

1 A modelling framework for the prediction of
2 the herd-level probability of infection from
3 longitudinal data

4 Supplementary material 1

5 Madouasse et al.

6 April 13, 2021

7 **Contents**

8	1 Definition of a cut-off and associated characteristics for bulk	
9	 tank milk antibody testing	2
10	2 Simplified JAGS code	3
11	3 Stan code	5

1 Definition of a cut-off and associated characteristics for bulk tank milk antibody testing

14 Test results were reported as optical density ratios (ODR). In the Loire-
15 Atlantique CP, these ODRs are discretised into 3 categories using threshold
16 values of 35 and 60. ODR values below 35 are associated with low antibody
17 levels and ODR values above 60 are associated with high antibody levels.
18 Decision regarding which herds require further testing for the identification
19 and removal of PI animals is complex and involves the combination of test
20 categories on 3 consecutive tests, spanning a year.

21 In this work, the ODR values were discretised in order to convert them
22 into either seropositive (antibodies detected) or seronegative (no antibodies
23 detected) outcomes. The choice of the threshold to apply for the discretisation
24 as well as the sensitivity and specificity of this threshold for the detection
25 of seropositivity were based on the ODR distributions from test data
26 collected outside of the study period. Test data were available until spring
27 2019. However, in spring 2017, a more sensitive test was used, so this testing
28 campaign was not analysed. Therefore, test data collected over the 4
29 campaigns that took place between autumn 2017 and spring 2019 were used.
30 From these data, the overall ODR distribution was modelled as a mixture of
31 normal distributions using the R `mixdist` package (Macdonald & Du, 2018).
32 This was used to select a threshold and determine plausible sensitivity and
33 specificity values for the selected threshold.

34 Test sensitivity and specificity were estimated from 5536 test results collected
35 between 2017 and 2019. Different combinations of normal and log-normal
36 distributions were tested to model the overall ODR distribution, using the R
37 `mixdist` package. The best fit was obtained using 4 normal distributions
38 (Figure 1). The 2 distributions on the left were assumed to be associated with
39 seronegative herds and the 2 distributions on the right were assumed to be
40 associated with seropositive herds. The cut-off of 35 used in the CP seemed
41 to discriminate well between the distributions associated with seronegative
42 and seropositive herds respectively, and was therefore retained in the remainder
43 of the analysis. Using this threshold, there were 44.1% of seropositive tests
44 between 2014 and 2016 and 40.4% between 2017 and 2019, indicating a decrease
45 in the infection prevalence over time. With these assumptions, the test
46 sensitivity and specificity were 0.978 and 0.948 respectively.
47

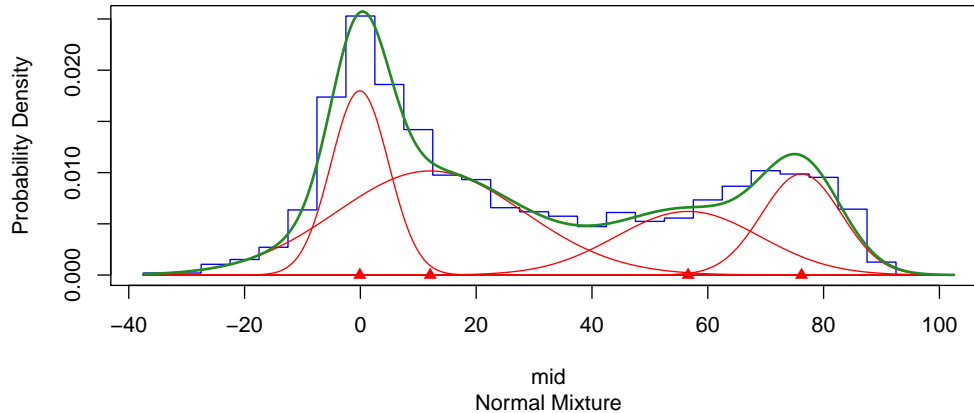


Figure 1: Distribution of the observed optical density ratios (histogram in blue) and fitted mixture of 4 normal distributions (red curves) for the bulk tank milk tests performed between 2017 and 2019. The green curve represents the overall fitted distribution.

48 2 Simplified JAGS code

49 Below is simplified version of the JAGS code. This code assumes that a test
 50 result is available for each month modelled. The model implemented in the
 51 package allows for missing test results using loops to handle different possibil-
 52 ities. The principles are the same, but providing the actual code would make
 53 it less readable. The full model code can be generated by running the exam-
 54 ples provided on the Github web page, by using the argument `model_code`
 55 = `TRUE` in the `STOCfree_JAGS()` function.

```
56 model{
57
58   ## loop over all herds
59   ## t1 is the vector of indices for first month of test in each herd
60   ## tf is the vector of indices for last month of test in each herd
61   for(i in 1:N_herds){
62
63     ### First monthly status of each herd
64     ## probability of being latent status positive for herd i at time = 1
```

```

65   logit_pi1[i] ~ dnorm(logit_pi1_mean, logit_pi1_prec)
66
67   ## latent status for herd i at time = 1
68   Status[t1[i]] ~ dbern(ilogit(logit_pi1[i]))
69
70   ## probability of being test positive given herd status
71   p_test_pos[t1[i]] <- Se * Status[t1[i]] + (1 - Sp) * (1 - Status[t1[i]])
72
73   ## test result associated with first Status => data
74   test_res[t1[i]] ~ dbern(p_test_pos[t1[i]])
75
76   ### Statuses 2 to 1 minus last
77   for(t in (t1[i] + 1):(tf[i] - 1)){
78
79       # probability of new infection
80       # logistic regression
81       logit(tau1[t]) <- inprod(risk_factors[t,], theta)
82
83       ## probability of being status positive given previous status,
84       ## tau1 and tau2
85       pi[t] <- (1 - Status[t - 1]) * tau1[t] +
86               Status[t - 1] * tau2
87
88       ## herd status at time t
89       Status[t] ~ dbern(pi[t])
90
91       ## probability of test positive at time t
92       p_test_pos[t] <- Se * Status[t] + (1 - Sp) * (1 - Status[t])
93
94       ## test result at time t => data
95       test_res[t] ~ dbern(p_test_pos[t])
96
97   }
98
99   # probability of new infection
100  logit(tau1[tf[i]]) <- inprod(risk_factors[tf[i],], theta)
101
102  ## Predicted probability of infection for herd i on last month

```

```

103   pi[tf[i]] <- tau1 * (1 - Status[tf[i] - 1]) +
104             tau2 * Status[tf[i] - 1]
105   # probability of infection updated with test result
106   predicted_proba[tf[i]] <- test_res[tf[i]] * ( # positive test result
107     Se * pi[tf[i]] / (Se * pi[tf[i]] + (1 - Sp) * (1 - pi[tf[i]]))
108   ) + (1 - test_res[tf[i]]) * ( # negative test test result
109     (1 - Se) * test_res[tf[i]] /
110     ((1 - Se) * pi[tf[i]] + Sp * (1 - pi[tf[i]] ) )
111   )
112
113 }
114
115 ### Priors
116 ## test characteristics
117 Se ~ dbeta(Se_beta_a, Se_beta_b)
118 Sp ~ dbeta(Sp_beta_a, Sp_beta_b)
119 ## Status dynamics - sampling on the logit scale
120 logit_tau2 ~ dnorm(logit_tau2_mean, logit_tau2_prec)
121 ## logit back to the probability scale
122 tau2 <- ilogit(logit_tau2)
123
124 ## Logistic regression coefficients
125 for(i_rf in 1:n_risk_factors){
126
127   theta[i_rf] ~ dnorm(theta_norm_mean[i_rf], theta_norm_prec[i_rf])
128
129 }
130
131 }

```

132 3 Stan code

133 Stan implementation of the STOC free model. Missing test results are coded
134 using the value 3 (instead of 0 or 1) which is handled using an if statement.

```

135 data{
136

```

```

137   int<lower=1> n_herds;
138   int<lower=1> herds_t1[n_herds];
139   int<lower=1> herds_t2[n_herds];
140   int<lower=1> herds_T[n_herds];
141   int<lower=1> N;
142   int<lower=0, upper=3> test_res[N];
143   real<lower = 0> Se_beta_a;
144   real<lower = 0> Se_beta_b;
145   real<lower = 0> Sp_beta_a;
146   real<lower = 0> Sp_beta_b;
147   real logit_pi1_mean;
148   real logit_pi1_sd;
149   real logit_tau2_mean;
150   real logit_tau2_sd;
151   int<lower = 0> n_risk_factors;
152   real theta_norm_mean[n_risk_factors];
153   real theta_norm_sd[n_risk_factors];
154   matrix[N, n_risk_factors] risk_factors;
155
156 }
157 parameters{
158
159   real<lower = 0, upper = 1> Se;
160   real<lower = 0, upper = 1> Sp;
161   real<lower = 0, upper = 1> pi1;
162   real<lower = 0, upper = 1> tau2;
163   vector[n_risk_factors] theta;
164
165 }
166 transformed parameters{
167
168   // logalpha needs to be accessible to toher blocks
169   matrix[N, 2] logalpha;
170
171   {
172
173     // accumulator used at each time step
174     real tau1[N];

```

```

175     real accumulator[2];
176
177     // logistic regression for tau1
178     for(n in 1:N){
179
180         tau1[n] = inv_logit(risk_factors[n,] * theta);
181
182     }
183
184
185     // looping over all herds
186     for(h in 1:n_herds){
187
188         // first test in sequence
189         // negative status
190         logalpha[herds_t1[h], 1] = log(1 - pi1) +
191                                     bernoulli_lpmf(test_res[herds_t1[h]] | 1 - Sp);
192         // positive status
193         logalpha[herds_t1[h], 2] = log(pi1) +
194                                     bernoulli_lpmf(test_res[herds_t1[h]] | Se);
195
196         // tests 2 in T in sequence
197         for(t in herds_t2[h]:herds_T[h]){
198
199             if(test_res[t] == 3){
200
201                 // transition from status negative to status negative (j = 1; i = 1)
202                 accumulator[1] = logalpha[t-1, 1] + log(1 - tau1[t]);
203                 // transition from status positive to status negative (j = 1; i = 1)
204                 accumulator[2] = logalpha[t-1, 2] + log(1 - tau2);
205
206                 logalpha[t, 1] = log_sum_exp(accumulator);
207
208
209                 // transition from status negative to status negative (j = 1; i = 1)
210                 accumulator[1] = logalpha[t-1, 1] + log(tau1[t]);
211                 // transition from status positive to status positive (j = 1; i = 1)
212                 accumulator[2] = logalpha[t-1, 2] + log(tau2);

```

```

213
214     logalpha[t, 2] = log_sum_exp(accumulator);
215
216 } else {
217
218     // transition from status negative to status negative (j = 1; i = 1)
219     accumulator[1] = logalpha[t-1, 1] +
220                     log(1 - tau1[t]) +
221                     bernoulli_lpmf(test_res[t] | 1 - Sp);
222     // transition from status positive to status negative (j = 1; i = 1)
223     accumulator[2] = logalpha[t-1, 2] +
224                     log(1 - tau2) +
225                     bernoulli_lpmf(test_res[t] | 1 - Sp);
226
227     logalpha[t, 1] = log_sum_exp(accumulator);
228
229     // transition from status negative to status negative (j = 1; i = 1)
230     accumulator[1] = logalpha[t-1, 1] +
231                     log(tau1[t]) +
232                     bernoulli_lpmf(test_res[t] | Se);
233     // transition from status positive to status positive (j = 1; i = 1)
234     accumulator[2] = logalpha[t-1, 2] +
235                     log(tau2) +
236                     bernoulli_lpmf(test_res[t] | Se);
237
238     logalpha[t, 2] = log_sum_exp(accumulator);
239
240 } // if
241
242 } // time sequence loop
243
244 } // herd loop
245
246 } //local
247
248 } // end of block
249 model{
250

```



```

251
252 // priors for test characteristics
253     Se ~ beta(Se_beta_a, Se_beta_b);
254     Sp ~ beta(Sp_beta_a, Sp_beta_b);
255
256 // priors for status dynamics
257     logit(pi1) ~ normal(logit_pi1_mean, logit_pi1_sd);
258     logit(tau2) ~ normal(logit_tau2_mean, logit_tau2_sd);
259
260 // priors for the logistic regression coefficients
261     for(k in 1:n_risk_factors){
262
263         theta[k] ~ normal(theta_norm_mean[k], theta_norm_sd[k]);
264
265     }
266
267 // update based only on last logalpha of each herd
268     for(i in 1:n_herds)
269         target += log_sum_exp(logalpha[herds_T[i]]);
270
271 }
272 generated quantities{
273
274     // variable in which predictions are stored
275     real pred[n_herds];
276
277     {
278         matrix[n_herds, 2] alpha;
279
280         // loop in which the probabilities of infection are predicted
281         for(i in 1:n_herds){
282             alpha[i] = softmax(logalpha[herds_T[i],])';
283             pred[i] = alpha[i, 2];
284         }
285     }
286 }

```

287 **References**

- 288 Macdonald, P., & Du, J. 2018. *mixdist: Finite Mixture Distribution Models*.
289 R package version 0.5-5.