

# Session 2

Basic Hui-Walter models

---

Matt Denwood

2021-06-28

# Hui-Walter Model

- A particular model formulation that was originally designed for evaluating diagnostic tests in the absence of a gold standard
- Not necessarily (or originally) Bayesian but often implemented using Bayesian MCMC
- But evaluating an imperfect test against another imperfect test is a bit like pulling a rabbit out of a hat
  - If we don't know the true disease status, how can we estimate sensitivity or specificity for either test?

# Model Specification

```
model{
  Tally ~ dmulti(prob, N)

  # Test1- Test2-
  prob[1] <- (prev * ((1-se[1])*(1-se[2]))) + ((1-prev) *
    ↪ ((sp[1])*(sp[2])))

  # Test1+ Test2-
  prob[2] <- (prev * ((se[1])*(1-se[2]))) + ((1-prev) *
    ↪ ((1-sp[1])*(sp[2])))

  # Test1- Test2+
  prob[3] <- (prev * ((1-se[1])*(se[2]))) + ((1-prev) *
    ↪ ((sp[1])*(1-sp[2])))
```

```

# Test1+ Test2+
  prob[4] <- (prev * ((se[1])*(se[2]))) + ((1-prev) *
    ↪ ((1-sp[1])*(1-sp[2])))

prev ~ dbeta(1, 1)
se[1] ~ dbeta(1, 1)
sp[1] ~ dbeta(1, 1)
se[2] ~ dbeta(1, 1)
sp[2] ~ dbeta(1, 1)

#data# Tally, N
#monitor# prev, prob, se, sp, deviance
#inits# prev, se, sp
}

```

```
twoXtwo <- matrix(c(48, 12, 4, 36), ncol=2, nrow=2)
twoXtwo
##      [,1] [,2]
## [1,]   48   4
## [2,]   12  36
```

```
library('runjags')
```

```
Tally <- as.numeric(twoXtwo)
N <- sum(Tally)
```

```
prev <- list(chain1=0.05, chain2=0.95)
se <- list(chain1=c(0.01,0.99), chain2=c(0.99,0.01))
sp <- list(chain1=c(0.01,0.99), chain2=c(0.99,0.01))

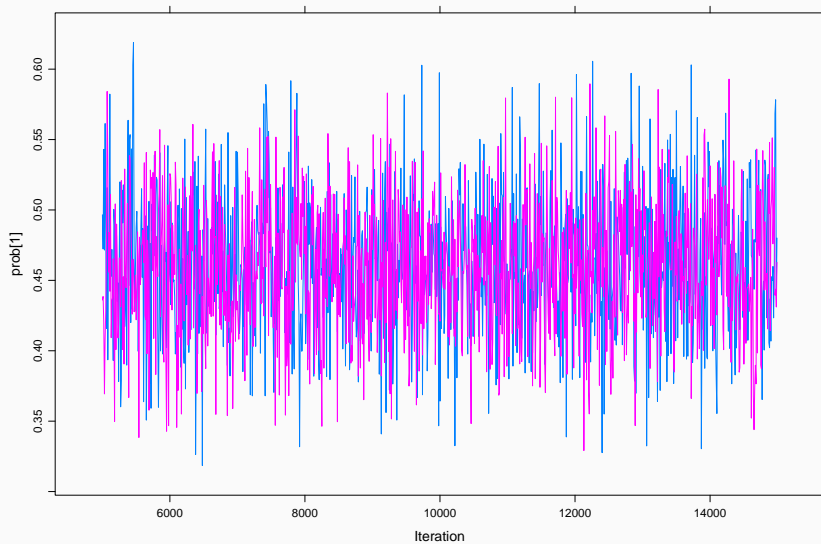
results <- run.jags('basic_hw.txt', n.chains=2)
```

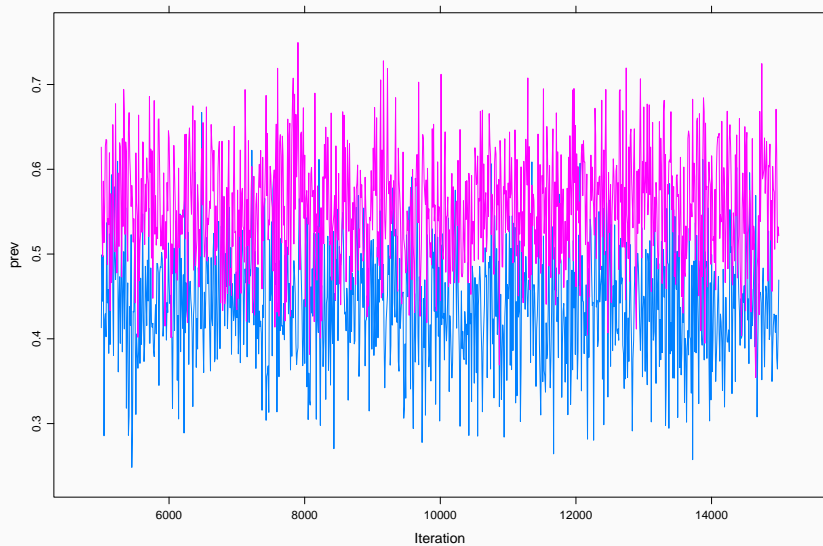
[Remember to check convergence and effective sample size!]

## results

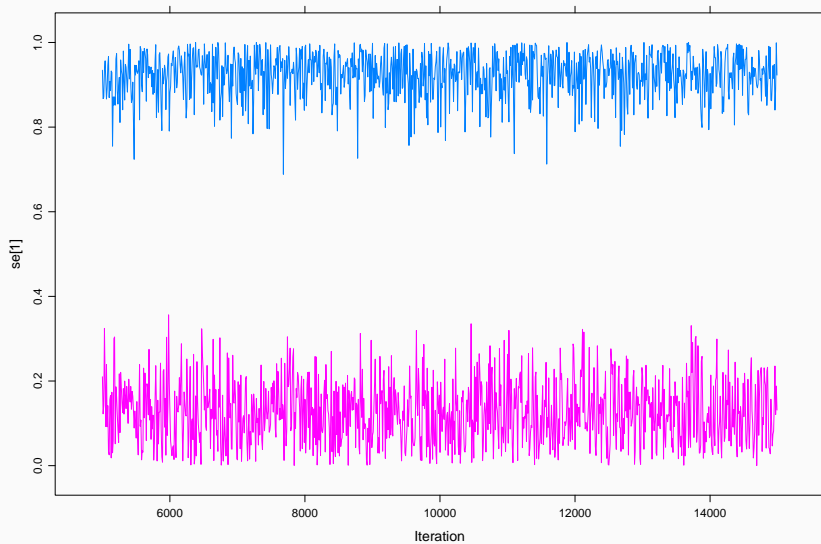
	Lower95	Median	Upper95	SSEff	psrf
prev	0.328	0.499	0.668	4250	2.304
prob[1]	0.367	0.462	0.558	13858	1.000
prob[2]	0.072	0.132	0.202	14200	1.000
prob[3]	0.018	0.055	0.104	9602	1.000
prob[4]	0.255	0.343	0.438	13555	1.000
se[1]	0.028	0.570	1.000	4564	15.155
se[2]	0.000	0.385	0.965	4585	13.619
sp[1]	0.000	0.461	0.970	4527	15.261
sp[2]	0.036	0.581	1.000	4593	13.641
deviance	12.304	15.134	21.447	8838	1.000

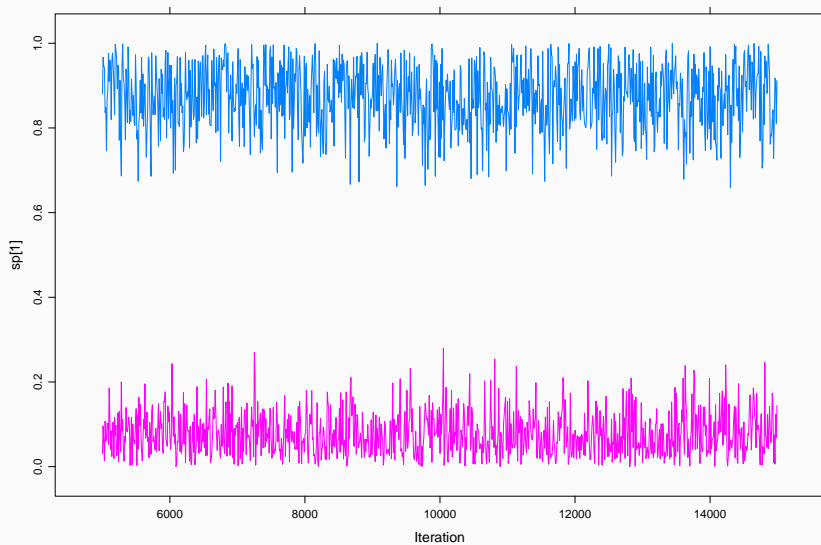
- Does anybody spot a problem?

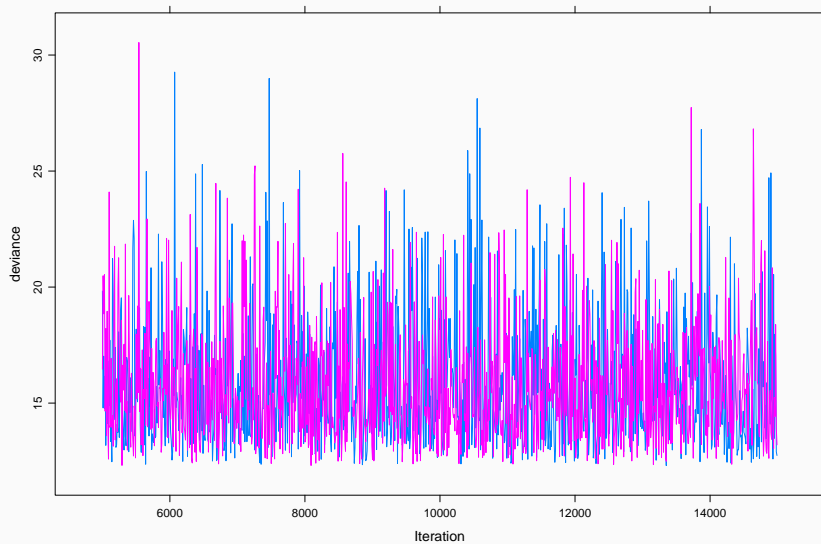


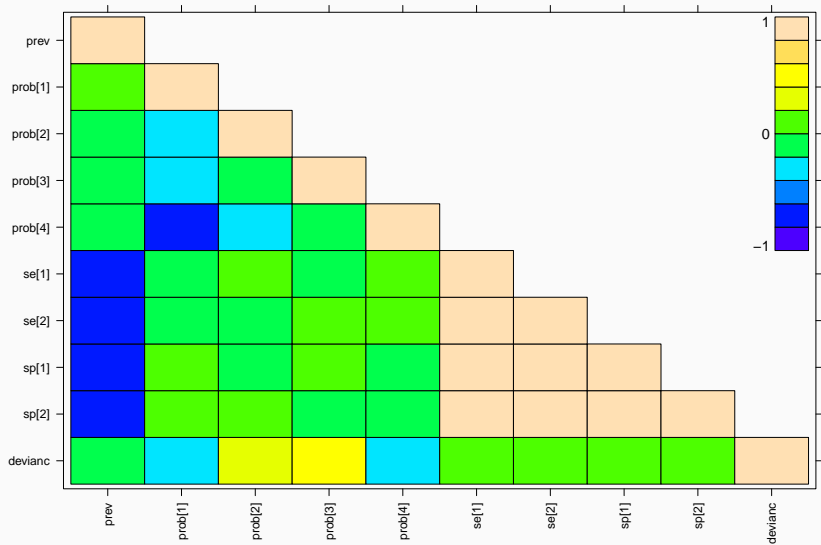












## Label Switching

How to interpret a test with  $Se=0\%$  and  $Sp=0\%$ ?

How to interpret a test with  $Se=0\%$  and  $Sp=0\%$ ?

- The test is perfect - we are just holding it upside down. . .

# Label Switching

How to interpret a test with  $Se=0\%$  and  $Sp=0\%$ ?

- The test is perfect - we are just holding it upside down...

We can force  $se+sp \geq 1$ :

```
se[1] ~ dbeta(1, 1)
sp[1] ~ dbeta(1, 1)T(1-se[1], )
```

Or:

```
se[1] ~ dbeta(1, 1)T(1-sp[1], )
sp[1] ~ dbeta(1, 1)
```

This allows the test to be useless, but not worse than useless.

Alternatively we can have the weakly informative priors:

```
se[1] ~ dbeta(2, 1)
```

```
sp[1] ~ dbeta(2, 1)
```

To give the model some information that we expect the test characteristics to be closer to 100% than 0%.



Alternatively we can have the weakly informative priors:

```
se[1] ~ dbeta(2, 1)
sp[1] ~ dbeta(2, 1)
```

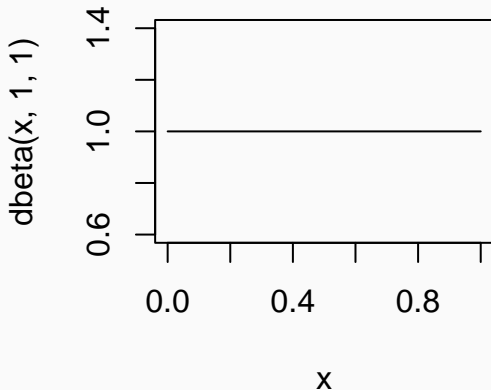
To give the model some information that we expect the test characteristics to be closer to 100% than 0%.

Or we can use stronger priors for one or both tests.

# Priors

A quick way to see the distribution of a prior:

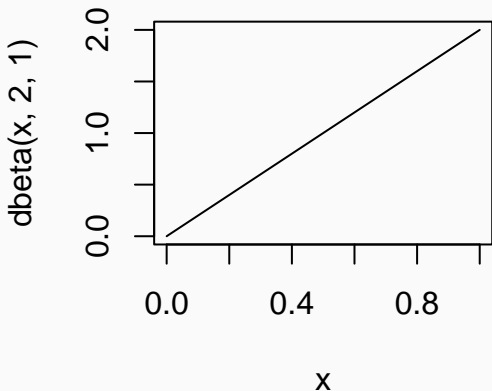
```
curve(dbeta(x, 1, 1), from=0, to=1)
```



```
qbeta(c(0.025,0.975), shape1=1, shape2=1)  
## [1] 0.025 0.975
```

This was minimally informative, but how does that compare to a weakly informative prior for e.g. sensitivity?

```
curve(dbeta(x, 2, 1), from=0, to=1)
```



```
qbeta(c(0.025,0.975), shape1=2, shape2=1)  
## [1] 0.1581139 0.9874209
```

```
qbeta(c(0.025,0.975), shape1=2, shape2=1)  
## [1] 0.1581139 0.9874209
```

Or more accurately:

```
library("TeachingDemos")  
hpd(qbeta, shape1=2, shape2=1)  
## [1] 0.2236068 1.0000000
```

```
qbeta(c(0.025,0.975), shape1=2, shape2=1)
## [1] 0.1581139 0.9874209
```

Or more accurately:

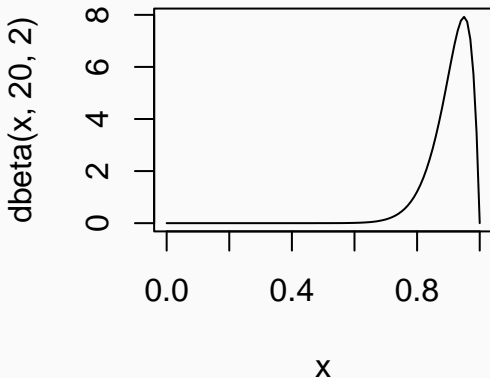
```
library("TeachingDemos")
hpd(qbeta, shape1=2, shape2=1)
## [1] 0.2236068 1.0000000
```

Credible vs confidence intervals:

- For MCMC these are usually calculated using highest posterior density (HPD) intervals
- Therefore there is a difference between:
  - `qbeta(c(0.025,0.975), ...)`
  - `hpd(qbeta, ...)`
- Technically HPD intervals are credible intervals. . .

What about a more informative prior?

```
curve(dbeta(x, 20, 2), from=0, to=1)
```



```
qbeta(c(0.025,0.975), shape1=20, shape2=2)
```

```
## [1] 0.7618401 0.9882507
```

```
hpd(qbeta, shape1=20, shape2=2)
```

```
## [1] 0.7919691 0.9973994
```

## Choosing a prior

What we want is e.g.  $\text{Beta}(20,1)$

But typically we have median and 95% confidence intervals from a paper, e.g.:

“The median (95% CI) estimates of the sensitivity and specificity of the shiny new test were 94% (92-96%) and 99% (97-100%) respectively”

## Choosing a prior

What we want is e.g.  $\text{Beta}(20,1)$

But typically we have median and 95% confidence intervals from a paper, e.g.:

“The median (95% CI) estimates of the sensitivity and specificity of the shiny new test were 94% (92-96%) and 99% (97-100%) respectively”

How can we generate a  $\text{Beta}( , )$  prior from this?

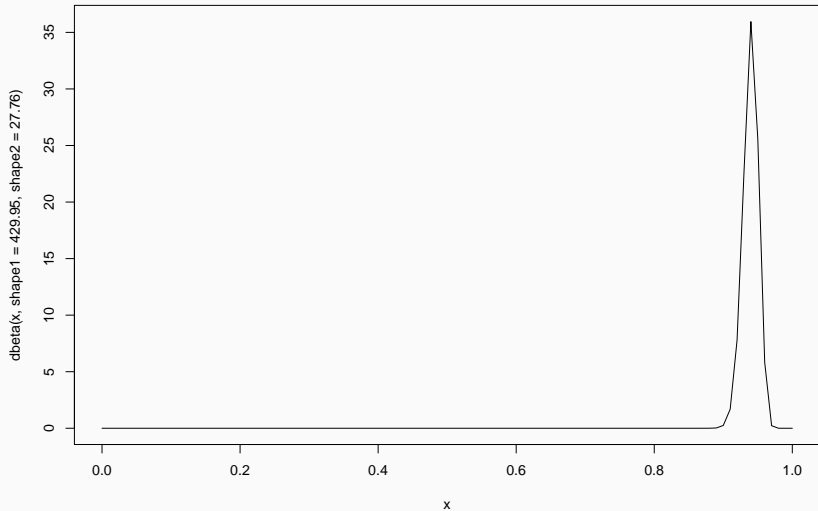


# The PriorGen package

"The median (95% CI) estimates of the sensitivity and specificity of the shiny new test were 94% (92-96%) and 99% (97-100%)"

```
library("PriorGen")
## Loading required package: rootSolve
findbeta(themedmedian = 0.94, percentile=0.95, percentile.value = 0.92)
## [1] "The desired Beta distribution that satisfies the specified
↳ conditions is: Beta( 429.95 27.76 )"
## [1] "Here is a plot of the specified distribution."
## [1] "Descriptive statistics for this distribution are:"
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.8910 0.9322 0.9399 0.9393 0.9473 0.9749
## [1] "Verification: The percentile value 0.92 corresponds to the 0.05
↳ th percentile"
hpd(qbeta, shape1=429.95, shape2=27.76)
## [1] 0.917172 0.960435
```

```
curve(dbeta(x, shape1=429.95, shape2=27.76))
```



## Initial values

Part of the problem before was also that we were specifying extreme initial values:

```
se <- list(chain1=c(0.01,0.99), chain2=c(0.99,0.01))  
sp <- list(chain1=c(0.01,0.99), chain2=c(0.99,0.01))
```

## Initial values

Part of the problem before was also that we were specifying extreme initial values:

```
se <- list(chain1=c(0.01,0.99), chain2=c(0.99,0.01))  
sp <- list(chain1=c(0.01,0.99), chain2=c(0.99,0.01))
```

Let's change these to:

```
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))
```

# Analysing simulated data

This is useful to check that we can recover parameter values!

```
# Set a random seed so that the data are reproducible:
set.seed(2021-06-28)

se <- c(0.9, 0.6)
sp <- c(0.95, 0.9)
N <- 1000
prevalence <- 0.5

data <- tibble(Status = rbinom(N, 1, prevalence)) %>%
  mutate(Test1 = rbinom(N, 1, se[1]*Status + (1-sp[1])*(1-Status))) %>%
  mutate(Test2 = rbinom(N, 1, se[2]*Status + (1-sp[2])*(1-Status)))

twoXtwo <- with(data, table(Test1, Test2))
Tally <- as.numeric(twoXtwo)
```

# Analysing simulated data

This is useful to check that we can recover parameter values!

```
# Set a random seed so that the data are reproducible:
set.seed(2021-06-28)

se <- c(0.9, 0.6)
sp <- c(0.95, 0.9)
N <- 1000
prevalence <- 0.5

data <- tibble(Status = rbinom(N, 1, prevalence)) %>%
  mutate(Test1 = rbinom(N, 1, se[1]*Status + (1-sp[1])*(1-Status))) %>%
  mutate(Test2 = rbinom(N, 1, se[2]*Status + (1-sp[2])*(1-Status)))

twoXtwo <- with(data, table(Test1, Test2))
Tally <- as.numeric(twoXtwo)
```

We know that e.g. the first test has Sensitivity of 90% and Specificity of 95% - so the model *should* be able to tell us that...

## **Practical Session 2**

---

## Points to consider

1. What is the typical autocorrelation (and therefore effective sample size) of Hui-Walter models compared to the simpler models we were running earlier? Is there any practical consequence of this?
2. How does changing the prior distributions for the  $\text{se}$  and  $\text{sp}$  of one test affect the inference for the other test parameters?



# Summary

- Hui-Walter models can do magical things, but:
  - They typically exhibit high autocorrelation
  - They may not converge
  - Need a larger sample for the same effective sample size
- More informative priors for one test will
  - Improve identifiability of the model
  - Impact the inference for all other parameters in the model!