Session 6

Validation of model assumptions

Matt Denwood

2022-06-09

Key model assumptions

NOTE: THIS MATERIAL IS NOT YET FINALISED, PLEASE CHECK BACK SOON!

The following model assumptions are critical:

- Consistent sensitivity and specificity across populations

- Populations are not based on a diagnostic test that is correlated with those used in the model

- Any missing data is missing completely at random (MCAR) or missing at random (MAR)

- Any between-test correlation structure is described (for >=3 tests)

Types of missingness

MCAR: Missing completely at random

- There is absolutely no pattern to the missingness
- This is the best kind

Types of missingness

MCAR: Missing completely at random

- There is absolutely no pattern to the missingness
- This is the best kind

MAR: Missing at random

- There is a pattern to the missingness but we know what it is
- This is usually possible to deal with but needs some consideration

Types of missingness

MCAR: Missing completely at random

- There is absolutely no pattern to the missingness
- This is the best kind

MAR: Missing at random

- There is a pattern to the missingness but we know what it is
- This is usually possible to deal with but needs some consideration

MNAR: Missing not at random

- There is an unknown (or unrecorded) pattern to the missingness
- It is therefore possible that the prevalence is confounded with missingness

Missingness and template Hui-Walter

We can simulate MCAR data as follows:

```
set.seed(2021-06-30)
# Parameter values to simulate:
N <- 1000
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)
```

Now introduce missingness in all 3 tests:

```
missingness <- c(0.1, 0.2, 0.3)
data <- data %>%
  mutate(Test1 = case_when(
    rbinom(n(), 1, missingness[1]) == 1L ~ NA_integer_,
    TRUE ~ Test1
  )) %>%
  mutate(Test2 = case_when(
    rbinom(n(), 1, missingness[2]) == 1L ~ NA_integer_,
    TRUE ~ Test2
  )) %>%
  mutate(Test3 = case_when(
    rbinom(n(), 1, missingness[3]) == 1L ~ NA_integer_,
    TRUE ~ Test3
  ))
```

```r
data %>% count(Missing1 = is.na(Test1), Missing2 = is.na(Test2),
↪  Missing3 = is.na(Test3))
## # A tibble: 8 x 4
##   Missing1 Missing2 Missing3     n
##   <lgl>    <lgl>    <lgl>    <int>
## 1 FALSE    FALSE    FALSE      513
## 2 FALSE    FALSE    TRUE       210
## 3 FALSE    TRUE     FALSE      126
## 4 FALSE    TRUE     TRUE        56
## 5 TRUE     FALSE    FALSE       54
## 6 TRUE     FALSE    TRUE        20
## 7 TRUE     TRUE     FALSE       14
## 8 TRUE     TRUE     TRUE         7
```

We can simply feed this data to `template_huiwalter`:

```
template_huiwalter(data, outfile="huiwalter_MAR.txt")
## The model and data have been written to huiwalter_MAR.txt in the
↪  current working directory
## You should check and alter priors before running the model
```

What does that look like. . . ?

```
model{

    ## Observation layer:

    # Complete observations (N=513):
    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]
    }
```

```
# Partial observations (Test1: Recorded, Test2: Missing, Test3:
↪  Missing; N=56):
for(p in 1:Populations){
    Tally_RMM[1:2,p] ~ dmulti(prob_RMM[1:2,p], N_RMM[p])

    prob_RMM[1:2,p] <-  se_prob[c(1,2),p] + sp_prob[c(1,2),p] +
                        se_prob[c(3,4),p] + sp_prob[c(3,4),p] +
                        se_prob[c(5,6),p] + sp_prob[c(5,6),p] +
                        se_prob[c(7,8),p] + sp_prob[c(7,8),p]
}

# Partial observations (Test1: Recorded, Test2: Recorded, Test3:
↪  Missing; N=210):
for(p in 1:Populations){
    Tally_RRM[1:4,p] ~ dmulti(prob_RRM[1:4,p], N_RRM[p])

    prob_RRM[1:4,p] <-  se_prob[c(1,2,3,4),p] +
    ↪  sp_prob[c(1,2,3,4),p] +
                        se_prob[c(5,6,7,8),p] +
                        ↪  sp_prob[c(5,6,7,8),p]
}
```

```
# Partial observations (Test1: Missing, Test2: Recorded, Test3:
↪  Recorded; N=54):
for(p in 1:Populations){
    Tally_MRR[1:4,p] ~ dmulti(prob_MRR[1:4,p], N_MRR[p])

    prob_MRR[1:4,p] <-  se_prob[c(1,3,5,7),p] +
    ↪  sp_prob[c(1,3,5,7),p] +
                        se_prob[c(2,4,6,8),p] +
                        ↪  sp_prob[c(2,4,6,8),p]
}

# Partial observations (Test1: Missing, Test2: Recorded, Test3:
↪  Missing; N=20):
for(p in 1:Populations){
    Tally_MRM[1:2,p] ~ dmulti(prob_MRM[1:2,p], N_MRM[p])

    prob_MRM[1:2,p] <-  se_prob[c(1,3),p] + sp_prob[c(1,3),p] +
                        se_prob[c(2,4),p] + sp_prob[c(2,4),p] +
                        se_prob[c(5,7),p] + sp_prob[c(5,7),p] +
                        se_prob[c(6,8),p] + sp_prob[c(6,8),p]
}
```

```
# Partial observations (Test1: Missing, Test2: Missing, Test3:
↪  Recorded; N=14):
for(p in 1:Populations){
    Tally_MMR[1:2,p] ~ dmulti(prob_MMR[1:2,p], N_MMR[p])

    prob_MMR[1:2,p] <-  se_prob[c(1,5),p] + sp_prob[c(1,5),p] +
                        se_prob[c(2,6),p] + sp_prob[c(2,6),p] +
                        se_prob[c(3,7),p] + sp_prob[c(3,7),p] +
                        se_prob[c(4,8),p] + sp_prob[c(4,8),p]
}
```

```
# Partial observations (Test1: Missing, Test2: Missing, Test3:
↪  Recorded; N=14):
for(p in 1:Populations){
    Tally_MMR[1:2,p] ~ dmulti(prob_MMR[1:2,p], N_MMR[p])

    prob_MMR[1:2,p] <-  se_prob[c(1,5),p] + sp_prob[c(1,5),p] +
                        se_prob[c(2,6),p] + sp_prob[c(2,6),p] +
                        se_prob[c(3,7),p] + sp_prob[c(3,7),p] +
                        se_prob[c(4,8),p] + sp_prob[c(4,8),p]
}
```

NB: `MMM` combinations have been removed!

```
## Observation probabilities:

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

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

```
# "covsp13" <- 0
# "covsp23" <- 0
}
inits{
"se" <- c(0.99, 0.5, 0.99)
"sp" <- c(0.75, 0.99, 0.75)
"prev" <- c(0.95, 0.05)
# "covse12" <- 0
# "covse13" <- 0
# "covse23" <- 0
# "covsp12" <- 0
# "covsp13" <- 0
# "covsp23" <- 0
}

## Data:
data{
"Populations" <- 2
```

How to form populations

Clearly valid strategies:

- Temporal and/or spatial separation (e.g. farms)
- Experimental separation (different blocks of a trial)
- Separation based on testing other individuals within the same cohort (e.g. historical data)

...

Clearly invalid strategies:

- Grouping based on the results of a diagnostic test being evaluated in the same individuals

...

Potentially OK but sometimes risky strategies:

Consistent sensitivity and specificity

Strategies to verify this:

- Eliminate one test at a time and re-run the model (if >=3 tests)
- Eliminate one population at a time and re-run the model (if >=3 populations, or >=2 populations with strong priors)
- Allow sensitivity or specificity to differ between populations (requires a lot of data) - see session 7

Making your data missing

If we have >2 populations *and* >2 tests then we can eliminate one combination at a time!

- This is a very useful form of cross-validation

Estimating the full model:

```
template_huiwalter(data, "model_full.txt")
results_full <- run.jags("model_full.txt")
## Loading required namespace: rjags
# Check convergence etc:
# plot(results_full)
# results_full

summary_full <- summary(results_full, vars="^s") %>%
  as.data.frame() %>%
  rownames_to_column("Parameter") %>%
  mutate(Model = "Full") %>%
  select(Model, Parameter, Median, Lower95, Upper95)
```

How can we make a specific population missing?

```
crossval_data <- data %>%
  filter(Population != 1)

template_huiwalter(crossval_data, "model_mp1.txt")
results_crossval <- run.jags("model_mp1.txt")
summary_crossval <- summary(results_crossval, vars="^s") %>%
  as.data.frame() %>%
  rownames_to_column("Parameter") %>%
  mutate(Model = "MP1") %>%
  select(Model, Parameter, Median, Lower95, Upper95) %>%
  bind_rows(summary_full) %>%
  arrange(Parameter, Model)
```

```
summary_crossval
##    Model Parameter    Median   Lower95   Upper95
## 1   Full     se[1] 0.8320092 0.7854857 0.8765929
## 2    MP1     se[1] 0.8245077 0.7637147 0.8798102
## 3   Full     se[2] 0.9029031 0.8586467 0.9457273
## 4    MP1     se[2] 0.8936085 0.8382836 0.9425320
## 5   Full     se[3] 0.9430374 0.9050877 0.9745947
## 6    MP1     se[3] 0.9420197 0.8977240 0.9806856
## 7   Full     sp[1] 0.9600054 0.9386068 0.9794505
## 8    MP1     sp[1] 0.9714057 0.9394112 0.9978597
## 9   Full     sp[2] 0.9887678 0.9742119 0.9999967
## 10   MP1     sp[2] 0.9770200 0.9485559 0.9999028
## 11  Full     sp[3] 0.9486042 0.9214019 0.9736189
## 12   MP1     sp[3] 0.9459068 0.8978384 0.9864053
```

How many combinations of test missingness and population do we have?

```
all_combinations <- data %>%
  pivot_longer(-Population, names_to = "Test", values_to = "Result") %>%
  filter(!is.na(Result)) %>%
  count(Population, Test) %>%
  print()
## # A tibble: 6 x 3
##   Population Test      n
##        <int> <chr> <int>
## 1          1 Test1   420
## 2          1 Test2   370
## 3          1 Test3   331
## 4          2 Test1   485
## 5          2 Test2   427
## 6          2 Test3   376
```

How can we make a specific combination of test and population missing?

```
all_results <- vector('list', length=nrow(all_combinations))
all_summary <- vector('list', length=nrow(all_combinations))

crossval_data <- data %>%
  mutate(Test1 = case_when(
    Population == 1 ~ NA_integer_,
    TRUE ~ Test1
  ))

template_huiwalter(crossval_data, "model_mc11.txt")
all_results[[1]] <- run.jags("model_mc11.txt")
# Assess convergence and sample size!
all_summary[[1]] <- summary(all_results[[1]], vars="^s") %>%
  as.data.frame() %>%
  rownames_to_column("Parameter") %>%
  mutate(Model = "MC11") %>%
  select(Model, Parameter, Median, Lower95, Upper95)
```

```r
crossval_data <- data %>%
  mutate(Test2 = case_when(
    Population == 1 ~ NA_integer_,
    TRUE ~ Test2
  ))

template_huiwalter(crossval_data, "model_mc12.txt")
all_results[[2]] <- run.jags("model_mc12.txt")
# Assess convergence and sample size!
all_summary[[2]] <- summary(all_results[[2]], vars="^s") %>%
  as.data.frame() %>%
  rownames_to_column("Parameter") %>%
  mutate(Model = "MC12") %>%
  select(Model, Parameter, Median, Lower95, Upper95)
```

```r
crossval_data <- data %>%
  mutate(Test2 = case_when(
    Population == 1 ~ NA_integer_,
    TRUE ~ Test2
  ))

template_huiwalter(crossval_data, "model_mc12.txt")
all_results[[2]] <- run.jags("model_mc12.txt")
# Assess convergence and sample size!
all_summary[[2]] <- summary(all_results[[2]], vars="^s") %>%
  as.data.frame() %>%
  rownames_to_column("Parameter") %>%
  mutate(Model = "MC12") %>%
  select(Model, Parameter, Median, Lower95, Upper95)
```

etc. . . !

Are there any substantial disagreements:

```
bind_rows(list(summary_full, all_summary)) %>% arrange(Parameter, Model)
```

```
##    Model Parameter    Median   Lower95   Upper95
## 1   Full     se[1] 0.8320092 0.7854857 0.8765929
## 2   MC11     se[1] 0.8206532 0.7627613 0.8763111
## 3   MC12     se[1] 0.8261458 0.7699415 0.8802758
## 4   Full     se[2] 0.9029031 0.8586467 0.9457273
## 5   MC11     se[2] 0.8947199 0.8400039 0.9438601
## 6   MC12     se[2] 0.8948672 0.8419727 0.9446560
## 7   Full     se[3] 0.9430374 0.9050877 0.9745947
## 8   MC11     se[3] 0.9458194 0.9065020 0.9794946
## 9   MC12     se[3] 0.9393193 0.8942116 0.9795340
## 10  Full     sp[1] 0.9600054 0.9386068 0.9794505
## 11  MC11     sp[1] 0.9706818 0.9381884 0.9967874
## 12  MC12     sp[1] 0.9648350 0.9384825 0.9909521
## 13  Full     sp[2] 0.9887678 0.9742119 0.9999967
## 14  MC11     sp[2] 0.9853411 0.9675813 0.9999857
## 15  MC12     sp[2] 0.9773488 0.9491652 0.9999992
## 16  Full     sp[3] 0.9486042 0.9214019 0.9736189
## 17  MC11     sp[3] 0.9501088 0.9170618 0.9814484
## 18  MC12     sp[3] 0.9528520 0.9180864 0.9858530
```

Practical session 6

Exercise 1

For this exercise you will need the 3-test, 3-population dataset provided as "anthrax.Rdata" under day 3. Here is what the data look like:

```
##         Population         PMB            AzureB
##  Population_A:136   Negative:556   Negative:558
##  Population_B:174   Positive:110   Positive:108
##  Population_C:356
##        qPCR
##  Negative:519
##  Positive:147
##
```

We have the result of 3 anthrax tests on cattle carcasses from 3 populations:

- PMB (polychrome methylene blue) is a stain used to help detect the capsule of anthrax bacteria on blood smears
- AzureB is a similar stain that is easier to perform in low

Summary

- Validation of model assumptions is essential but tricky
- Where we have 2 tests and 2 populations it is difficult to do anything other than biological justification
- Dropping one population/test at a time is a useful form of cross-validation if we have enough data
- Some further reading: Toft et al, STARD BLCM guidelines, covid paper for varying se across populations