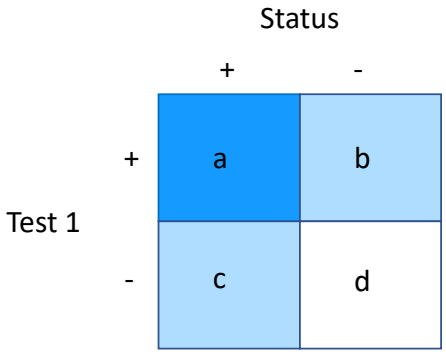
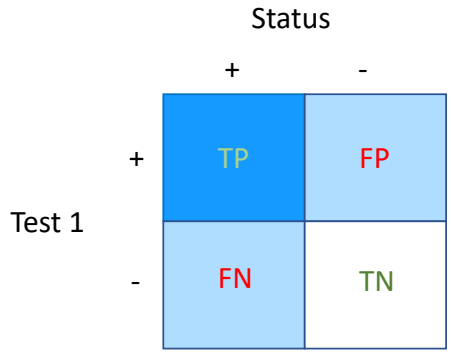
Day 1 Hui Walter I

Sonja Hartnack Valerie Hungerbühler

With some inspiration from Matt Denwood & Giles Innocent from a previous training

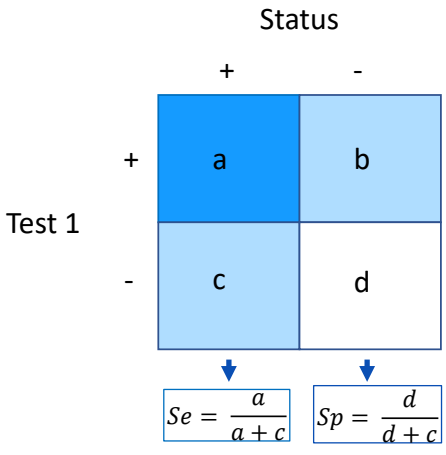
Recap on Se and Sp

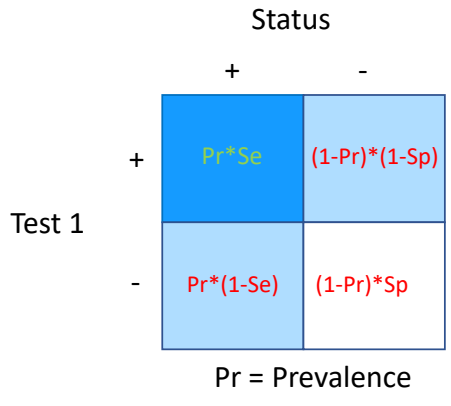




TP : true positive
FP : false negative

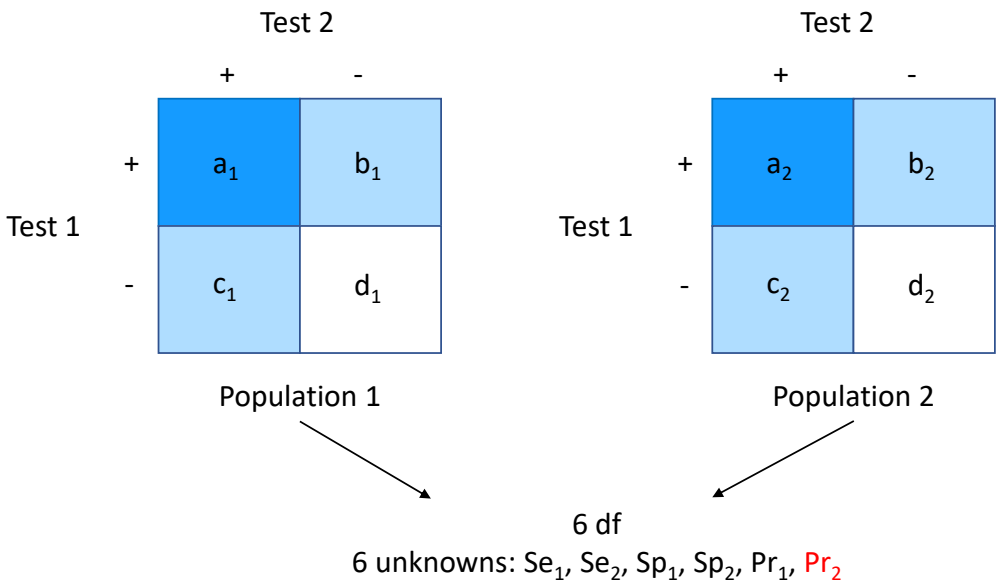
FN: false negative
TN: true negative





Hui-Walter model: a quick start

A particular model formulation that was originally designed for evaluating diagnostic tests in the absence of a gold standard



Hui-Walter model: a quick start

After Hui and Walter was named a paradigm, with S indicating the number of populations (P) and R the number of tests (T):

$$S \geq \frac{R}{(2^{\text{R-1}} - 1)}$$

If this condition is fulfilled, any combination of S and R may allow to estimate Se and Sp e.g. (2T, 2P), (3T,1P), (4T,1P), ...

- A Maximum Likelihood approach (ML) to estimate the values of the unknown parameters (sensitivities, specificities and prevalences) can be utilised.
- Here the value of the unknown parameters are estimated from the sample data such that the values chosen maximize the probability of obtaining the observed data.
- For each of the four cells in the 2x2 table, the likelihood contributions are determined as the probability of observing data in each cell conditional on the parameters, raised to the power of the observed frequency for that cell.

For example for two diagnostic tests named T_1 and T_2 the probabilities P of the four different options of binary test results $(++, +-, -+, --)$ could be modelled as follows:

$$P({T_1}^+, {T_2}^+) = [Pr \cdot Se_1 \cdot Se_2 + (1 - Pr) \cdot (1 - Sp_1) \cdot (1 - Sp_2)]^{a_i}$$
$$P({T_1}^+, {T_2}^-) = [Pr \cdot Se_1 \cdot (1 - Se_2) + (1 - Pr) \cdot (1 - Sp_1) \cdot Sp_2]^{b_i}$$
$$P({T_1}^-, {T_2}^+) = [Pr \cdot (1 - Se_1) \cdot Se_2 + (1 - Pr) \cdot Sp_1 \cdot (1 - Sp_2)]^{c_i}$$
$$P({T_1}^-, {T_2}^-) = [Pr \cdot (1 - Se_1) \cdot (1 - Se_2) + (1 - Pr) \cdot Sp_1 \cdot Sp_2]^{d_i}$$

- The likelihood function for the unknown parameters based on the overall data is obtained by multiplying the likelihood contributions across the i populations (or summing the log likelihood contributions) of the number of independently sampled populations.
- After the (log)likelihood function for the six unknown parameters has been formulated, first and second derivatives are needed to obtain the maximum likelihood estimates (MLEs) for the sensitivities, specificities and prevalences as well as the variance-covariance matrix.

Since an exact analytical solution will not exist, numerical iteration methods such as Newton-Raphson or Expectation-Maximisation (EM) algorithm are needed to estimate the six model parameters.

$$P({T_1}^+, {T_2}^+) = [Pr \cdot Se_1 \cdot Se_2 + (1 - Pr) \cdot (1 - Sp_1) \cdot (1 - Sp_2)]^{a_i}$$
$$P({T_1}^+, {T_2}^-) = [Pr \cdot Se_1 \cdot (1 - Se_2) + (1 - Pr) \cdot (1 - Sp_1) \cdot Sp_2]^{b_i}$$
$$P({T_1}^-, {T_2}^+) = [Pr \cdot (1 - Se_1) \cdot Se_2 + (1 - Pr) \cdot Sp_1 \cdot (1 - Sp_2)]^{c_i}$$
$$P({T_1}^-, {T_2}^-) = [Pr \cdot (1 - Se_1) \cdot (1 - Se_2) + (1 - Pr) \cdot Sp_1 \cdot Sp_2]^{d_i}$$

Hui-Walter model: a quick start

- Based on the following assumptions: (we will discuss them later)
  - The population is divided into two or more populations in which two or more tests are evaluated
  - sensitivity and specificity are the same in all populations
  - the tests are conditionally independent given the disease status.

Hui-Walter model: a quick start

- Based on the following assumptions: (we will discuss them later)
  - The population is divided into two or more populations in which two or more tests are evaluated
  - sensitivity and specificity are the same in all populations
  - the tests are conditionally independent given the disease status.
- Not necessarily (or originally) Bayesian but often implemented using Bayesian MCMC

Bayesian and MCMC

Bayesian statistics

Conditional probabilities

$Pr = P(D^+)$

$Se = P(T^+|D^+)$

$Sp = P(T^-|D^-)$

an application of Bayes theorem:

$$P(D^+|T^+) = \frac{P(T^+|D^+) \cdot P(D^+)}{P(T^+)}$$

$PPV = P(D^+|T^+)$

$NPV = P(D^-|T^-)$

Bayesian statistics

$$P(D^+|T^+) = \frac{P(T^+|D^+) \cdot P(D^+)}{P(T^+)}$$

or with Pr indicating the prevalence and thus relating the positive predictive value to the prevalence as well as to sensitivity and specificity

$$P(D^+|T^+) = \frac{Se \cdot Pr}{Se \cdot Pr + (1 - Sp)(1 - Pr)}$$

In the Bayesian perspective, prior information and the actual data (likelihood) considered together provide a posterior information.

Bayesian statistics

Bayes' theorem is at the heart of Bayesian statistics. This states that:

$$P(\theta|Y) = \frac{P(Y|\theta) \cdot P(\theta)}{P(Y)}$$

Where:

θ is our parameter value(s);

Y is the data that we have observed;

$P(\theta|Y)$ is the posterior probability of the parameter value(s) given the data and priors;

$P(\theta)$ is the prior probability of the parameters BEFORE we had observed the data;

$P(Y|\theta)$ is the likelihood of the data given the parameters value(s), as discussed above;

$P(Y)$ is the probability of the data, integrated over all parameter

Bayesian statistics

Note that $P(Y)$ is rarely calculable except in the simplest of cases, but is a constant for a given model. So in practice we usually work with the following:

$$P(\theta|Y) \propto P(Y|\theta) \cdot P(\theta)$$

- For frequentist statistics we only had the likelihood, but Bayesian statistics allows us to combine this likelihood with a prior to obtain a posterior.
- There are a lot of advantages of working with a posterior rather than a likelihood, because it allows us to make much more direct (and useful) inference about the probability of our parameters given the data.
- The cost of working with the posterior is that we must also define a prior, and accept that our posterior is affected by prior that we choose.

MCMC Markov Chain Monte Carlo

- BLCM rely on MCMC simulations.
- Markov chains designate random or stochastic processes that undergo transitions from one state to another on a state space.
- During numerous iterations, the chain will explore every point (or possible state) and will do so proportionally to its probability (ergodic).

MCMC Markov Chain Monte Carlo

- To be considered ergodic, a Markov chain must be
  - irreducible, meaning for every state there is a positive probability of moving to any other state
  - aperiodic, the chain mist not get trapped in cycles
  - reversible, when in equilibrium that rate at which the system moves from x to y is the same as from y to x.
- The initial or starting values are of minor importance, since after a number of steps (the burn-in phase), the Markov chain will eventually *converge* to a stationary distribution.
- To construct Markov chains, two main algorithms are used: Metropolis-Hastings and Gibbs. The Gibbs sampler is used in WinBUGS and JAGS (*Just Another Gibbs Sampler*).

MCMC in brief

- A way of obtaining a numerical approximation of the posterior
- Highly flexible
- Not inherently Bayesian but most widely used in this context
- Assessing convergence is essential, otherwise we may not be summarising the true posterior
- Our chains are correlated, so we need to consider the effective sample size

Model Specification Hui Walter

Called here 'basic_hw.bug'

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

# Test1+ Test2+
    prob[1] <- (prev * ((se[1])*(se[2])))
    + ((1-prev) * ((1-sp[1])*(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 * ((1-se[1])*(1-se[2])))
    + ((1-prev) * ((sp[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, TotalTests
  #monitor# prev, se, sp, prob
  #inits# prev, se, sp
}
```

```r
# Data set
twoXtwo <- matrix(c(48, 12, 4, 36), ncol=2, nrow=2)
colnames(twoXtwo) <- c("T1+", "T1-")
rownames(twoXtwo) <- c("T2+", "T2-")
twoXtwo
```

```
##      T1+ T1-
## T2+   48   4
## T2-   12  36
```

```
library('runjags')

Tally <- as.numeric(twoXtwo)
TotalTests <- sum(Tally)

prev <- list(chain1=0.05, chain2=0.95, chain3=0.99)
se <- list(chain1=c(0.5,0.99), chain2=c(0.99,0.5), chain3=c
sp <- list(chain1=c(0.5,0.99), chain2=c(0.99,0.5), chain3=c
set.seed(1104)
results <- run.jags('basic_hw.bug', n.chains=3)
```

	Lower95	Median	Upper95	SSeff	psrf
prev	0.425	0.560	0.689	6202	1
se[1]	0.861	0.947	1.000	8737	1
se[2]	0.746	0.876	1.000	5166	1
sp[1]	0.692	0.849	1.000	5076	1
sp[2]	0.823	0.933	1.000	8608	1
prob[1]	0.363	0.462	0.555	20058	1
prob[2]	0.073	0.132	0.201	20717	1
prob[3]	0.017	0.055	0.103	14368	1
prob[4]	0.255	0.344	0.438	21611	1

[Remember to check convergence and effective sample size!]

psrf: potential scale reduction factor

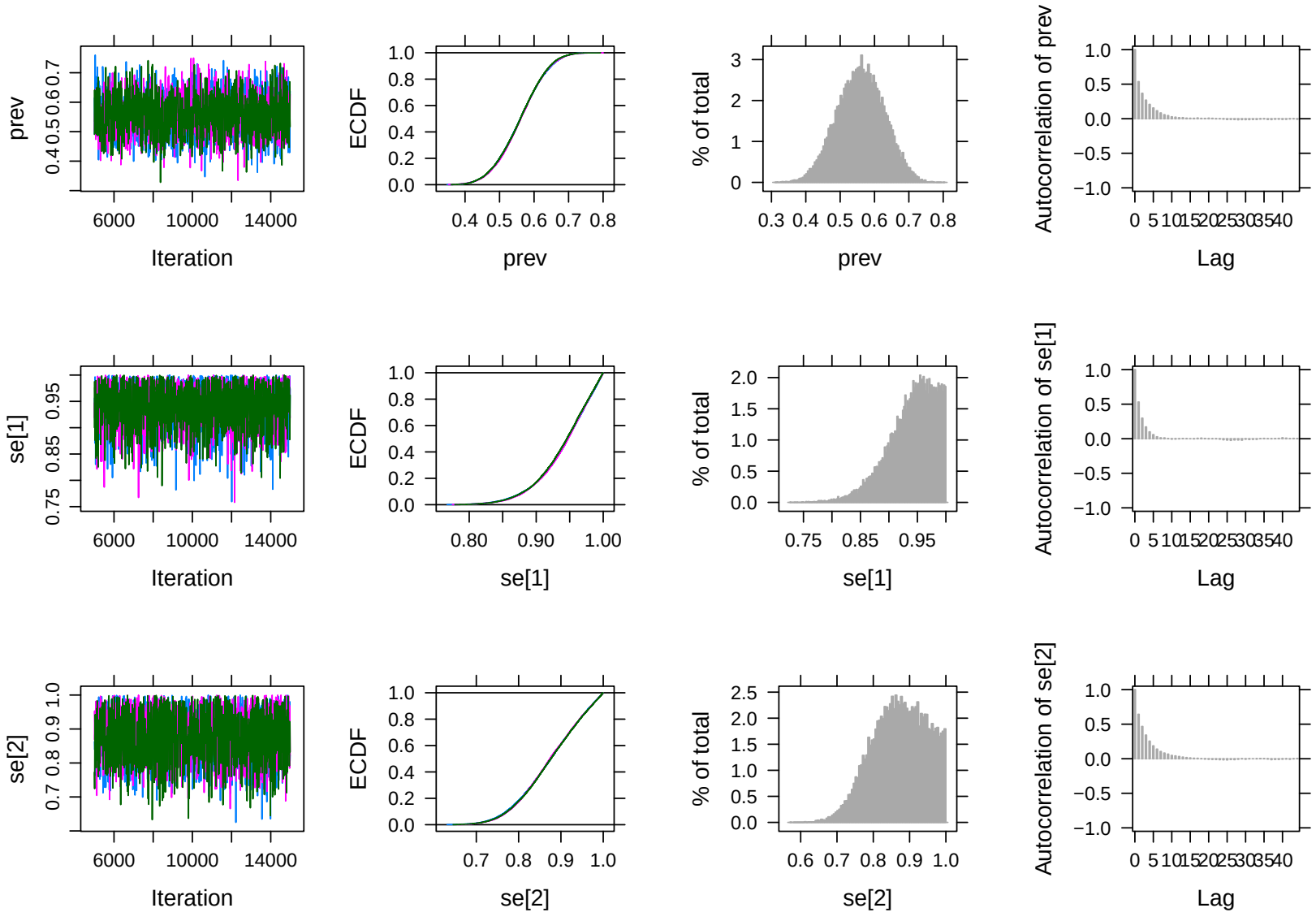
- The psrf or Gelman-Rubin statistic is an estimated factor by which the scale of the current distribution for the target distribution might be reduced if the simulations were continued for an infinite number of iterations.
- psrf estimates the potential decrease in the between-chains variability with respect to the within-chain variability.
- If psrf is large, then longer simulation sequences are expected to either decrease between-chains variability or increase within-chain variability because the simulations have not yet explored the full posterior distribution.
- If psrf < 1.1 for all model parameters, one can be fairly confident that convergence has been reached. Otherwise, longer chains or other means for improving the convergence may be needed.

SSeff: effective sample size

- Effective sample size is a calculation of the equivalent number of independent samples that would contain the same posterior accuracy as the correlated samples from an MCMC. Thus, MCMC Efficiency is the number of effectively independent samples generated per second.
- With high autocorrelation, the effective sample size will be lower.
- We want SSeff > 1000

Plotting

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



Exercises 1

- Can you reproduce the results?
- Check the run.jags() command and explore what happens if
  - the burn-in and/or the sample size is increased or decreased,
  - less or more iterations are run,
  - does a higher thinning than 1 affect the autocorrelation?
- Could you change the initial values? Does this have an effect?
- Could you add more chains?

Ex

Plotting

Practicalities

- These models need A LOT of data
  - And/or strong priors for one of the tests
- Convergence is more problematic than usual
- Check your results carefully to ensure they make sense!

What has happened here?

```
.RNG.name <- "lecuyer::RngStream"
.RNG.seed <- list(chain1=98765, chain2=33456)
```

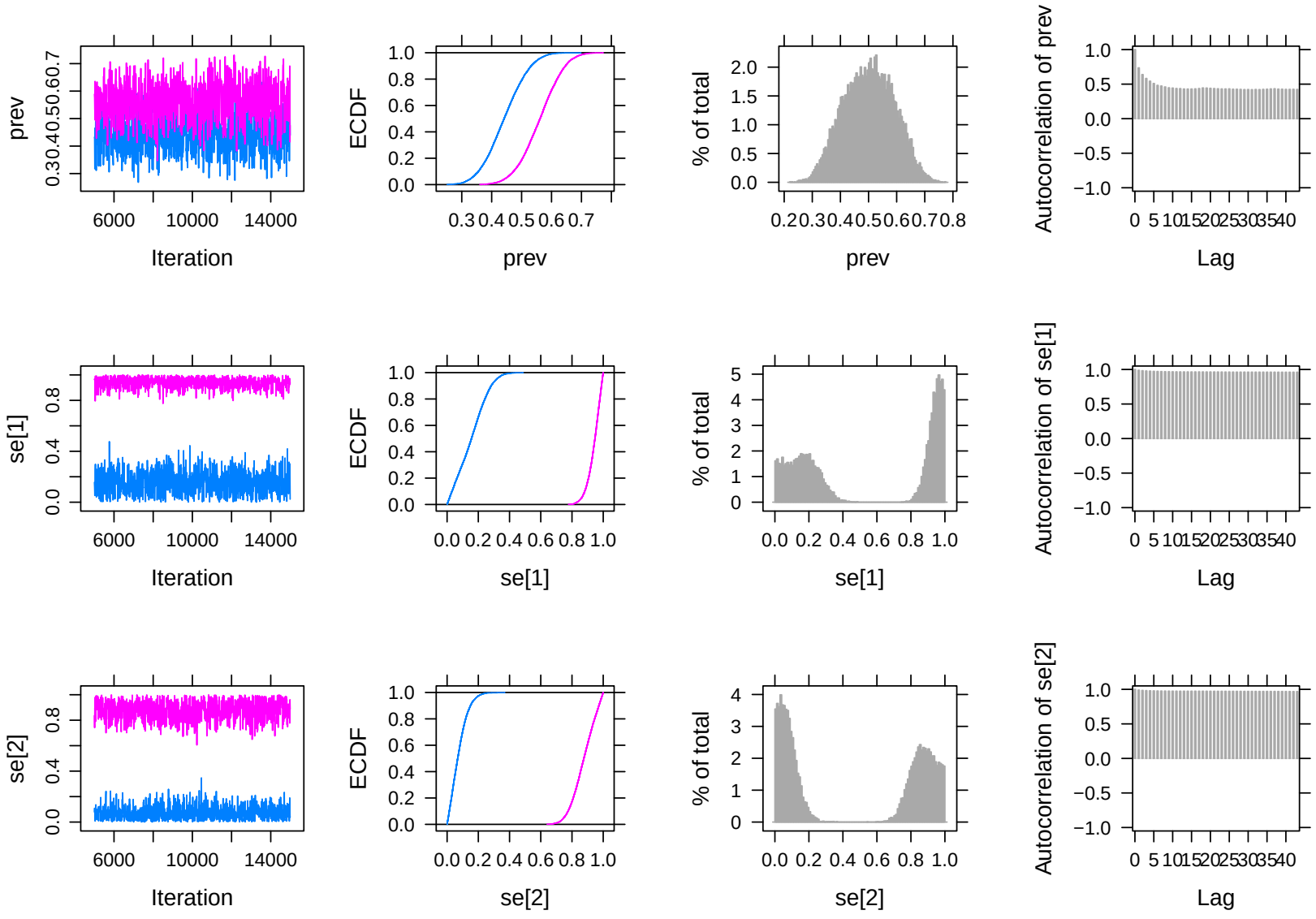
```
library('runjags')

Tally <- as.numeric(twoXtwo)
TotalTests <- sum(Tally)

prev <- list(chain1=0.05, chain2=0.95)
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))


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

	Lower95	Median	Upper95	SSeff	psrf
prev	0.329	0.503	0.664	4404	2.305
se[1]	0.031	0.616	1.000	4475	13.230
se[2]	0.000	0.457	0.972	4509	15.178
sp[1]	0.000	0.389	0.966	4672	13.510
sp[2]	0.026	0.522	1.000	4415	14.979
prob[1]	0.365	0.461	0.557	13493	1.000
prob[2]	0.071	0.133	0.201	13559	1.000
prob[3]	0.019	0.055	0.104	9470	1.000
prob[4]	0.258	0.344	0.439	13153	1.000

[Remember to check convergence and effective sample size!]

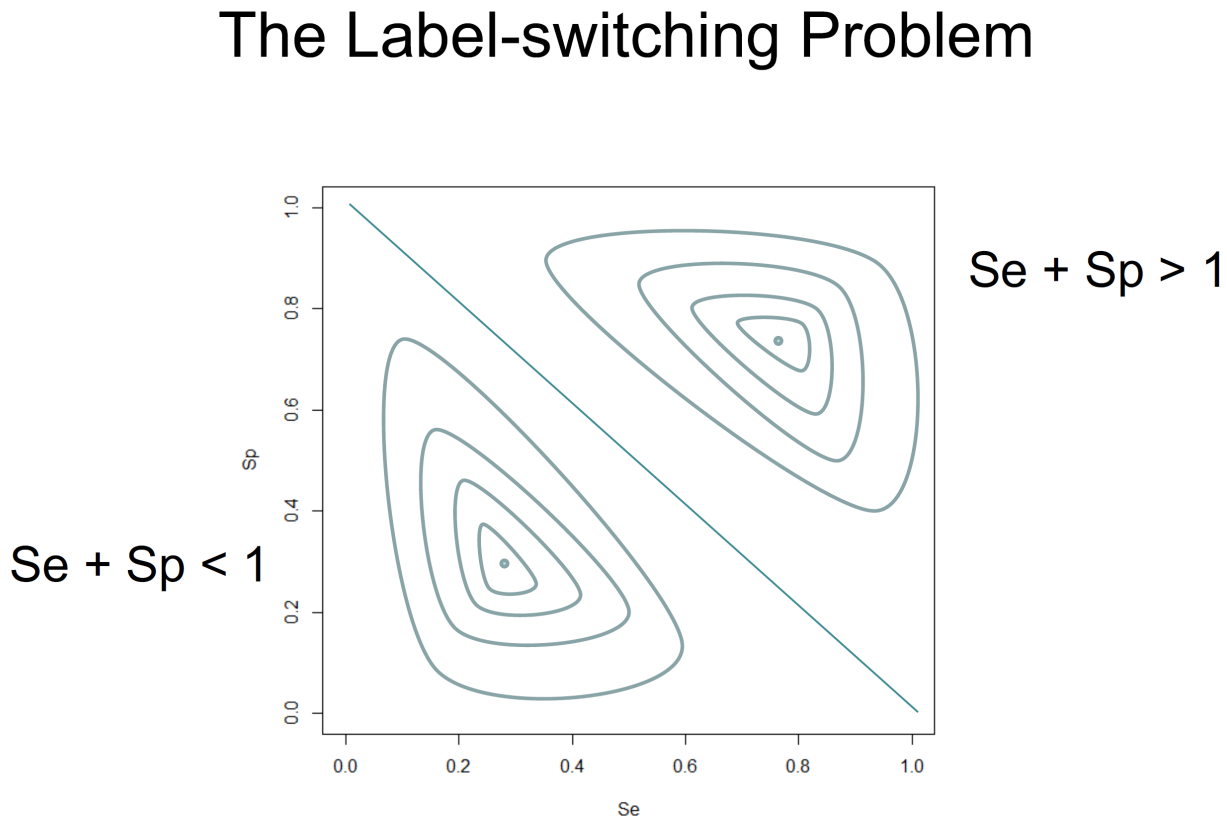
Plotting

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



Label Switching

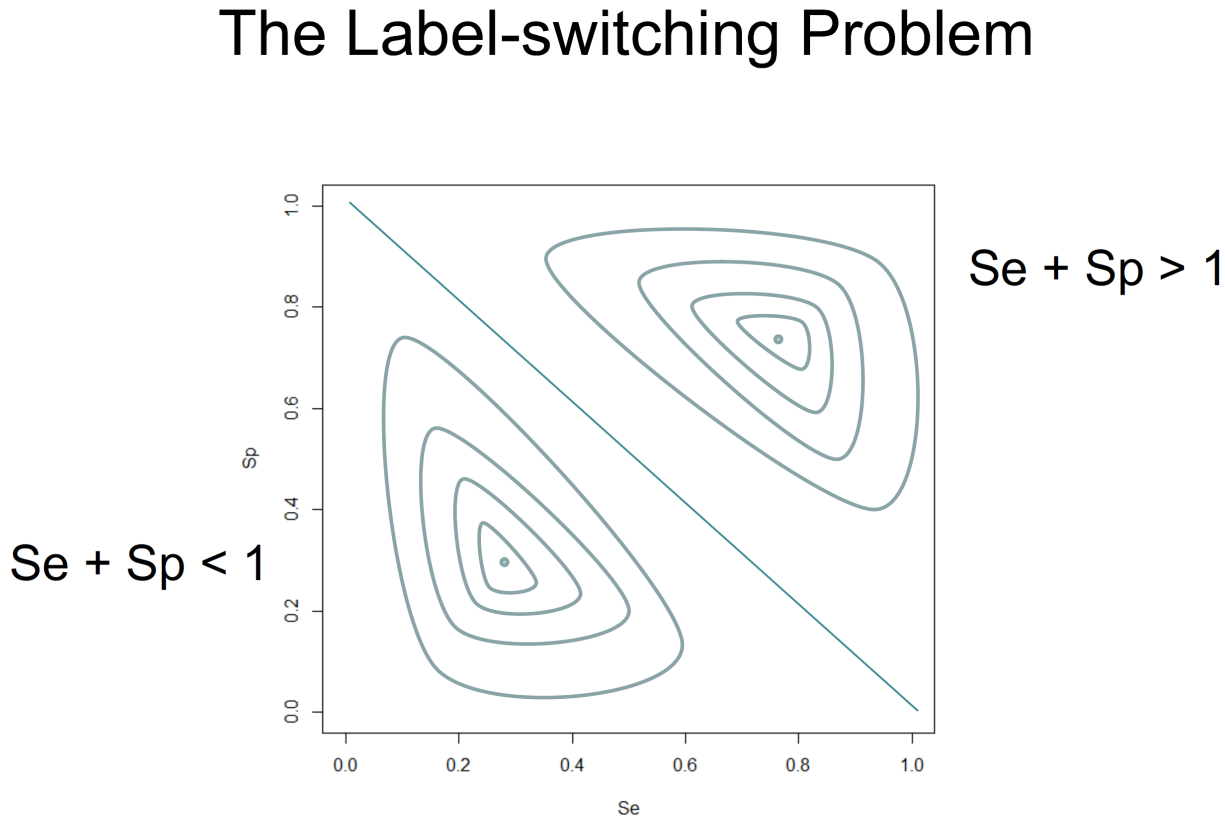
How to interpret a test with Se=0% and Sp=0%?



Label Switching

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

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



Avoid label switching

We can force se+sp >= 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)
```

But not both!

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

```
## [1] 10000     9
```

Ex 2

- Can you fix the problem with the label switching?