Session 4

Multi-test, multi-population models

Matt Denwood

2021-06-29

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

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

TODO: clean up R code

```
# Parameter values to simulate:
N <- 200
se <- c(0.8, 0.9, 0.95)
sp <- c(0.95, 0.99, 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, se[1]*true + (1-sp[1])*(1-true))
# Simulate test results for test 2:
test2 <- rbinom(N, 1, se[2]*true + (1-sp[2])*(1-true))
# Simulate test results for test 3:
test3 <- rbinom(N, 1, se[3]*true + (1-sp[3])*(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

    - We need to take **extreme** care with multinomial tabulation

Model specification

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

  - We need to take **extreme** care with multinomial tabulation

TODO: show equations

Are the tests conditionally independent?

- What do we mean by "conditionally independent?"

  - Independent of each other conditional on the latent state

Are the tests conditionally independent?

- What do we mean by "conditionally independent?"

  - Independent of each other conditional on the latent state

- Example: we have one PCR test for viral DNA and two antibody tests

  - The two antibody tests are more likely to give the same result than the PCR test

Are the tests conditionally independent?

- What do we mean by "conditionally independent?"
  - Independent of each other conditional on the latent state

- Example: we have one PCR test for viral DNA and two antibody tests
  - The two antibody tests are more likely to give the same result than the PCR test

- Example: we have one blood test, one milk test, and one faecal test
  - But the blood and milk test are basically the same test, just on different samples
  - Therefore the blood and milk tests are more likely to give the same result

6

Are the tests conditionally independent?

- What do we mean by "conditionally independent?"

    - Independent of each other conditional on the latent state

- Example: we have one PCR test for viral DNA and two antibody tests

    - The two antibody tests are more likely to give the same result than the PCR test

- Example: we have one blood test, one milk test, and one faecal test

    - But the blood and milk test are basically the same test, just on different samples
    - Therefore the blood and milk tests are more likely to give the same result

- In both situations we have pairwise correlation between same

Dealing with correlation

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

```
# 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 <- rep(1:Populations, each=N)

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

# The probability of an antibody response given disease:
abse <- 0.8
# The probability of no antibody response given no disease:
absp <- 1 - 0.2
```

```r
# Simulate the true latent state:
true <- rbinom(N*Populations, 1, prevalence[Group])

# Tests 1 & 2 will be co-dependent on antibody response:
antibody <- rbinom(N*Populations, 1, abse*true + (1-absp)*(1-true))
# Simulate test 1 & 2 results based on this other latent state:
test1 <- rbinom(N*Populations, 1, se1*antibody + (1-sp1)*(1-antibody))
test2 <- rbinom(N*Populations, 1, se2*antibody + (1-sp2)*(1-antibody))

# Simulate test results for the independent test 3:
test3 <- rbinom(N*Populations, 1, se3*true + (1-sp3)*(1-true))

ind3tests <- data.frame(Population=Group, Test1=test1, Test2=test2,
↪   Test3=test3)
```

```r
# The overall sensitivity of the correlated tests is:
abse*se1 + (1-abse)*(1-sp1)
## [1] 0.65
abse*se2 + (1-abse)*(1-sp2)
## [1] 0.722
# The overall specificity of the correlated tests is:
absp*sp1 + (1-absp)*(1-se1)
## [1] 0.8
absp*sp2 + (1-absp)*(1-se2)
## [1] 0.812
```

```r
# The overall sensitivity of the correlated tests is:
abse*se1 + (1-abse)*(1-sp1)
## [1] 0.65
abse*se2 + (1-abse)*(1-sp2)
## [1] 0.722
# The overall specificity of the correlated tests is:
absp*sp1 + (1-absp)*(1-se1)
## [1] 0.8
absp*sp2 + (1-absp)*(1-se2)
## [1] 0.812
```

We need to think carefully about what we are conditioning on when interpreting sensitivity and specificity!

Model specification

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

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

    ...

    # 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] )
```

Generating the model

First use template_huiwalter to create a model file:

```
template_huiwalter(ind3tests, 'auto3tihw.bug')
```

Then find the lines for the covariances that we want to activate:

```
# Covariance in sensitivity between Test1 and Test2 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
# Covariance in specificity between Test1 and Test2 tests:
# covsp12 ~ dunif( (sp[1]-1)*(1-sp[2]) , min(sp[1],sp[2]) - sp[1]*sp[2]
↪ ) ## if the specificity of these tests may be correlated
 covsp12 <- 0 ## if the specificity of these tests can be assumed to be
 ↪ independent
```

And edit so it looks like:

```
# Covariance in sensitivity between Test1 and Test2 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
# Covariance in specificity between Test1 and Test2 tests:
covsp12 ~ dunif( (sp[1]-1)*(1-sp[2]) , min(sp[1],sp[2]) - sp[1]*sp[2] )
↪  ## if the specificity of these tests may be correlated
 # covsp12 <- 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 117-122 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('auto3tihw.bug')
## Loading required namespace: rjags
```

Practical considerations

- Adds complexity to the model i.e. reduces identifiability

- Probabilities need to be constrained

- We need extreme extreme care to make sure the equations are correct

Template Hui-Walter

We would usually start with individual-level data in a dataframe:

```
se1 <- 0.9; se2 <- 0.8; sp1 <- 0.95; sp2 <- 0.99
N <- 100
prevalences <- tribble(~Population, ~Prevalence,
                        1, 0.1,
                        2, 0.5,
                        3, 0.9 ) %>%
  mutate(Population = factor(Population, levels=1:3,
  ↪ labels=str_c("Pop_",1:3)))

simdata <- tibble(Population = sample(levels(prevalences$Population), N,
↪ replace=TRUE)) %>%
  left_join(prevalences, by="Population") %>%
  mutate(Status = rbinom(N, 1, Prevalence)) %>%
  mutate(Test_1 = rbinom(N, 1, se[1]*Status + (1-sp[1])*(1-Status))) %>%
  mutate(Test_2 = rbinom(N, 1, se[2]*Status + (1-sp[2])*(1-Status)))
```

```r
head(simdata)
```

```
## # A tibble: 6 x 5
##   Population Prevalence Status Test_1 Test_2
##   <chr>           <dbl>  <int>  <int>  <int>
## 1 Pop_1             0.1      0      0      0
## 2 Pop_3             0.9      1      0      1
## 3 Pop_1             0.1      0      0      0
## 4 Pop_1             0.1      0      0      0
## 5 Pop_1             0.1      0      0      0
## 6 Pop_1             0.1      0      0      0
```

[Except that Prevalence and Status would not normally be known!]

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 runjas package:

```
template_huiwalter(simdata %>% select(Population, Test_1, Test_2),
↪   outfile='autohw.bug')
```

This generates self-contained model/data/initial values etc (you can ignore covse and covsp for now):

```
model{

    ## Observation layer:

    # Complete observations (N=100):
    for(p in 1:Populations){
        Tally_RR[1:4,p] ~ dmulti(prob_RR[1:4,p], N_RR[p])

        prob_RR[1:4,p] <- se_prob[1:4,p] + sp_prob[1:4,p]
    }


    ## Observation probabilities:

    for(p in 1:Populations){

        # Probability of observing Test_1- Test_2- from a true
        ↪   positive::
        se_prob[1,p] <- prev[p] * ((1-se[1])*(1-se[2]) +covse12)
        # Probability of observing Test_1- Test_2- from a true
        ↪   negative::
        sp_prob[1,p] <- (1-prev[p]) * (sp[1]*sp[2] +covsp12)
```

And can be run directly from R:

```
results <- run.jags('autohw.bug')
results
```

	Lower95	Median	Upper95	SSeff	psrf
se[1]	0.567	0.710	0.843	9833	1.000
se[2]	0.834	0.941	1.000	5617	1.001
sp[1]	0.809	0.909	0.988	8267	1.000
sp[2]	0.817	0.928	1.000	6651	1.000
prev[1]	0.000	0.101	0.235	6503	1.000
prev[2]	0.239	0.444	0.644	8811	1.000
prev[3]	0.757	0.885	0.993	7162	1.000
covse12	0.000	0.000	0.000	NA	NA
covsp12	0.000	0.000	0.000	NA	NA

- Modifying priors must still be done directly in the model file

- 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

NB: change priors, turn on/off covariances, add deviance monitor

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?

3. Can we include correlation if we only have 2 tests?

Summary

- Including multiple tests is technically easy but philosophically more difficult

- Complexity of adding correlation terms increases 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
    - Prevlanece

22

Summary

- Including multiple tests is technically easy but philosophically more difficult

- Complexity of adding correlation terms increases 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
    - Prevlanece

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

1. An antigen plus antibody test