Brussels Advanced training

Giles Innocent

2024-01-31

Introduction

- Building on the basic course.
- **NOT** lecture:practical.
- More intro: practical or discussion.
- You'll get more out of it if you try the exercises **before** looking at the sample code.
- Nothing Bayesian, or specific to diagnostic tests or latent class analyses (although it is particularly useful here)
- There is more than one way to code

Simulation

- Why simulate?
- How to simulate
    - within JAGS
    - R or equivalent
    - from an identical model to the analysis model
    - from a different model to the analysis

Simulating in R

- Functions like rbinom, rpois, rnorm, etc.
- All take a first parameter, n the number of data points you wish to simulate
- E.G. Hui-Walter paradigm:

```
set.seed(1)
n.sim <- 1
prev <- c(0.25, 0.8)
Se.1 <- Se.2 <- 0.8
Sp.1 <- Sp.2 <- 0.95
n.sampled <- c(100, 100)
test.results <- data.frame(pp=numeric(length(prev)),
                           pn=numeric(length(prev)),
                           np=numeric(length(prev)),
                           nn=numeric(length(prev)))
for(pop in 1:length(prev)){
  n.pos <- rbinom(n.sim,n.sampled[pop],prev[pop])
  test.results$pp[pop] <- rbinom(n.sim, n.pos, Se.1*Se.2
    rbinom(n.sim, n.sampled[pop]-n.pos, (1-Sp.1)*(1-Sp.2
```

JAGS code

```
cat(
"model{
  # Likelihood part:
  for (i in 1:n.pop) {
    p.test.result[1,i] <-prev[i]*Se[1]*Se[2] + (1-prev[i])*
    p.test.result[3,i] <-prev[i]*(1-Se[1])*Se[2] + (1-prev
    p.test.result[2,i] <-prev[i]*Se[1]*(1-Se[2]) + (1-prev
    p.test.result[4,i] <-prev[i]*(1-Se[1])*(1-Se[2]) + (1-p
    test.results[,i] ~dmulti(p.test.result[,i], n.sampled[i
  }




  # Prior part:
  for (pop in 1:n.pop)  {
    prev[pop] ~ dbeta(1,1)
  }
  for(test in 1:2)  {
```

Analysis

```
runjags.options(silent.jags=TRUE)
n.burnin <- n.sample <- 5000
results.jags <- run.jags('h-w.jags', n.chains=2, burnin=n.b
```

```
## Loading required namespace: rjags

## Warning: No initial values were provided - JAGS will use
## values for all chains

## Finished running the simulation
```

```
pander(summary(results.jags))
```

Table 1: Table continues below

	Lower95	Median	Upper95	Mean	SD	Mode
prev[1]	0.1109	0.2135	0.337	0.2162	0.05855	0.2111
prev[2]	0.6232	0.7577	0.8772	0.7553	0.06486	0.7679
Se[1]	0.6239	0.7402	0.8462	0.7398	0.05615	0.7397
Se[2]	0.7347	0.8466	0.9484	0.8446	0.05529	0.8526

Graphs

```r
plot(results.jags)
```

```
## Generating plots...
```

Exercise

- 3 tests; 1 population
- 7 parameters:
  - 3 test sensitivities
  - 3 test specificities
  - 1 prevalence
- 2^3 combinations: 7 df in the data
  - is this identifiable?
  - are the estimates unbiased?
  - what if prevalence is very low ∼1%?
  - even with 1000 individuals only ∼10 are positive
  - can't estimate Se well
  - does a biased estimate of Se bias our estimates of Sp and/or prevalence?

Example R code

```r
simulation.3.test <- function(prev,n.sampled,Se,Sp) {
  if((length(prev)!=1)&(length(n.sampled)!=1)&(length(Se)!=
    print("Error in parameters sent to simulation.3.test")
    stop()
  }
  n.pos <- rbinom(1,n.sampled,prev)
  n.neg <- n.sampled - n.pos
  test.1 <- c(rbinom(n.pos,1,Se[1]), rbinom(n.neg,1,(1-Sp[1
  test.2 <- c(rbinom(n.pos,1,Se[2]), rbinom(n.neg,1,(1-Sp[2
  test.3 <- c(rbinom(n.pos,1,Se[3]), rbinom(n.neg,1,(1-Sp[3
  test.results <- c(
    sum(test.1 & test.2 & test.3),
    sum(test.1 & test.2 & !test.3),
    sum(test.1 & !test.2 & test.3),
    sum(test.1 & !test.2 & !test.3),
    sum(!test.1 & test.2 & test.3),
    sum(!test.1 & test.2 & !test.3),
    sum(!test.1 & !test.2 & test.3),
```

Example R/JAGS code

```
cat(
"model{
  # Likelihood part:
  p.test.result[1] <-prev*Se[1]*Se[2]*Se[3] + (1-prev)*(1-S
  p.test.result[2] <-prev*Se[1]*Se[2]*(1-Se[3]) + (1-prev)*
  p.test.result[3] <-prev*Se[1]*(1-Se[2])*Se[3] + (1-prev)*
  p.test.result[4] <-prev*Se[1]*(1-Se[2])*(1-Se[3]) + (1-pr
  p.test.result[5] <-prev*(1-Se[1])*Se[2]*Se[3] + (1-prev)*
  p.test.result[6] <-prev*(1-Se[1])*Se[2]*(1-Se[3]) + (1-pr
  p.test.result[7] <-prev*(1-Se[1])*(1-Se[2])*Se[3] + (1-pr
  p.test.result[8] <-prev*(1-Se[1])*(1-Se[2])*(1-Se[3]) + (
  test.results ~dmulti(p.test.result, n.tested)

  # Prior part:
  prev ~ dbeta(1,1)
  for(test in 1:3)  {
    Se[test] ~dbeta(1,1)
    Sp[test] ~dbeta(1,1)
```

How good is our posterior?

- Eyeball posterior distribution
- Are the means (medians, modes?) close to the values used for the simulation
- If we wish to be more formal about this we would repeat the simulation-analysis cycle many (400+) times
  - this takes a long time, typically
  - which is a better (less biased) predictor: mean, median or mode
  - are the 95% Credible Intervals true 95% Confidence Intervals
  - varying the parameter values within sensible ranges

Multiple simulations

```
set.seed(1)
n.sim <- 1000
n.pop <- 2
comparison.df <- data.frame(prev = c(runif(n.sim, 0.05,
                            Se.1 = runif(n.sim, 0.65, 0.
                            Sp.1 = rbeta(n.sim, 21,1),
                            Se.2 = rbeta(n.sim, 17,5),
                            Se.2 = rbeta(n.sim, 19,3),
                            prev.1.median = numeric(n.si
                            prev.2.median = numeric(n.si
                            prev.1.mean = numeric(n.sim)
                            prev.2.mean = numeric(n.sim)
                            Se.1.lcl = numeric(n.sim),
                            Se.1.ucl = numeric(n.sim),
                            Se.2.lcl = numeric(n.sim),
                            Se.2.ucl = numeric(n.sim))
for (sim in 1:n.sim) {
  ## Simulate data
```

Summary

- What is our interest?
  - which summary is better?
  - how accurate is the summary?
  - can we eliminate certain values?
  - 95% CI?
- What is the context?
  - range of possible values
  - effect of priors
  - convergence
  - sample size (see Matt)
- What if we believe that a parameter comes from a distribution, not a point value?