Supplementary material

Table S1: Outcomes of clinical examination vs three laboratory assays (virus isolation and two conventional PCRs) to identify the presence of BEFV in 88 clinically diseased South African cattle

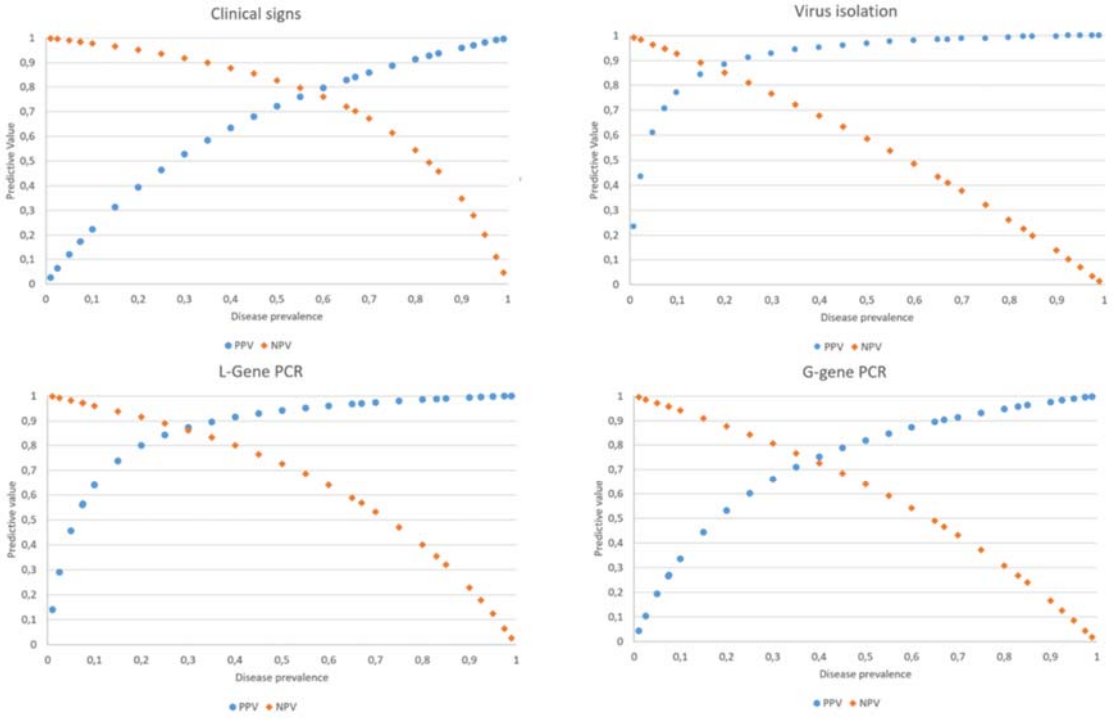
	Positive: Number (Percentage)	Negative: Number (Percentage
Clinical signs	69 (78%)	19 (22%)
Virus isolation	18 (20%)	70 (80%)
L-gene PCR	38 (43%)	50 (57%)
G420-gene PCR	29 (33%)	59 (67%)
Any laboratory assay	50 (57%)	38 (43%)
Either PCR	46 (42%)	42 (48%)

Table S2: STARD-Checklist

Section & Topic	No	Item	Reported on page #
TITLE OR ABSTRACT			
	1	Identification as a study of diagnostic accuracy using at least one measure of accuracy (such as sensitivity, specificity, predictive values, or AUC)	Title page
ABSTRACT			
	2	Structured summary of study design, methods, results, and conclusions (for specific guidance, see STARD for Abstracts)	1
INTRODUCTION			
	3	Scientific and clinical background, including the intended use and clinical role of the index test	2
	4	Study objectives and hypotheses	5
METHODS			
Study design	**5**	Whether data collection was planned before the index test and reference standard were performed (prospective study) or after (retrospective study)	6
Participants	**6**	Eligibility criteria	6 and 7
	7	On what basis potentially eligible participants were identified(such as symptoms, results from previous tests, inclusion in registry)	
	8	Where and when potentially eligible participants were identified (setting, location and dates)	
	9	Whether participants formed a consecutive, random or convenience series	
Test methods	**10a**	Index test, in sufficient detail to allow replication	8 to 10
	10b	Reference standard, in sufficient detail to allow replication	
	11	Rationale for choosing the reference standard (if alternatives exist)	
	12a	Definition of and rationale for test positivity cut-offs or result categories of the index test, distinguishing pre-specified from exploratory	
	12b	Definition of and rationale for test positivity cut-offs or result categories of the reference standard, distinguishing pre-specified from exploratory	
	13a	Whether clinical information and reference standard results were available to the performers/readers of the index test	
	13b	Whether clinical information and index test results were available to the assessors of the reference standard	
Analysis	**14**	Methods for estimating or comparing measures of diagnostic accuracy	11 to 13
	15	How indeterminate index test or reference standard results were handled	
	16	How missing data on the index test and reference standard were handled	
	17	Any analyses of variability in diagnostic accuracy, distinguishing pre-specified from exploratory	
	18	Intended sample size and how it was determined	
RESULTS			
Participants	**19**	Flow of participants, using a diagram	NA
	20	Baseline demographic and clinical characteristics of participants	13
	21a	Distribution of severity of disease in those with the target condition	NA
	21b	Distribution of alternative diagnoses in those without the target condition	NA
	22	Time interval and any clinical interventions between index test and reference standard	NA
Test results	**23**	Cross tabulation of the index test results (or their distribution) by the results of the reference standard	Table 4
	24	Estimates of diagnostic accuracy and their precision (such as 95% confidence intervals)	Table 5

Section & Topic	No	Item	Reported on page #
	25	Any adverse events from performing the index test or the reference standard	NA
DISCUSSION			
	26	Study limitations, including sources of potential bias, statistical uncertainty, and generalisability	21 and 22
	27	Implications for practice, including the intended use and clinical role of the index test	22 and 23
OTHER INFORMATION			
	28	Registration number and name of registry	NA
	29	Where the full study protocol can be accessed	NA
	30	Sources of funding and other support; role of funders	Title page

Figure S1



Positive predictive value and negative predictive value graphs for different theoretical disease prevalences for the four assays included in this analysis, namely examination of clinical signs, virus isolation, L-gene PCR and G-gene PCR.

PPV (Blue dots): Positive predictive value

NPV (orange diamonds): Negative predictive value

Addendum A: WinBUGs Code

4 Tests, 1 population; Conditional dependence between PCR tests

```
model;
{

# define multinomial distribution

        # x array denotes cell frequency data
        # total is the number of animals sampled
        # p is the probability estimate for multinomial cell
        # K is the number of species

                for (k in 1:K) {
                        x[k, 1:16] ~ dmulti (p[k, 1:16], total[k])
                }


# Main probability calculations
        # 0 is coded test negative
        # 1 is coded test positive
        # Test order:  CS -  VI  -  LG  -  GG

                        # define acceptable range for conditional covariance--positive (Se)

                        # alpha[1] <- max(-(1-Se[1])*(1-Se[2]), -Se[1]*Se[2])
                        # beta[1] <- min(Se[1]*(1-Se[2]),Se[2]*(1-Se[1]))
                        # alpha[2] <- max(-(1-Se[1])*(1-Se[3]), -Se[1]*Se[3])
                        # beta[2] <- min(Se[1]*(1-Se[3]),Se[3]*(1-Se[1]))
                        # alpha[3] <- max(-(1-Se[1])*(1-Se[4]), -Se[1]*Se[4])
                        # beta[3] <- min(Se[1]*(1-Se[4]),Se[4]*(1-Se[1]))
                        # alpha[4] <- max(-(1-Se[2])*(1-Se[3]), -Se[2]*Se[3])
                        # beta[4] <- min(Se[2]*(1-Se[3]),Se[3]*(1-Se[2]))
                        # alpha[5] <- max(-(1-Se[2])*(1-Se[4]), -Se[2]*Se[4])
                        # beta[5] <- min(Se[2]*(1-Se[4]),Se[4]*(1-Se[2]))
                        alpha[6] <- max(-(1-Se[3])*(1-Se[4]), -Se[3]*Se[4])
                        beta[6] <- min(Se[3]*(1-Se[4]),Se[4]*(1-Se[3]))


                        # define conditional covariance positive

                        CSxVISe <- 0
                        CSxLGSe <- 0
                        CSxGGSe <- 0
                        VIxLGSe <- 0
                        VIxGGSe <- 0
                        # LGxGGSe <- 0
                        LGxGGSe ~ dunif(alpha[6], beta[6])

                        # define acceptable range for conditional covariance--negative (Sp)

                        # gamma[1] <- max(-(1-Sp[1])*(1-Sp[2]), -Sp[1]*Sp[2])
                        # delta[1] <- min(Sp[1]*(1-Sp[2]),Sp[2]*(1-Sp[1]))
                        # gamma[2] <- max(-(1-Sp[1])*(1-Sp[3]), -Sp[1]*Sp[3])
                        # delta[2] <- min(Sp[1]*(1-Sp[3]),Sp[3]*(1-Sp[1]))
                        # gamma[3] <- max(-(1-Sp[1])*(1-Sp[4]), -Sp[1]*Sp[4])
                        # delta[3] <- min(Sp[1]*(1-Sp[4]),Sp[4]*(1-Sp[1]))
                        # gamma[4] <- max(-(1-Sp[2])*(1-Sp[3]), -Sp[2]*Sp[3])
                        # delta[4] <- min(Sp[2]*(1-Sp[3]),Sp[3]*(1-Sp[2]))
                        # gamma[5] <- max(-(1-Sp[2])*(1-Sp[4]), -Sp[2]*Sp[4])
                        # delta[5] <- min(Sp[2]*(1-Sp[4]),Sp[4]*(1-Sp[2]))
                        gamma[6] <- max(-(1-Sp[3])*(1-Sp[4]), -Sp[3]*Sp[4])
                        delta[6] <- min(Sp[3]*(1-Sp[4]),Sp[4]*(1-Sp[3]))


                        # define conditional covariance negative

                        CSxVISp <- 0
                        CSxLGSp <- 0
                        CSxGGSp <- 0
```

```
        VIxLGSp <- 0
        VIxGGSp <- 0
        # LGxGGSp <- 0
        LGxGGSp ~ dunif(gamma[6], delta[6])

for (k in 1:K) {

        p[k,1] <-  # 0 0 0 0 test pattern

                   # Probability due to diseased animals

                   ((prev[k])*(1-Se[1])*(1-Se[2])*(1-Se[3])*(1-Se[4]))+

                   # Probability due to test positive dependence

                   (prev[k]*(CSxVISe+CSxLGSe+CSxGGSe+
                   VIxLGSe+VIxGGSe+LGxGGSe))+

                   # Probability due to non-diseased animals

                   ((1-prev[k])*Sp[1]*Sp[2]*Sp[3]*Sp[4])+

                   # Probaility due to test negative dependence

                   ((1-prev[k])*(CSxVISp+CSxLGSp+CSxGGSp+
                   VIxLGSp+VIxGGSp+LGxGGSp))

        p[k,2] <-  # 0 0 01 test pattern

                   ((prev[k])*(1-Se[1])*(1-Se[2])*(1-Se[3])*(Se[4]))+

                   (prev[k]*(CSxVISe+CSxLGSe-CSxGGSe+
                   VIxLGSe-VIxGGSe-LGxGGSe))+

                   ((1-prev[k])*Sp[1]*Sp[2]*Sp[3]*(1-Sp[4]))+

                   ((1-prev[k])*(CSxVISp+CSxLGSp-CSxGGSp+
                   VIxLGSp-VIxGGSp-LGxGGSp))

        p[k,3] <-  # 0 01 0 test pattern

                   ((prev[k])*(1-Se[1])*(1-Se[2])*(Se[3])*(1-Se[4]))+

                   (prev[k]*(CSxVISe-CSxLGSe+CSxGGSe-
                   VIxLGSe+VIxGGSe-LGxGGSe))+

                   ((1-prev[k])*Sp[1]*Sp[2]*(1-Sp[3])*Sp[4])+

                   ((1-prev[k])*(CSxVISp-CSxLGSp+CSxGGSp-
                   VIxLGSp+VIxGGSp-LGxGGSp))

        p[k,4] <-  # 0 011 test pattern

                   ((prev[k])*(1-Se[1])*(1-Se[2])*(Se[3])*(Se[4]))+

                   (prev[k]*(CSxVISe-CSxLGSe-CSxGGSe-
                   VIxLGSe-VIxGGSe+LGxGGSe))+

                   ((1-prev[k])*Sp[1]*Sp[2]*(1-Sp[3])*(1-Sp[4]))+

                   ((1-prev[k])*(CSxVISp-CSxLGSp-CSxGGSp-
                   VIxLGSp-VIxGGSp+LGxGGSp))

        p[k, 5] <- # 01 0 0 test pattern

                   ((prev[k])*(1-Se[1])*(Se[2])*(1-Se[3])*(1-Se[4]))+

                   (prev[k]*(-CSxVISe+CSxLGSe+CSxGGSe-
                   VIxLGSe-VIxGGSe+LGxGGSe))+

                   ((1-prev[k])*Sp[1]*(1-Sp[2])*Sp[3]*Sp[4])+

                   ((1-prev[k])*(-CSxVISp+CSxLGSp+CSxGGSp-
                   VIxLGSp-VIxGGSp+LGxGGSp))
```

```
p[k, 6] <-  # 01 01 test pattern

            ((prev[k])*(1-Se[1])*(Se[2])*(1-Se[3])*(Se[4]))+

            (prev[k]*(-CSxVISe+CSxLGSe-CSxGGSe-
            VIxLGSe+VIxGGSe-LGxGGSe))+

            ((1-prev[k])*Sp[1]*(1-Sp[2])*Sp[3]*(1-Sp[4]))+

            ((1-prev[k])*(-CSxVISp+CSxLGSp-CSxGGSp-
            VIxLGSp+VIxGGSp-LGxGGSp))

p[k, 7] <-  # 011 0 test pattern

            ((prev[k])*(1-Se[1])*(Se[2])*(Se[3])*(1-Se[4]))+

            (prev[k]*(-CSxVISe-CSxLGSe+CSxGGSe+
            VIxLGSe-VIxGGSe-LGxGGSe))+

            ((1-prev[k])*Sp[1]*(1-Sp[2])*(1-Sp[3])*Sp[4])+

            ((1-prev[k])*(-CSxVISp-CSxLGSp+CSxGGSp+
            VIxLGSp-VIxGGSp-LGxGGSp))

p[k, 8] <-  # 0111 test pattern

            ((prev[k])*(1-Se[1])*(Se[2])*(Se[3])*(Se[4]))+

            (prev[k]*(-CSxVISe-CSxLGSe-CSxGGSe+
            VIxLGSe+VIxGGSe+LGxGGSe))+

            ((1-prev[k])*Sp[1]*(1-Sp[2])*(1-Sp[3])*(1-Sp[4]))+

            ((1-prev[k])*(-CSxVISp-CSxLGSp-CSxGGSp+
            VIxLGSp+VIxGGSp+LGxGGSp))

p[k, 9] <-  #1 0 0 0 test pattern

            ((prev[k])*(Se[1])*(1-Se[2])*(1-Se[3])*(1-Se[4]))+

            (prev[k]*(-CSxVISe-CSxLGSe-CSxGGSe+
            VIxLGSe+VIxGGSe+LGxGGSe))+

            ((1-prev[k])*(1-Sp[1])*Sp[2]*Sp[3]*Sp[4])+

            ((1-prev[k])*(-CSxVISp-CSxLGSp-CSxGGSp+
            VIxLGSp+VIxGGSp+LGxGGSp))

p[k,10] <- #1 0 01 test pattern

            ((prev[k])*(Se[1])*(1-Se[2])*(1-Se[3])*(Se[4]))+

            (prev[k]*(-CSxVISe-CSxLGSe+CSxGGSe+
            VIxLGSe-VIxGGSe-LGxGGSe))+

            ((1-prev[k])*(1-Sp[1])*Sp[2]*Sp[3]*(1-Sp[4]))+

            ((1-prev[k])*(-CSxVISp-CSxLGSp+CSxGGSp+
            VIxLGSp-VIxGGSp-LGxGGSp))

p[k,11] <- #1 01 0 test pattern

            ((prev[k])*(Se[1])*(1-Se[2])*(Se[3])*(1-Se[4]))+

            (prev[k]*(-CSxVISe+CSxLGSe-CSxGGSe-
            VIxLGSe+VIxGGSe-LGxGGSe))+

            ((1-prev[k])*(1-Sp[1])*Sp[2]*(1-Sp[3])*Sp[4])+

            ((1-prev[k])*(-CSxVISp+CSxLGSp-CSxGGSp-
            VIxLGSp+VIxGGSp-LGxGGSp))

p[k,12] <- #1 011 test pattern

            ((prev[k])*(Se[1])*(1-Se[2])*(Se[3])*(Se[4]))+
```

```
                (prev[k]*(-CSxVISe+CSxLGSe+CSxGGSe-
                VIxLGSe-VIxGGSe+LGxGGSe))+

                ((1-prev[k])*(1-Sp[1])*Sp[2]*(1-Sp[3])*(1-Sp[4]))+

                ((1-prev[k])*(-CSxVISp+CSxLGSp+CSxGGSp-
                VIxLGSp-VIxGGSp+LGxGGSp))

        p[k,13] <- #11 0 0 test pattern

                ((prev[k])*(Se[1])*(Se[2])*(1-Se[3])*(1-Se[4]))+

                (prev[k]*(CSxVISe-CSxLGSe-CSxGGSe-
                VIxLGSe-VIxGGSe+LGxGGSe))+

                ((1-prev[k])*(1-Sp[1])*(1-Sp[2])*Sp[3]*Sp[4])+

                ((1-prev[k])*(CSxVISp-CSxLGSp-CSxGGSp-
                VIxLGSp-VIxGGSp+LGxGGSp))

        p[k,14] <- #11 01 test pattern

                ((prev[k])*(Se[1])*(Se[2])*(1-Se[3])*(Se[4]))+

                (prev[k]*(CSxVISe-CSxLGSe+CSxGGSe-
                VIxLGSe+VIxGGSe-LGxGGSe))+

                ((1-prev[k])*(1-Sp[1])*(1-Sp[2])*Sp[3]*(1-Sp[4]))+

                ((1-prev[k])*(CSxVISp-CSxLGSp+CSxGGSp-
                VIxLGSp+VIxGGSp-LGxGGSp))

        p[k,15] <- #111 0 test pattern

                ((prev[k])*(Se[1])*(Se[2])*(Se[3])*(1-Se[4]))+

                (prev[k]*(CSxVISe+CSxLGSe-CSxGGSe+
                VIxLGSe-VIxGGSe-LGxGGSe))+

                ((1-prev[k])*(1-Sp[1])*(1-Sp[2])*(1-Sp[3])*Sp[4])+

                ((1-prev[k])*(CSxVISp+CSxLGSp-CSxGGSp+
                VIxLGSp-VIxGGSp-LGxGGSp))

        p[k,16] <- #1111 test pattern

                ((prev[k])*(Se[1])*(Se[2])*(Se[3])*(Se[4]))+

                (prev[k]*(CSxVISe+CSxLGSe+CSxGGSe+
                VIxLGSe+VIxGGSe+LGxGGSe))+

                ((1-prev[k])*(1-Sp[1])*(1-Sp[2])*(1-Sp[3])*(1-Sp[4]))+

                ((1-prev[k])*(CSxVISp+CSxLGSp+CSxGGSp+
                VIxLGSp+VIxGGSp+LGxGGSp))

        PPV[1] <- ((Se[1]*prev[k])/((Se[1]*prev[k])+(1-Sp[1])*(1-prev[k])))
        NPV[1] <- ((Sp[1]*(1-prev[k]))/((Sp[1]*(1-prev[k]))+((1-Se[1])*prev[k])))

        PPV[2] <- ((Se[2]*prev[k])/((Se[2]*prev[k])+(1-Sp[2])*(1-prev[k])))
        NPV[2] <- ((Sp[2]*(1-prev[k]))/((Sp[2]*(1-prev[k]))+((1-Se[2])*prev[k])))

        PPV[3] <- ((Se[3]*prev[k])/((Se[3]*prev[k])+(1-Sp[3])*(1-prev[k])))
        NPV[3] <- ((Sp[3]*(1-prev[k]))/((Sp[3]*(1-prev[k]))+((1-Se[3])*prev[k])))

        PPV[4] <- ((Se[4]*prev[k])/((Se[4]*prev[k])+(1-Sp[4])*(1-prev[k])))
        NPV[4] <- ((Sp[4]*(1-prev[k]))/((Sp[4]*(1-prev[k]))+((1-Se[4])*prev[k])))
        }

# prior probability distributions

prev[1] ~ dbeta (1,1)           # mode =
Se[1] ~ dbeta (20,10)                   # mode =
```

```
Se[2] ~ dbeta (4.5,11)                                  # mode =
Se[3] ~ dbeta (22,8)                                 # mode =
Se[4] ~ dbeta (13.5,17)                                 # mode =
Sp[1] ~ dbeta (21,8.5)                                 # mode =
Sp[2] ~ dbeta (99,1)                                # mode =
Sp[3] ~ dbeta (11,1.5)                                 # mode =
Sp[4] ~ dbeta (9,1)                                 # mode =

}

# data
list(K = 1, total = c(88),
            x = structure(.Data = c(18,1,0,0,0,0,0,0,20,2,13,16,8,1,0,9),
                                                                                                .Dim = c(1,16))
```

WinBUGS code: 4 Independent Tests, 1 population

```
model;
{

# define multinomial distribution

        # x array denotes cell frequency data
        # total is the number of animals sampled
        # p is the probability estimate for multinomial cell
        # K is the number of species

                for (k in 1:K) {
                        x[k, 1:16] ~ dmulti (p[k, 1:16], total[k])
                }


# Main probability calculations
        # 0 is coded test negative
        # 1 is coded test positive
        # Test order:  CS -  VI  -  LG  -  GG

                        # define acceptable range for conditional covariance--positive (Se)

                        # alpha[1] <- max(-(1-Se[1])*(1-Se[2]), -Se[1]*Se[2])
                        # beta[1] <- min(Se[1]*(1-Se[2]),Se[2]*(1-Se[1]))
                        # alpha[2] <- max(-(1-Se[1])*(1-Se[3]), -Se[1]*Se[3])
                        # beta[2] <- min(Se[1]*(1-Se[3]),Se[3]*(1-Se[1]))
                        # alpha[3] <- max(-(1-Se[1])*(1-Se[4]), -Se[1]*Se[4])
                        # beta[3] <- min(Se[1]*(1-Se[4]),Se[4]*(1-Se[1]))
                        # alpha[4] <- max(-(1-Se[2])*(1-Se[3]), -Se[2]*Se[3])
                        # beta[4] <- min(Se[2]*(1-Se[3]),Se[3]*(1-Se[2]))
                        # alpha[5] <- max(-(1-Se[2])*(1-Se[4]), -Se[2]*Se[4])
                        # beta[5] <- min(Se[2]*(1-Se[4]),Se[4]*(1-Se[2]))
                        #alpha[6] <- max(-(1-Se[3])*(1-Se[4]), -Se[3]*Se[4])
                        #beta[6] <- min(Se[3]*(1-Se[4]),Se[4]*(1-Se[3]))


                        # define conditional covariance positive

                        CSxVISe <- 0
                        CSxLGSe <- 0
                        CSxGGSe <- 0
                        VIxLGSe <- 0
                        VIxGGSe <- 0
                        LGxGGSe <- 0
                        #LGxGGSe ~ dunif(alpha[6], beta[6])

                        # define acceptable range for conditional covariance--negative (Sp)

                        # gamma[1] <- max(-(1-Sp[1])*(1-Sp[2]), -Sp[1]*Sp[2])
                        # delta[1] <- min(Sp[1]*(1-Sp[2]),Sp[2]*(1-Sp[1]))
                        # gamma[2] <- max(-(1-Sp[1])*(1-Sp[3]), -Sp[1]*Sp[3])
                        # delta[2] <- min(Sp[1]*(1-Sp[3]),Sp[3]*(1-Sp[1]))
                        # gamma[3] <- max(-(1-Sp[1])*(1-Sp[4]), -Sp[1]*Sp[4])
                        # delta[3] <- min(Sp[1]*(1-Sp[4]),Sp[4]*(1-Sp[1]))
                        # gamma[4] <- max(-(1-Sp[2])*(1-Sp[3]), -Sp[2]*Sp[3])
                        # delta[4] <- min(Sp[2]*(1-Sp[3]),Sp[3]*(1-Sp[2]))
                        # gamma[5] <- max(-(1-Sp[2])*(1-Sp[4]), -Sp[2]*Sp[4])
                        # delta[5] <- min(Sp[2]*(1-Sp[4]),Sp[4]*(1-Sp[2]))
                        #gamma[6] <- max(-(1-Sp[3])*(1-Sp[4]), -Sp[3]*Sp[4])
                        #delta[6] <- min(Sp[3]*(1-Sp[4]),Sp[4]*(1-Sp[3]))


                        # define conditional covariance negative

                        CSxVISp <- 0
                        CSxLGSp <- 0
                        CSxGGSp <- 0
                        VIxLGSp <- 0
                        VIxGGSp <- 0
                        LGxGGSp <- 0
```

```
        #LGxGGSp ~ dunif(gamma[6], delta[6])

for (k in 1:K) {

        p[k,1] <-  # 0 0 0 0 test pattern

                   # Probability due to diseased animals

                   ((prev[k])*(1-Se[1])*(1-Se[2])*(1-Se[3])*(1-Se[4]))+

                   # Probability due to test positive dependence

                   (prev[k]*(CSxVISe+CSxLGSe+CSxGGSe+
                   VIxLGSe+VIxGGSe+LGxGGSe))+

                   # Probability due to non-diseased animals

                   ((1-prev[k])*Sp[1]*Sp[2]*Sp[3]*Sp[4])+

                   # Probaility due to test negative dependence

                   ((1-prev[k])*(CSxVISp+CSxLGSp+CSxGGSp+
                   VIxLGSp+VIxGGSp+LGxGGSp))

        p[k,2] <-  # 0 0 01 test pattern

                   ((prev[k])*(1-Se[1])*(1-Se[2])*(1-Se[3])*(Se[4]))+

                   (prev[k]*(CSxVISe+CSxLGSe-CSxGGSe+
                   VIxLGSe-VIxGGSe-LGxGGSe))+

                   ((1-prev[k])*Sp[1]*Sp[2]*Sp[3]*(1-Sp[4]))+

                   ((1-prev[k])*(CSxVISp+CSxLGSp-CSxGGSp+
                   VIxLGSp-VIxGGSp-LGxGGSp))

        p[k,3] <-  # 0 01 0 test pattern

                   ((prev[k])*(1-Se[1])*(1-Se[2])*(Se[3])*(1-Se[4]))+

                   (prev[k]*(CSxVISe-CSxLGSe+CSxGGSe-
                   VIxLGSe+VIxGGSe-LGxGGSe))+

                   ((1-prev[k])*Sp[1]*Sp[2]*(1-Sp[3])*Sp[4])+

                   ((1-prev[k])*(CSxVISp-CSxLGSp+CSxGGSp-
                   VIxLGSp+VIxGGSp-LGxGGSp))

        p[k,4] <-  # 0 011 test pattern

                   ((prev[k])*(1-Se[1])*(1-Se[2])*(Se[3])*(Se[4]))+

                   (prev[k]*(CSxVISe-CSxLGSe-CSxGGSe-
                   VIxLGSe-VIxGGSe+LGxGGSe))+

                   ((1-prev[k])*Sp[1]*Sp[2]*(1-Sp[3])*(1-Sp[4]))+

                   ((1-prev[k])*(CSxVISp-CSxLGSp-CSxGGSp-
                   VIxLGSp-VIxGGSp+LGxGGSp))

        p[k, 5] <- # 01 0 0 test pattern

                   ((prev[k])*(1-Se[1])*(Se[2])*(1-Se[3])*(1-Se[4]))+

                   (prev[k]*(-CSxVISe+CSxLGSe+CSxGGSe-
                   VIxLGSe-VIxGGSe+LGxGGSe))+

                   ((1-prev[k])*Sp[1]*(1-Sp[2])*Sp[3]*Sp[4])+

                   ((1-prev[k])*(-CSxVISp+CSxLGSp+CSxGGSp-
                   VIxLGSp-VIxGGSp+LGxGGSp))

        p[k, 6] <- # 01 01 test pattern

                   ((prev[k])*(1-Se[1])*(Se[2])*(1-Se[3])*(Se[4]))+
```

```
                (prev[k]*(-CSxVISe+CSxLGSe-CSxGGSe-
                VIxLGSe+VIxGGSe-LGxGGSe))+

                ((1-prev[k])*Sp[1]*(1-Sp[2])*Sp[3]*(1-Sp[4]))+

                ((1-prev[k])*(-CSxVISp+CSxLGSp-CSxGGSp-
                VIxLGSp+VIxGGSp-LGxGGSp))

p[k, 7] <-  # 011 0 test pattern

                ((prev[k])*(1-Se[1])*(Se[2])*(Se[3])*(1-Se[4]))+

                (prev[k]*(-CSxVISe-CSxLGSe+CSxGGSe+
                VIxLGSe-VIxGGSe-LGxGGSe))+

                ((1-prev[k])*Sp[1]*(1-Sp[2])*(1-Sp[3])*Sp[4])+

                ((1-prev[k])*(-CSxVISp-CSxLGSp+CSxGGSp+
                VIxLGSp-VIxGGSp-LGxGGSp))

p[k, 8] <-  # 0111 test pattern

                ((prev[k])*(1-Se[1])*(Se[2])*(Se[3])*(Se[4]))+

                (prev[k]*(-CSxVISe-CSxLGSe-CSxGGSe+
                VIxLGSe+VIxGGSe+LGxGGSe))+

                ((1-prev[k])*Sp[1]*(1-Sp[2])*(1-Sp[3])*(1-Sp[4]))+

                ((1-prev[k])*(-CSxVISp-CSxLGSp-CSxGGSp+
                VIxLGSp+VIxGGSp+LGxGGSp))

p[k, 9] <-  #1 0 0 0 test pattern

                ((prev[k])*(Se[1])*(1-Se[2])*(1-Se[3])*(1-Se[4]))+

                (prev[k]*(-CSxVISe-CSxLGSe-CSxGGSe+
                VIxLGSe+VIxGGSe+LGxGGSe))+

                ((1-prev[k])*(1-Sp[1])*Sp[2]*Sp[3]*Sp[4])+

                ((1-prev[k])*(-CSxVISp-CSxLGSp-CSxGGSp+
                VIxLGSp+VIxGGSp+LGxGGSp))

p[k,10] <- #1 0 01 test pattern

                ((prev[k])*(Se[1])*(1-Se[2])*(1-Se[3])*(Se[4]))+

                (prev[k]*(-CSxVISe-CSxLGSe+CSxGGSe+
                VIxLGSe-VIxGGSe-LGxGGSe))+

                ((1-prev[k])*(1-Sp[1])*Sp[2]*Sp[3]*(1-Sp[4]))+

                ((1-prev[k])*(-CSxVISp-CSxLGSp+CSxGGSp+
                VIxLGSp-VIxGGSp-LGxGGSp))

p[k,11] <- #1 01 0 test pattern

                ((prev[k])*(Se[1])*(1-Se[2])*(Se[3])*(1-Se[4]))+

                (prev[k]*(-CSxVISe+CSxLGSe-CSxGGSe-
                VIxLGSe+VIxGGSe-LGxGGSe))+

                ((1-prev[k])*(1-Sp[1])*Sp[2]*(1-Sp[3])*Sp[4])+

                ((1-prev[k])*(-CSxVISp+CSxLGSp-CSxGGSp-
                VIxLGSp+VIxGGSp-LGxGGSp))

p[k,12] <- #1 011 test pattern

                ((prev[k])*(Se[1])*(1-Se[2])*(Se[3])*(Se[4]))+

                (prev[k]*(-CSxVISe+CSxLGSe+CSxGGSe-
                VIxLGSe-VIxGGSe+LGxGGSe))+
```

```
                ((1-prev[k])*(1-Sp[1])*Sp[2]*(1-Sp[3])*(1-Sp[4]))+

                ((1-prev[k])*(-CSxVISp+CSxLGSp+CSxGGSp-
                VIxLGSp-VIxGGSp+LGxGGSp))

        p[k,13] <- #11 0 0 test pattern

                ((prev[k])*(Se[1])*(Se[2])*(1-Se[3])*(1-Se[4]))+

                (prev[k]*(CSxVISe-CSxLGSe-CSxGGSe-
                VIxLGSe-VIxGGSe+LGxGGSe))+

                ((1-prev[k])*(1-Sp[1])*(1-Sp[2])*Sp[3]*Sp[4])+

                ((1-prev[k])*(CSxVISp-CSxLGSp-CSxGGSp-
                VIxLGSp-VIxGGSp+LGxGGSp))

        p[k,14] <- #11 01 test pattern

                ((prev[k])*(Se[1])*(Se[2])*(1-Se[3])*(Se[4]))+

                (prev[k]*(CSxVISe-CSxLGSe+CSxGGSe-
                VIxLGSe+VIxGGSe-LGxGGSe))+

                ((1-prev[k])*(1-Sp[1])*(1-Sp[2])*Sp[3]*(1-Sp[4]))+

                ((1-prev[k])*(CSxVISp-CSxLGSp+CSxGGSp-
                VIxLGSp+VIxGGSp-LGxGGSp))

        p[k,15] <- #111 0 test pattern

                ((prev[k])*(Se[1])*(Se[2])*(Se[3])*(1-Se[4]))+

                (prev[k]*(CSxVISe+CSxLGSe-CSxGGSe+
                VIxLGSe-VIxGGSe-LGxGGSe))+

                ((1-prev[k])*(1-Sp[1])*(1-Sp[2])*(1-Sp[3])*Sp[4])+

                ((1-prev[k])*(CSxVISp+CSxLGSp-CSxGGSp+
                VIxLGSp-VIxGGSp-LGxGGSp))

        p[k,16] <- #1111 test pattern

                ((prev[k])*(Se[1])*(Se[2])*(Se[3])*(Se[4]))+

        (prev[k]*(CSxVISe+CSxLGSe+CSxGGSe+
                VIxLGSe+VIxGGSe+LGxGGSe))+

                ((1-prev[k])*(1-Sp[1])*(1-Sp[2])*(1-Sp[3])*(1-Sp[4]))+

                ((1-prev[k])*(CSxVISp+CSxLGSp+CSxGGSp+
                VIxLGSp+VIxGGSp+LGxGGSp))

        PPV[1] <- ((Se[1]*prev[k])/((Se[1]*prev[k])+(1-Sp[1])*(1-prev[k])))
        NPV[1] <- ((Sp[1]*(1-prev[k]))/((Sp[1]*(1-prev[k]))+((1-Se[1])*prev[k])))

        PPV[2] <- ((Se[2]*prev[k])/((Se[2]*prev[k])+(1-Sp[2])*(1-prev[k])))
        NPV[2] <- ((Sp[2]*(1-prev[k]))/((Sp[2]*(1-prev[k]))+((1-Se[2])*prev[k])))

        PPV[3] <- ((Se[3]*prev[k])/((Se[3]*prev[k])+(1-Sp[3])*(1-prev[k])))
        NPV[3] <- ((Sp[3]*(1-prev[k]))/((Sp[3]*(1-prev[k]))+((1-Se[3])*prev[k])))

        PPV[4] <- ((Se[4]*prev[k])/((Se[4]*prev[k])+(1-Sp[4])*(1-prev[k])))
        NPV[4] <- ((Sp[4]*(1-prev[k]))/((Sp[4]*(1-prev[k]))+((1-Se[4])*prev[k])))
        }


# prior probability distributions

prev[1] ~ dbeta (1,1)                   # mode =
Se[1] ~ dbeta (20,10)                           # mode =
Se[2] ~ dbeta (4.5,11)                          # mode =
Se[3] ~ dbeta (22,8)                  # mode =
Se[4] ~ dbeta (13.5,17)                         # mode =
```

```
Sp[1] ~ dbeta (21,8.5)                          # mode =
Sp[2] ~ dbeta (99,1)                    # mode =
Sp[3] ~ dbeta (11,1.5)                          # mode =
Sp[4] ~ dbeta (9,1)                     # mode =

}

# data
list(K = 1, total = c(88),
        x = structure(.Data = c(18,1,0,0,0,0,0,0,20,2,13,16,8,1,0,9),
                                                                                .Dim = c(1,16))
```