Session 4

Multi-test, multi-population models

Matt Denwood, Giles Innocent

2022-09-13

Why stop at two tests?

In *traditional* diagnostic test evaluation, one test is assumed to be a gold standard from which all other tests are evaluated

- So it makes no difference if you assess one test at a time or do multiple tests at the same time

Why stop at two tests?

In *traditional* diagnostic test evaluation, one test is assumed to be a gold standard from which all other tests are evaluated

- So it makes no difference if you assess one test at a time or do multiple tests at the same time

Using a latent class model each new test adds new information - so we should analyse all available test results in the same model

Simulating data: simple example

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

```
# Parameter values to simulate:
N <- 200
sensitivity <- c(0.8, 0.9, 0.95)
specificity <- c(0.95, 0.99, 0.95)

Populations <- 2
prevalence <- c(0.25,0.5)

data <- tibble(Population = sample(seq_len(Populations), N, replace=TRUE)) %>%
  mutate(Status = rbinom(N, 1, prevalence[Population])) %>%
  mutate(Test1 = rbinom(N, 1, sensitivity[1]*Status + (1-specificity[1])*(1-Status))) %>%
  mutate(Test2 = rbinom(N, 1, sensitivity[2]*Status + (1-specificity[2])*(1-Status))) %>%
  mutate(Test3 = rbinom(N, 1, sensitivity[3]*Status + (1-specificity[3])*(1-Status))) %>%
  select(-Status)
```

Model specification

Like for two tests, except it is now a 2x2x2 table

Model specification

Like for two tests, except it is now a 2x2x2 table

```
Tally[1:8,p] ~ dmulti(prob[1:8,p], TotalTests[p])

# Probability of observing Test1- Test2- Test3-
prob[1,p] <-  prev[p] * ((1-se[1])*(1-se[2])*(1-se[3]) +
              (1-prev[p]) * (sp[1]*sp[2]*sp[3])

# Probability of observing Test1+ Test2- Test3-
prob[2,p] <-  prev[p] * (se[1]*(1-se[2])*(1-se[3])) +
              (1-prev[p]) * ((1-sp[1])*sp[2]*sp[3])

## snip ##

# Probability of observing Test1+ Test2+ Test3+
prob[3,p] <-  prev[p] * (se[1]*se[2]*se[3]) +
              (1-prev[p]) * ((1-sp[1])*(1-sp[2])*(1-sp[3]))
```

Model specification

Like for two tests, except it is now a 2x2x2 table

```
Tally[1:8,p] ~ dmulti(prob[1:8,p], TotalTests[p])

# Probability of observing Test1- Test2- Test3-
prob[1,p] <-  prev[p] * ((1-se[1])*(1-se[2])*(1-se[3]) +
              (1-prev[p]) * (sp[1]*sp[2]*sp[3])

# Probability of observing Test1+ Test2- Test3-
prob[2,p] <-  prev[p] * (se[1]*(1-se[2])*(1-se[3])) +
              (1-prev[p]) * ((1-sp[1])*sp[2]*sp[3])

## snip ##

# Probability of observing Test1+ Test2+ Test3+
prob[3,p] <-  prev[p] * (se[1]*se[2]*se[3]) +
              (1-prev[p]) * ((1-sp[1])*(1-sp[2])*(1-sp[3]))
```

- We need to take **extreme** care with these equations, and the multinomial tabulation!!!

Are the tests conditionally independent?

- Example: we have one blood, one milk, and one faecal test
  - But the blood and milk test are basically the same test
  - Therefore they are more likely to give the same result

Are the tests conditionally independent?

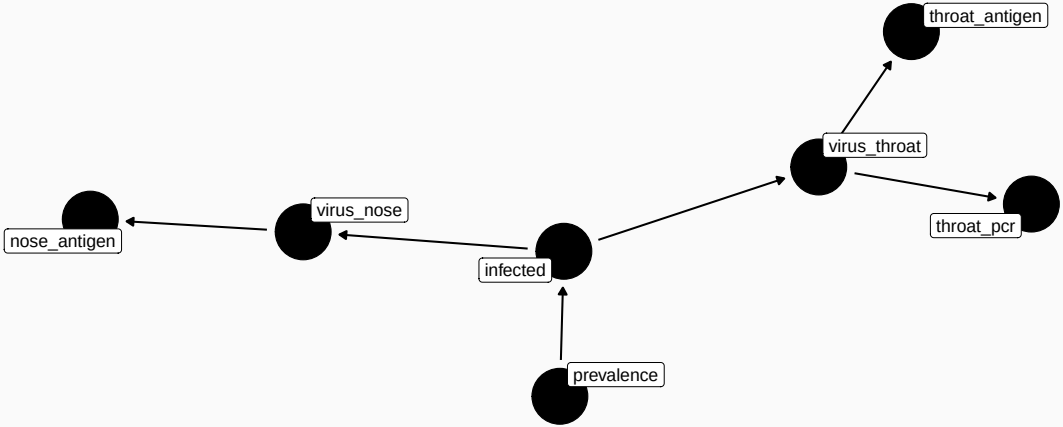
- Example: we have one blood, one milk, and one faecal test
  - But the blood and milk test are basically the same test
  - Therefore they are more likely to give the same result
- Example: we test people for COVID using an antigen test on a nasal swab, a PCR test on a throat swab, and the same antigen test on the same throat swab
  - The virus may be present in the throat, nose, neither, or both
  - But we use the same antigen test twice
    - Might it cross-react with the same non-target virus?

Are the tests conditionally independent?

- Example: we have one blood, one milk, and one faecal test
  - But the blood and milk test are basically the same test
  - Therefore they are more likely to give the same result
- Example: we test people for COVID using an antigen test on a nasal swab, a PCR test on a throat swab, and the same antigen test on the same throat swab
  - The virus may be present in the throat, nose, neither, or both
  - But we use the same antigen test twice
    - Might it cross-react with the same non-target virus?
- In both situations we have pairwise correlation between some of the tests

Directed Acyclic Graphs

- It may help you to visualise the relationships as a DAG:



Dealing with correlation: Covid example

It helps to consider the data simulation as a (simplified) biological process (where my parameters are not representative of real life!):

```
# The probability of infection with COVID in two populations:
prevalence <- c(0.01,0.05)
# The probability of shedding COVID in the nose conditional on infection:
nose_shedding <- 0.8
# The probability of shedding COVID in the throat conditional on infection:
throat_shedding <- 0.8
# The probability of detecting virus with the antigen test:
antigen_detection <- 0.75
# The probability of detecting virus with the PCR test:
pcr_detection <- 0.999
# The probability of random cross-reaction with the antigen test:
antigen_crossreact <- 0.05
# The probability of random cross-reaction with the PCR test:
pcr_crossreact <- 0.01
```

Dealing with correlation: Covid example

It helps to consider the data simulation as a (simplified) biological process (where my parameters are not representative of real life!):

```
# The probability of infection with COVID in two populations:
prevalence <- c(0.01,0.05)
# The probability of shedding COVID in the nose conditional on infection:
nose_shedding <- 0.8
# The probability of shedding COVID in the throat conditional on infection:
throat_shedding <- 0.8
# The probability of detecting virus with the antigen test:
antigen_detection <- 0.75
# The probability of detecting virus with the PCR test:
pcr_detection <- 0.999
# The probability of random cross-reaction with the antigen test:
antigen_crossreact <- 0.05
# The probability of random cross-reaction with the PCR test:
pcr_crossreact <- 0.01
```

Note: cross-reactions are assumed to be independent here!

Simulating latent states:

```
N <- 20000
Populations <- length(prevalence)

covid_data <- tibble(Population = sample(seq_len(Populations), N, replace=TRUE)) %>%
  ## True infection status:
  mutate(Status = rbinom(N, 1, prevalence[Population])) %>%
  ## Nose shedding status:
  mutate(Nose = Status * rbinom(N, 1, nose_shedding)) %>%
  ## Throat shedding status:
  mutate(Throat = Status * rbinom(N, 1, throat_shedding))
```

Simulating test results:

```
covid_data <- covid_data %>%
  ## The nose swab antigen test may be false or true positive:
  mutate(NoseAG = case_when(
    Nose == 1 ~ rbinom(N, 1, antigen_detection),
    Nose == 0 ~ rbinom(N, 1, antigen_crossreact)
  )) %>%
  ## The throat swab antigen test may be false or true positive:
  mutate(ThroatAG = case_when(
    Throat == 1 ~ rbinom(N, 1, antigen_detection),
    Throat == 0 ~ rbinom(N, 1, antigen_crossreact)
  )) %>%
  ## The PCR test may be false or true positive:
  mutate(ThroatPCR = case_when(
    Throat == 1 ~ rbinom(N, 1, pcr_detection),
    Throat == 0 ~ rbinom(N, 1, pcr_crossreact)
  ))
```

The overall sensitivity of the tests can be calculated as follows:

```
covid_sensitivity <- c(
  # Nose antigen:
  nose_shedding*antigen_detection + (1-nose_shedding)*antigen_crossreact,
  # Throat antigen:
  throat_shedding*antigen_detection + (1-throat_shedding)*antigen_crossreact,
  # Throat PCR:
  throat_shedding*pcr_detection + (1-throat_shedding)*pcr_crossreact
)
covid_sensitivity
## [1] 0.6100 0.6100 0.8012
```

The overall specificity of the tests is more straightforward:

```
covid_specificity <- c(
  # Nose antigen:
  1 - antigen_crossreact,
  # Throat antigen:
  1 - antigen_crossreact,
  # Throat PCR:
  1 - pcr_crossreact
)
covid_specificity
## [1] 0.95 0.95 0.99
```

The overall specificity of the tests is more straightforward:

```
covid_specificity <- c(
  # Nose antigen:
  1 - antigen_crossreact,
  # Throat antigen:
  1 - antigen_crossreact,
  # Throat PCR:
  1 - pcr_crossreact
)
covid_specificity
## [1] 0.95 0.95 0.99
```

However: this assumes that cross-reactions are independent!

Model specification

```
prob[1,p] <-  prev[p] * ((1-se[1])*(1-se[2])*(1-se[3])
                         +covse12 +covse13 +covse23) +
              (1-prev[p]) * (sp[1]*sp[2]*sp[3]
                             +covsp12 +covsp13 +covsp23)

prob[2,p] <- prev[p] * (se[1]*(1-se[2])*(1-se[3])
                           -covse12 -covse13 +covse23) +
               (1-prev[p]) * ((1-sp[1])*sp[2]*sp[3]
                              -covsp12 -covsp13 +covsp23)

## snip ##

# Covariance in sensitivity between tests 1 and 2:
covse12 ~ dunif( (se[1]-1)*(1-se[2]) ,
                 min(se[1],se[2]) - se[1]*se[2] )
# Covariance in specificity between tests 1 and 2:
covsp12 ~ dunif( (sp[1]-1)*(1-sp[2]) ,
                 min(sp[1],sp[2]) - sp[1]*sp[2] )
```

Model specification

```
prob[1,p] <-  prev[p] * ((1-se[1])*(1-se[2])*(1-se[3])
                         +covse12 +covse13 +covse23) +
              (1-prev[p]) * (sp[1]*sp[2]*sp[3]
                             +covsp12 +covsp13 +covsp23)

prob[2,p] <- prev[p] * (se[1]*(1-se[2])*(1-se[3])
                           -covse12 -covse13 +covse23) +
               (1-prev[p]) * ((1-sp[1])*sp[2]*sp[3]
                              -covsp12 -covsp13 +covsp23)

## snip ##

# Covariance in sensitivity between tests 1 and 2:
covse12 ~ dunif( (se[1]-1)*(1-se[2]) ,
                    min(se[1],se[2]) - se[1]*se[2] )
# Covariance in specificity between tests 1 and 2:
covsp12 ~ dunif( (sp[1]-1)*(1-sp[2]) ,
                    min(sp[1],sp[2]) - sp[1]*sp[2] )
```

It is quite easy to get the terms slightly wrong!

Template Hui-Walter

The model code and data format for an arbitrary number of populations (and tests) can be determined automatically using the template_huiwalter function from the runjags package:

```
template_huiwalter(
  covid_data %>% select(Population, NoseAG, ThroatAG, ThroatPCR),
  outfile = 'covidmodel.txt')
```

This generates self-contained model/data/initial values etc

```
model{

    ## Observation layer:

    # Complete observations (N=20000):
    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]
    }


    ## Observation probabilities:

    for(p in 1:Populations){

        # Probability of observing NoseAG- ThroatAG- ThroatPCR- 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 NoseAG- ThroatAG- ThroatPCR- from a true negative::
        sp_prob[1,p] <- (1-prev[p]) * (sp[1]*sp[2]*sp[3] +covsp12 +covsp13 +covsp23)

        # Probability of observing NoseAG+ ThroatAG- ThroatPCR- from a true positive::
        se_prob[2,p] <- prev[p] * (se[1]*(1-se[2])*(1-se[3]) -covse12 -covse13 +covse23)
        # Probability of observing NoseAG+ ThroatAG- ThroatPCR- from a true negative::
        sp_prob[2,p] <- (1-prev[p]) * ((1-sp[1])*sp[2]*sp[3] -covsp12 -covsp13 +covsp23)
```

```
## Data:
data{
"Populations" <- 2
"N_RRR" <- c(9934, 10066)
"Tally_RRR" <- structure(c(8795, 466, 496, 18, 89, 17, 20, 33, 8579, 507, 432, 25, 132, 74,
↪  124, 193), .Dim = c(8, 2))
}
```

And can be run directly from R:

```
results <- run.jags('covidmodel.txt')
## Loading required namespace: rjags
results
```

	Lower95	Median	Upper95	SSeff	psrf
se[1]	0.568	0.618	0.667	8021	1
se[2]	0.677	0.727	0.778	6348	1
se[3]	0.947	0.983	1.000	5633	1
sp[1]	0.944	0.948	0.951	11977	1
sp[2]	0.947	0.950	0.953	11924	1
sp[3]	0.989	0.990	0.992	6567	1
prev[1]	0.005	0.007	0.009	9641	1
prev[2]	0.039	0.043	0.048	7690	1
covse12	0.000	0.000	0.000	NA	NA
covsp12	0.000	0.000	0.000	NA	NA
covse13	0.000	0.000	0.000	NA	NA
covsp13	0.000	0.000	0.000	NA	NA
covse23	0.000	0.000	0.000	NA	NA
covsp23	0.000	0.000	0.000	NA	NA

Template Hui-Walter

- Modifying priors must still be done directly in the model file
  - Same for adding .RNG.seed and the deviance monitor
- The model needs to be re-generated if the data changes
  - But remember that your modified priors will be reset
- There must be a single column for the population (as a factor), and all of the other columns (either factor, logical or numeric) are interpreted as being test results

- Covariance terms are also calculated as proportion of possible correlation e.g.:

```
# Covariance in sensitivity between NoseAG and ThroatAG tests:
# covse12 ~ dunif( (se[1]-1)*(1-se[2]) , min(se[1],se[2]) - se[1]*se[2] )  ## if the
↪  sensitivity of these tests may be correlated
covse12 <- 0  ## if the sensitivity of these tests can be assumed to be independent
# Calculated relative to the min/max for ease of interpretation:
corse12 <- ifelse(covse12 < 0, -covse12 / ((se[1]-1)*(1-se[2])), covse12 /
↪  (min(se[1],se[2]) - se[1]*se[2]))
```

- Covariance terms are also calculated as proportion of possible correlation e.g.:

```
# Covariance in sensitivity between NoseAG and ThroatAG tests:
# covse12 ~ dunif( (se[1]-1)*(1-se[2]) , min(se[1],se[2]) - se[1]*se[2] )  ## if the
↪  sensitivity of these tests may be correlated
covse12 <- 0  ## if the sensitivity of these tests can be assumed to be independent
# Calculated relative to the min/max for ease of interpretation:
corse12 <- ifelse(covse12 < 0, -covse12 / ((se[1]-1)*(1-se[2])), covse12 /
↪  (min(se[1],se[2]) - se[1]*se[2]))
```

- But covariance terms are all deactivated by default!

Activating covariance terms

Find the lines for the covariances that we want to activate (i.e. the two Throat tests):

```
# Covariance in sensitivity between ThroatAG and ThroatPCR tests:
# covse23 ~ dunif( (se[2]-1)*(1-se[3]) , min(se[2],se[3]) - se[2]*se[3] )  ## if the
↪  sensitivity of these tests may be correlated
covse23 <- 0  ## if the sensitivity of these tests can be assumed to be independent

# Covariance in specificity between ThroatAG and ThroatPCR tests:
# covsp23 ~ dunif( (sp[2]-1)*(1-sp[3]) , min(sp[2],sp[3]) - sp[2]*sp[3] )  ## if the
↪  specificity of these tests may be correlated
covsp23 <- 0  ## if the specificity of these tests can be assumed to be independent
```

And edit so it looks like:

```
# Covariance in sensitivity between ThroatAG and ThroatPCR tests:
covse23 ~ dunif( (se[2]-1)*(1-se[3]) , min(se[2],se[3]) - se[2]*se[3] )  ## if the
↪  sensitivity of these tests may be correlated
 # covse23 <- 0  ## if the sensitivity of these tests can be assumed to be independent

# Covariance in specificity between ThroatAG and ThroatPCR tests:
covsp23 ~ dunif( (sp[2]-1)*(1-sp[3]) , min(sp[2],sp[3]) - sp[2]*sp[3] )  ## if the
↪  specificity of these tests may be correlated
 # covsp23 <- 0  ## if the specificity of these tests can be assumed to be independent
```

[i.e. swap the comments around]

You will also need to uncomment out the relevant initial values for BOTH chains (on lines 132-137 and 128-133):

```
# "covse12" <- 0
# "covse13" <- 0
# "covse23" <- 0
# "covsp12" <- 0
# "covsp13" <- 0
# "covsp23" <- 0
```

So that they look like:

```
# "covse12" <- 0
# "covse13" <- 0
"covse23" <- 0
# "covsp12" <- 0
# "covsp13" <- 0
"covsp23" <- 0
```

```
results <- run.jags('covidmodel.txt', sample=50000)
results
##
## JAGS model summary statistics from 100000 samples (chains = 2; adapt+burnin = 5000):
##
##             Lower95     Median    Upper95       Mean
## se[1]       0.57156    0.62952    0.68945    0.63014
## se[2]       0.52119    0.64401    0.75044    0.63972
## se[3]       0.75446    0.90081          1       0.89
## sp[1]       0.94512    0.94912    0.95339    0.94922
## sp[2]       0.94567    0.94891    0.95235    0.94889
## sp[3]       0.98757    0.98981     0.9919    0.98979
## prev[1]   0.0051835  0.0074605  0.0099441  0.0075498
## prev[2]     0.03966   0.046552   0.056235   0.047134
## covse12           0          0          0          0
## corse12           0          0          0          0
## covsp12           0          0          0          0
## corsp12           0          0          0          0
## covse13           0          0          0          0
## corse13           0          0          0          0
## covsp13           0          0          0          0
## corsp13           0          0          0          0
## covse23  -0.0028881   0.038433   0.082509   0.038724
## corse23      0.07303    0.58601    0.99966    0.54811
## covsp23 -0.00038021 0.00015414 0.00067602 0.00015539
## corsp23     -0.6184   0.015891   0.027342  -0.070742
```

Practical considerations

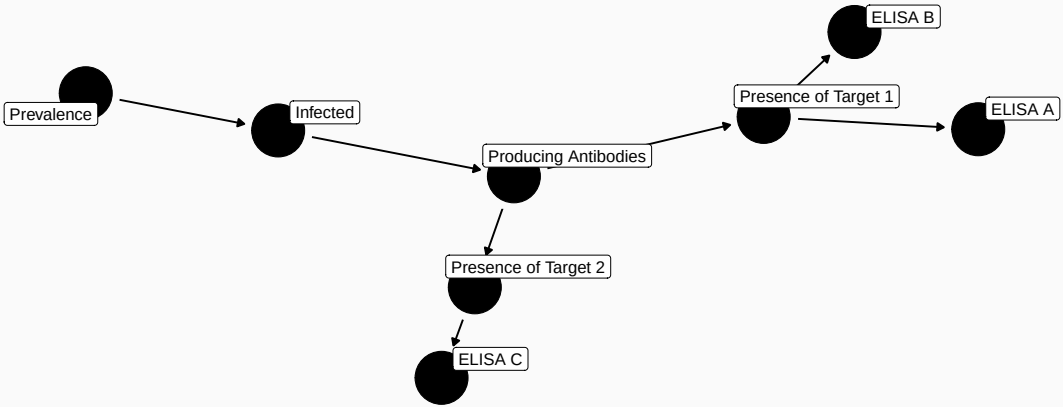
- Correlation terms add complexity to the model in terms of:
  - Opportunity to make a coding mistake
  - Reduced identifiability

Practical considerations

- Correlation terms add complexity to the model in terms of:
  - Opportunity to make a coding mistake
  - Reduced identifiability
- The template_huiwalter function helps us with coding mistakes
- Only careful consideration of covariance terms can help us with identifiability

How to interpret the latent class

A hierarchy of latent states



What is sensitivity and specificity?

- The probability of test status conditional on true disease status?
- The probability of test status conditional on the latent state?

What is sensitivity and specificity?

- The probability of test status conditional on true disease status?
- The probability of test status conditional on the latent state?

So is the latent state the same as the true disease state?

What is sensitivity and specificity?

- The probability of test status conditional on true disease status?
- The probability of test status conditional on the latent state?

So is the latent state the same as the true disease state?

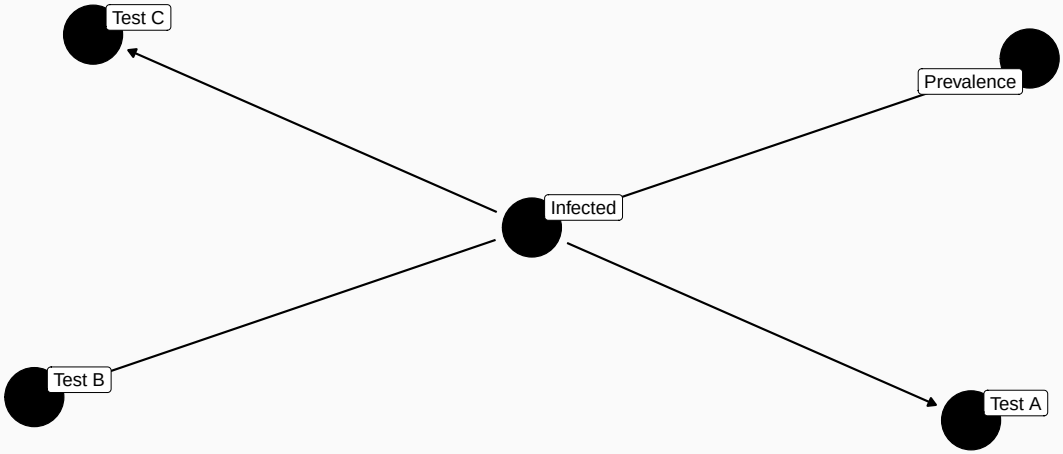
Important quote:

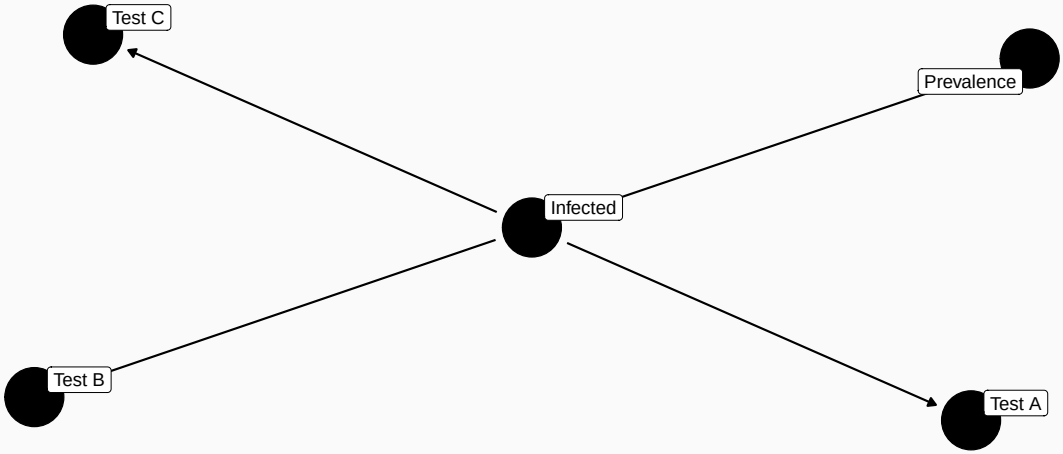
"Latent class models involve pulling **something** out of a hat, and deciding to call it a rabbit"

- Nils Toft

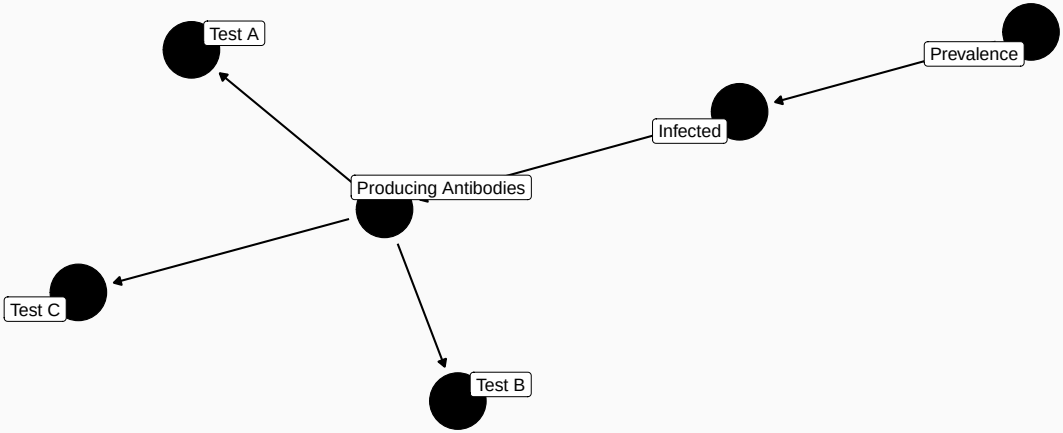
When should we correct for correlation?

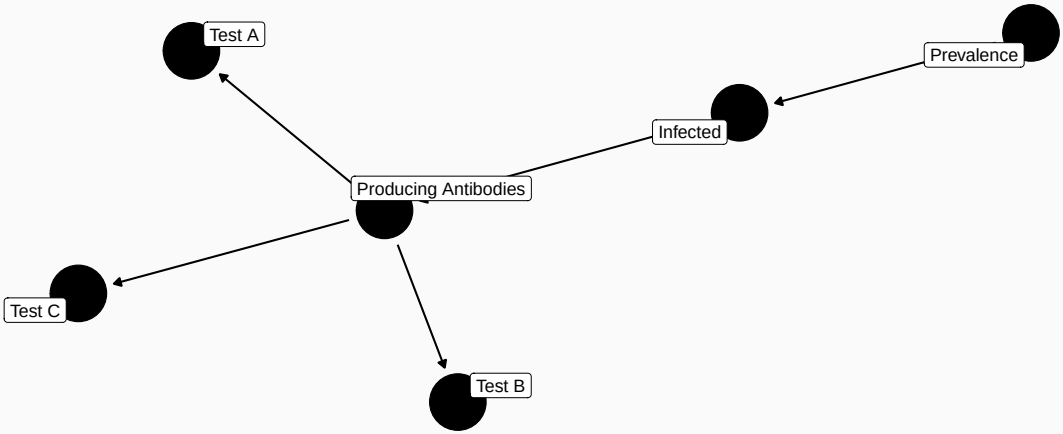
- For each of the following DAG:
  - Consider what is the latent class
  - Consider which correlation terms we should include and why


Test C
Prevalence
Infected
Test B
Test A

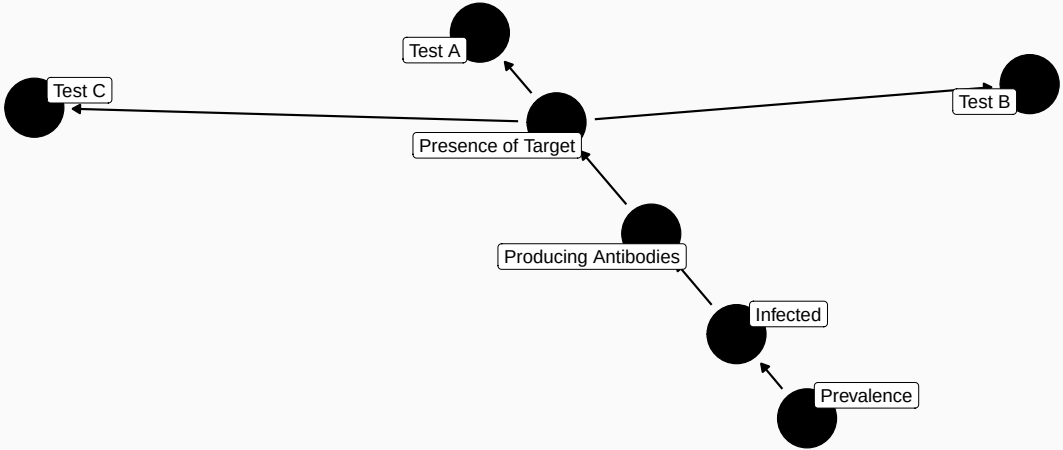


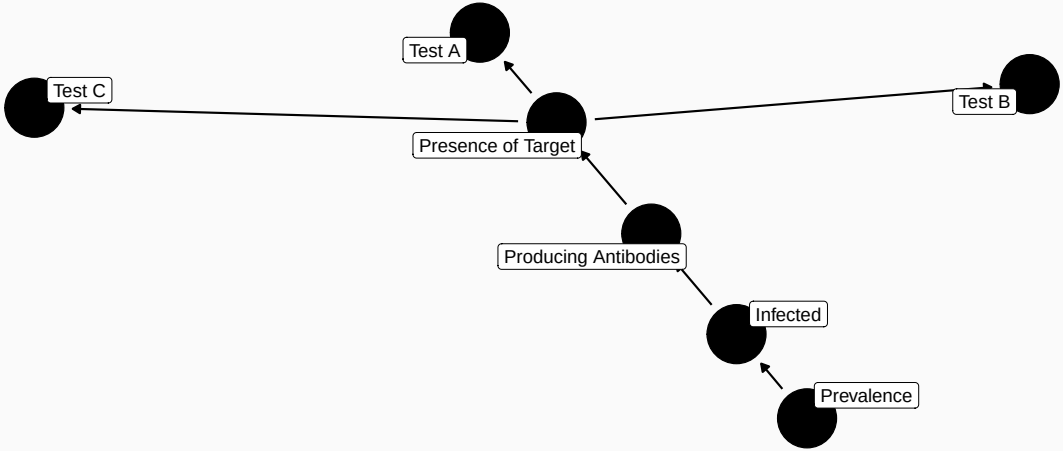
- No correlation to model!


Test A
Prevalence
Infected
Producing Antibodies
Test C
Test B

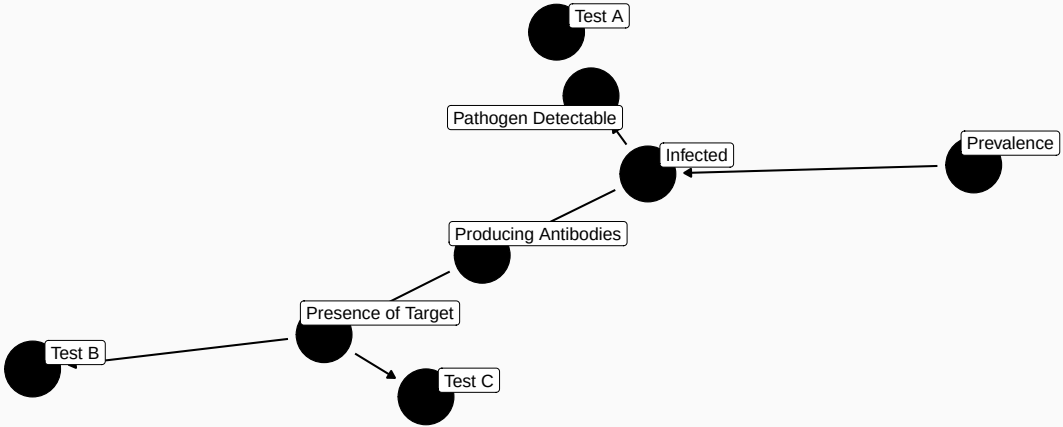


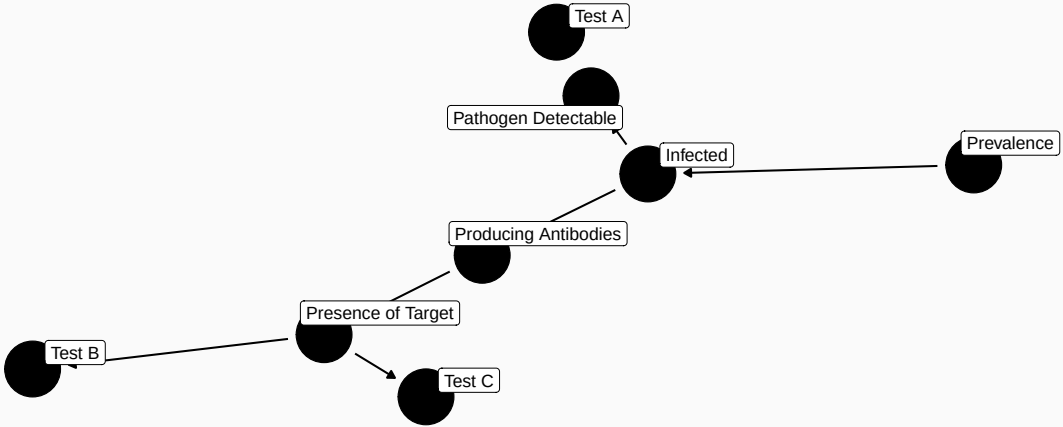
- No correlation to model . . . but “infected” is not the latent class


Test A
Test C
Test B
Presence of Target
Producing Antibodies
Infected
Prevalence

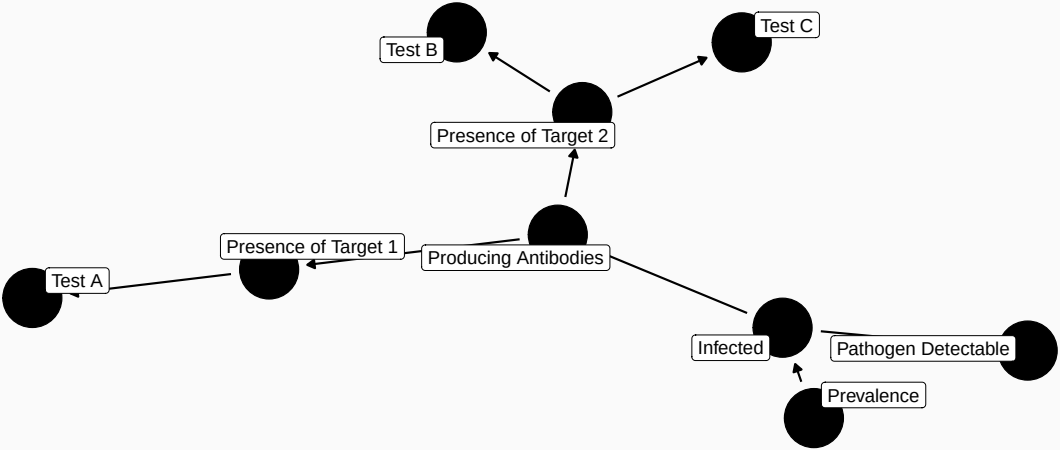


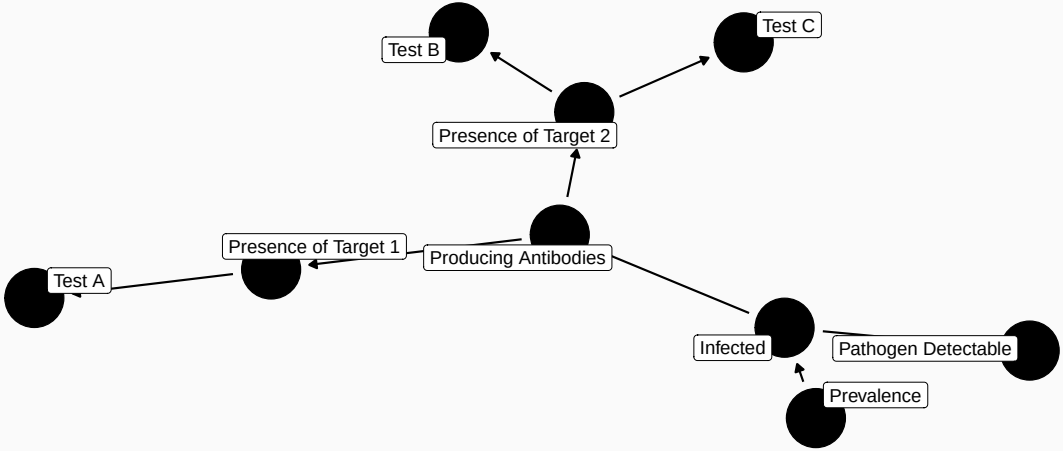
- Same as above!


Test A
Pathogen Detectable
Infected
Prevalence
Producing Antibodies
Presence of Target
Test B
Test C

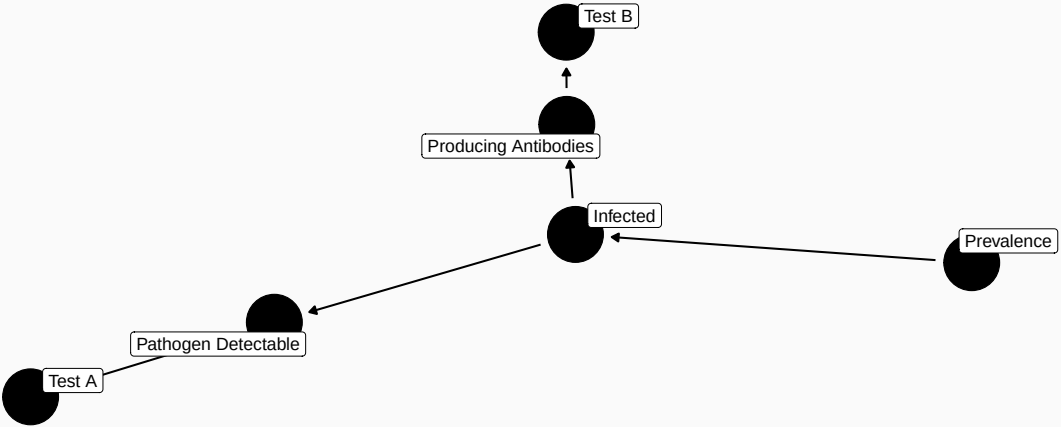


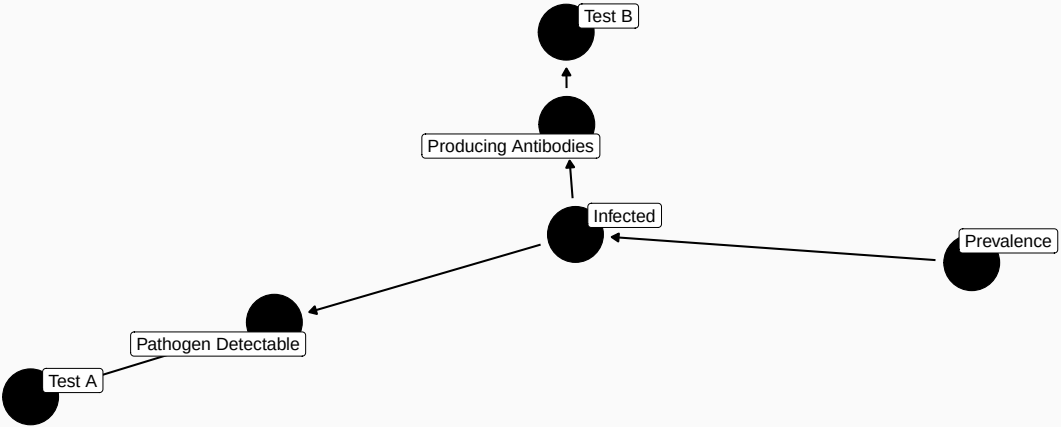
- Tests B and C are correlated


Test B
Test C
Presence of Target 2
Presence of Target 1
Producing Antibodies
Test A
Infected
Pathogen Detectable
Prevalence

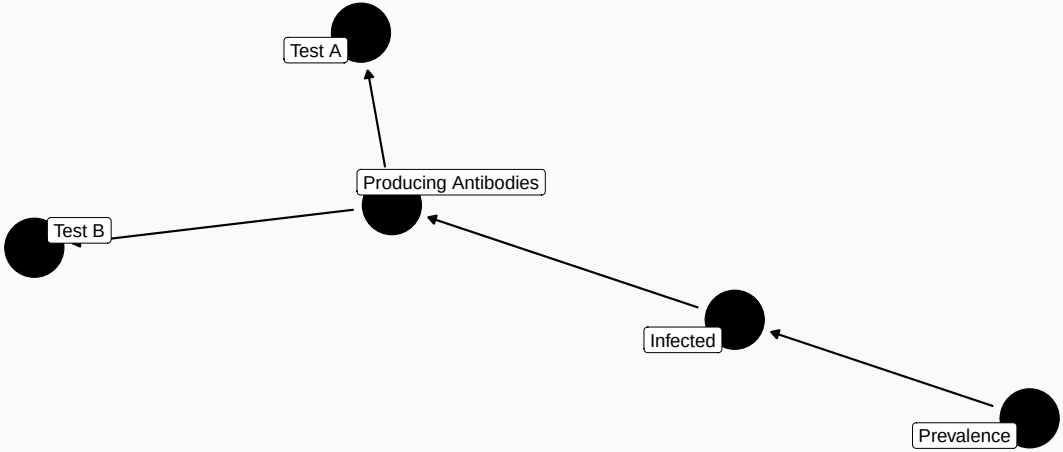


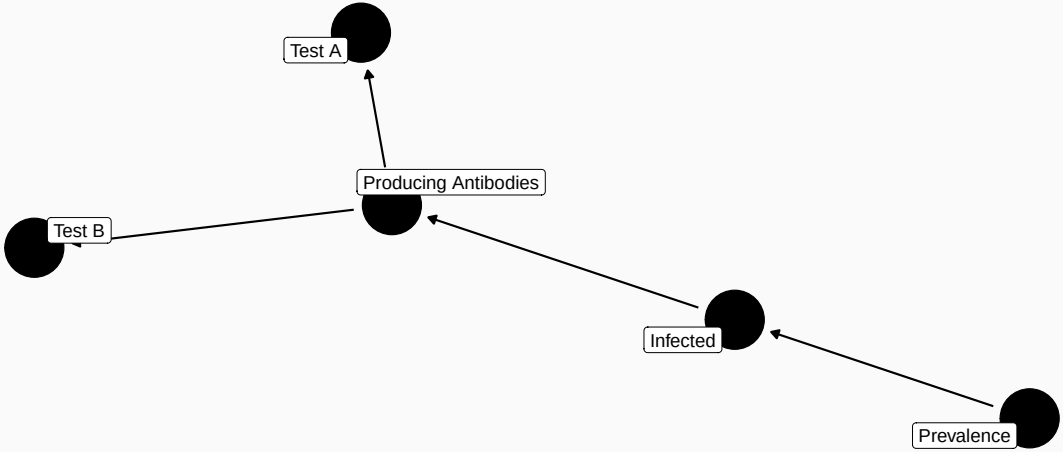
- All tests are correlated with respect to infected BUT infected is not the latent class
- Tests B and C are correlated with respect to antibodies - but maybe not substantially?


Test B
Producing Antibodies
Infected
Prevalence
Pathogen Detectable
Test A



- No correlation to model


Test A
Producing Antibodies
Test B
Infected
Prevalence



- No correlation to model - but “infected” is not the latent class

Publication of your results

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

- You should follow this and refer to it in your articles!

Publication of your results

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

- You should follow this and refer to it in your articles!

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

And R:

```
citation()
##
## To cite R in publications use:
##
##   R Core Team (2022). R: A language and environment
##   for statistical computing. R Foundation for
##   Statistical Computing, Vienna, Austria. URL
##   https://www.R-project.org/.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {R: A Language and Environment for Statistical Computing},
##     author = {{R Core Team}},
##     organization = {R Foundation for Statistical Computing},
##     address = {Vienna, Austria},
##     year = {2022},
##     url = {https://www.R-project.org/},
##   }
##
## We have invested a lot of time and effort in creating
## R, please cite it when using it for data analysis.
## See also 'citation("pkgname")' for citing R packages.
```

And runjags:

```
citation("runjags")
##
## To cite runjags in publications use:
##
##   Matthew J. Denwood (2016). runjags: An R Package
##   Providing Interface Utilities, Model Templates,
##   Parallel Computing Methods and Additional
##   Distributions for MCMC Models in JAGS. Journal of
##   Statistical Software, 71(9), 1-25.
##   doi:10.18637/jss.v071.i09
##
## A BibTeX entry for LaTeX users is
##
##   @Article{,
##     title = {{runjags}: An {R} Package Providing Interface Utilities, Model Templates,
↪  Parallel Computing Methods and Additional Distributions for {MCMC} Models in {JAGS}},
##     author = {Matthew J. Denwood},
##     journal = {Journal of Statistical Software},
##     year = {2016},
##     volume = {71},
##     number = {9},
##     pages = {1--25},
##     doi = {10.18637/jss.v071.i09},
```

Practical session 4

Points to consider

1. How does including a third test impact the inference for the first two tests?
2. What happens if we include correlation between tests?

Summary

- Including multiple tests is technically easy
  - But philosophically more difficult!!!
- Complexity of adding correlation terms increases non-linearly with more tests
  - Probably best to stick to correlations with biological justification?
- Adding/removing test results may change the posterior for
  - Other test Se / Sp
  - Prevalence