

# Branching of processes leading to test results

- Sometimes we have multiple tests that are detecting a similar thing
  - For example: two antibody tests and one antigen test
  - The antibody tests will be correlated
- Or even three antibody tests where two are primed to detect the same thing, and one has a different target!
  - In this case all three tests are correlated, but two are more strongly correlated

# Data simulation

# Optional Exercise Code

```r
# Probability of antibody response conditional on disease status (really
↪  bad to illustrate the point):
se_antibody <- 0.5
sp_antibody <- 0.75
N <- 100
# Otherwise the parameters are as before

# True latent infection status as before:
true <- rbinom(N, 1, prevalence[Group])

# Latent class of antibody response conditional on the true status:
antibody <- rbinom(N, 1, se_antibody*true + (1-sp_antibody)*(1-true))

# Simulate test results for test 1 conditional on antibody status:
test1 <- rbinom(N, 1, se1*antibody + (1-sp1)*(1-antibody))
# etc

# Note that the overall sensitivity and specificity of the tests needs to
↪  be corrected for the antibody positive step:
overall_se1 <- se_antibody*se1 + (1-se_antibody)*(1-sp1)
overall_sp1 <- sp_antibody*sp1 + (1-sp_antibody)*(1-se1)
# etc
```

# Optional Solution

```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

# Probability of antibody response conditional on disease status (really
↪  bad to illustrate the point):
se_antibody <- 0.5
sp_antibody <- 0.75

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

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

# True latent infection status as before:
true <- rbinom(N, 1, prevalence[Group])
```

# What is sensitivity and specificity

TODO: show how overall Se and Sp relates to probability of test positive given antibody response etc

# Alternative model formulation

TODO: change to explicit serial Disease -> Antibody -> Test etc

Note that autocorrelation is terrible

```
model{

  for(i in 1:N){
    Status[i] ~ dcat(prob[i, ])

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

      logit(prev[i]) <- intercept + population_effect[Population[i]]
  }
```

```
  intercept ~ dnorm(0, 0.33)
  population_effect[1] <- 0
  for(p in 2:Pops){
    population_effect[p] ~ dnorm(0, 0.1)
  }
  se[1] ~ dbeta(1, 1)T(1-sp[1], )
  sp[1] ~ dbeta(1, 1)
  se[2] ~ dbeta(1, 1)T(1-sp[2], )
  sp[2] ~ dbeta(1, 1)

  #data# Status, N, Population, Pops
  #monitor# intercept, population_effect, se, sp
  #inits# intercept, population_effect, se, sp
}
```

- The main difference is the prior for prevalence in each population
- We also need to give initial values for `intercept` and `population_effect` rather than `prev`, and tell `run.jags` the data frame from which to extract the data (except `N` and `Pops`):

```
intercept <- list(chain1=-1, chain2=1)
population_effect <- list(chain1=c(NA, 1, -1), chain2=c(NA, -1, 1))
se <- list(chain1=c(0.5,0.99), chain2=c(0.99,0.5))
sp <- list(chain1=c(0.5,0.99), chain2=c(0.99,0.5))

simdata$Status <- with(simdata, factor(interaction(Test1, Test2),
↪  levels=c('0.0','1.0','0.1','1.1')))
N <- nrow(simdata)
Pops <- length(levels(simdata$Population))
glm_results <- run.jags('glm_hw.bug', n.chains=2, data=simdata)
```

# Publication of your results

STARD-BLCM: A helpful structure to ensure that papers contain all necessary information

If you use the software, please cite JAGS:

Plummer, M. (2003). JAGS : A Program for Analysis of Bayesian Graphical Models Using Gibbs Sampling JAGS : Just Another Gibbs Sampler. Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003), March 20–22,Vienna, Austria. ISSN 1609-395X. https://doi.org/10.1.1.13.3406

R:

```
citation()
##
## To cite R in publications use:
##
##   R Core Team (2021). R: A language and environment
##   for statistical computing. R Foundation for
```

# Discussion session 5

# Points to consider

1. Interpreting the results of latent class models is much more difficult than running them
2. How can we be sure that e.g. `probability of a positive test result conditional on the latent state` is the same thing as `sensitivity`?
3. How can we make sure that our publications contain all of the necessary information to allow others to interpret our findings?

# Exercise

1. Read the STARD-BLCM guidelines, checklist, and examples documents
2. Read the `Diagnosing diagnostic tests` paper
3. Be ready with questions for the group discussion!

# Summary

- Latent class models are MUCH more complex to interpret than traditional models
    - Take time to think about what the latent class means
- Think about which tests might be correlated and if you should include covariance terms
- Think about the biology of where your data comes from, particularly if populations are fundamentally different
- Follow the STARD checklist!