



The STOC free model

Objectives, hypotheses and model overview

Aurélien Madouasse
&
the STOC free consortium

<https://www.stocfree.eu/>

May 3, 2021



Table of Contents

① Context and objectives

② Modelling framework

③ Parameter estimation and status prediction

④ Implementation

The STOC free model in JAGS

The STOC free model in Stan

Comparison of the JAGS and Stan implementations

STOCfree package

Table of Contents

1 Context and objectives

2 Modelling framework

3 Parameter estimation and status prediction

4 Implementation

The STOC free model in JAGS

The STOC free model in Stan

Comparison of the JAGS and Stan implementations

STOCfree package

Infectious diseases of cattle

Regulated diseases

- Public health threats
e.g. Tuberculosis
- Economic impact
e.g. Foot and mouth disease
- Legislation on how to perform surveillance in order to substantiate freedom from disease
- Every country performs surveillance in the same way → comparable output

⇒ **Input-based surveillance**

Infectious diseases of cattle

Non-regulated diseases

- A lot of important infectious diseases are not regulated but have regional / national control programmes in place
e.g. BVD, paratuberculosis . . .
 - No *legal* prescription on the way to perform surveillance
 - Important diversity in the design of surveillance programmes
 - The *free status* in one programme can have a different meaning than the *free status* in another programme
- ⇒ Creates difficulties when trading animals between herds enrolled in different programmes
- ⇒ **Output-based surveillance**: production of an output that is comparable regardless of the input surveillance data

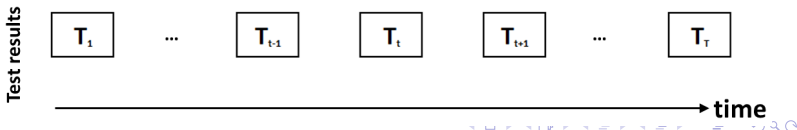
A method for output-based surveillance

- Need for a method taking inputs from diverse surveillance programmes able to produce an output that is comparable
 - Structure of the method/model determined by what is common across surveillance programmes
 - Probability of freedom from infection estimated in a given programme from:
 - surveillance data available in the programme
 - relevant knowledge (e.g. test characteristics)

Control programmes

Common features

- *Control* programmes against cattle non-regulated diseases: prevalence > 0
- Objective: disease control or eradication
- Organised at regional or country level
- Rely on a surveillance component for the identification of infected herds or animals
- Detection of infection followed by control phase
- Surveillance performed in all participating herds
- Herd tested repeatedly over time
 - ⇒ Longitudinal data



Control programmes

Common features

- Tests are imperfect
 - Sensitivity: $Se = p(T^+|D^+) < 1$
 - Specificity: $Sp = p(T^-|D^-) < 1$

⇒ Uncertainty in the true status of tested animals / herds

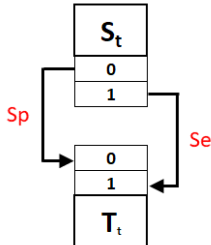


Table of Contents

1 Context and objectives

2 Modelling framework

3 Parameter estimation and status prediction

4 Implementation

The STOC free model in JAGS

The STOC free model in Stan

Comparison of the JAGS and Stan implementations

STOCfree package

Modelling objectives

- Objective: Predict herd level probabilities of (*freedom from*) infection from longitudinal test data collected as part of surveillance programmes against endemic infectious diseases of cattle
- The modelling framework should:
 - Allow the use of longitudinal data i.e. account for the fact that sequences of test results are not random
 - Account for imperfect test information

Hidden Markov models

- Hidden Markov Models (**HMMs**) model a latent discrete variable with a Markovian dynamics, whose state at a given time determines the distribution of an observed variable
 - **discrete variable**: the variable of interest can be in 1 of k states. In the STOC free model, $k = 2$ (positive or negative status)
 - **latent variable**: this discrete variable is not directly observed. In the STOC free model, the **latent status** determines the probability of a negative or positive test result through test sensitivity and specificity
 - **Markovian dynamics**: the latent status is modelled in discrete time steps. The latent status at time t only depends on the status at time $t - 1$. The probabilities of transition between the k states between 2 time points are described by a $k \times k$ transition matrix.

Representation of surveillance programmes as HMMs

Madouasse et
al.

Context and
objectives

Modelling
framework

Estimation &
prediction

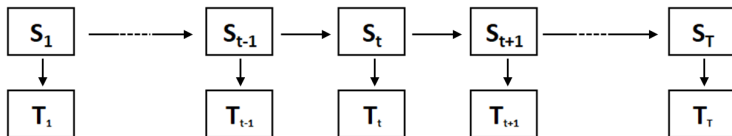
Implementation

JAGS

Stan

JAGS vs. Stan

STOCfree package



- S_t : latent status of interest at time t , from $t = 1$ to $t = T$
- T_t : test result at time t

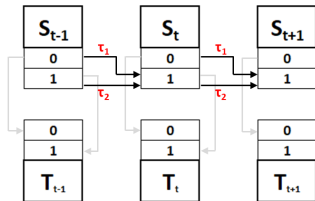
Representation of surveillance programmes as HMMs

Status dynamics

- Time steps of equal duration
(Discrete-time model)
- Herd status at time t depends on
herd status at time t_1
(Markovian property)

$$S_t \sim \text{Bernoulli}(\pi_t)$$

$$\pi_t = \begin{cases} \tau_1 & \text{if } S_{t-1} = 0 \\ \tau_2 & \text{if } S_{t-1} = 1 \end{cases}$$

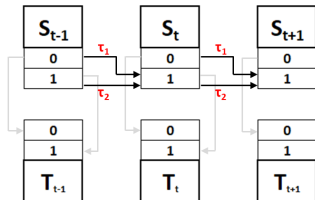


Representation of surveillance programmes as HMMs

Status dynamics

- Herd level probability of new infection (τ_{1t}) modelled as a function of one or several risk factors (X_t) using logistic regression

$$\ln\left(\frac{\tau_{1t}}{1 - \tau_{1t}}\right) = X_t\theta$$



Representation of surveillance programmes as HMMs

Status dynamics

Madouasse et
al.

Context and
objectives

Modelling
framework

Estimation &
prediction

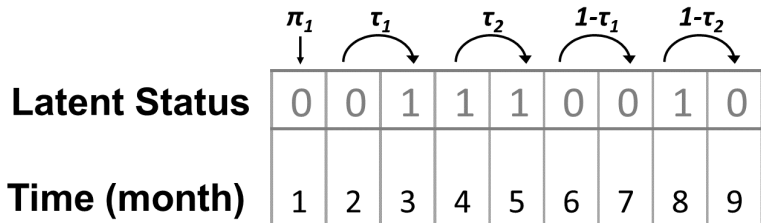
Implementation

JAGS

Stan

JAGS vs. Stan

STOCfree package



Representation of surveillance programmes as HMMs

Test results

- Herd status at time t is either negative (0) OR positive (1)

S_t
0
1

- Test result at time t is either negative (0) OR positive (1)

0
1
T_t

Representation of surveillance programmes as HMMs

Test results

Madouasse et
al.

Context and
objectives

Modelling
framework

Estimation &
prediction

Implementation

JAGS

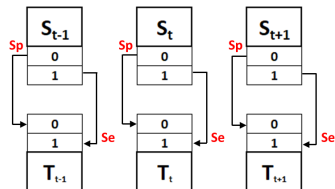
Stan

JAGS vs. Stan

STOCfree package

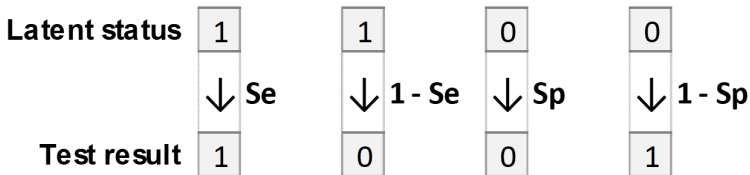
- Test results at the herd level
- Test results depend on the latent status through sensitivity and specificity

$$p(T_t = 1) = \begin{cases} 1 - Sp & \text{if } S_t = 0 \\ Se & \text{if } S_t = 1 \end{cases}$$

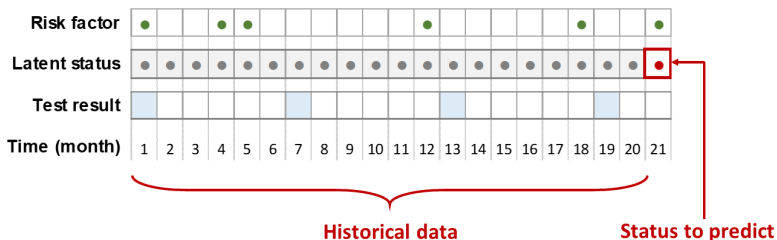


Representation of surveillance programmes as HMMs

Test results



- The model predicts a herd level probability of being status positive at time T
- All the data available up to T are used for parameter estimation



Summary of the different model parameters

- π_1 : probability of being status positive on the first test month
- τ_1 : probability of becoming status positive between 2 months
- $\theta_1, \theta_2 \dots$: coefficients of the logistic regression for the probability of becoming status positive
- τ_2 : probability of remaining status positive between 2 months
- Se : herd-level test sensitivity
- Sp : herd-level test specificity

The complete model - no risk factor

$$S_1 \sim \text{Bernoulli}(\pi_1)$$

$$S_t \sim \text{Bernoulli}(\pi_t) \quad \forall t > 1$$

$$\pi_t = \begin{cases} \tau_1 & \text{if } S_{t-1} = 0 \\ \tau_2 & \text{if } S_{t-1} = 1 \end{cases}$$

$$T_t \sim \text{Bernoulli}(p(T_t))$$

$$p(T_t) = \begin{cases} 1 - Sp & \text{if } S_t = 0 \\ Se & \text{if } S_t = 1 \end{cases}$$

The complete model - with risk factors

$$S_1 \sim \text{Bernoulli}(\pi_1)$$

$$S_t \sim \text{Bernoulli}(\pi_t) \quad \forall t > 1$$

$$\pi_t = \begin{cases} \tau_{1t} & \text{if } S_{t-1} = 0 \\ \tau_2 & \text{if } S_{t-1} = 1 \end{cases}$$

$$\ln\left(\frac{\tau_{1t}}{1 - \tau_{1t}}\right) = X_{ht}\theta$$

$$T_t \sim \text{Bernoulli}(p(T_t))$$

$$p(T_t) = \begin{cases} 1 - Sp & \text{if } S_t = 0 \\ Se & \text{if } S_t = 1 \end{cases}$$

Table of Contents

1 Context and objectives

2 Modelling framework

3 Parameter estimation and status prediction

4 Implementation

The STOC free model in JAGS

The STOC free model in Stan

Comparison of the JAGS and Stan implementations

STOCfree package

Data, hypotheses and parameters

- Data: test results, risk factors
- What we need to know: probability of infection on the current month
- What we know (more or less): test characteristics, characteristics of infection dynamics ...
- The modelling framework needs to be able to predict herd level probabilities of infection from test results and knowledge about test characteristics
- Chosen approach: Bayesian inference

Bayes' theorem

- What is a conditional probability?
 - Probability of an event given that another event has already happened

Sensitivity = probability of a positive test result (T^+) given that ($|$) an individual is diseased (D^+)

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

Positive predictive value = probability that an individual is diseased given a positive test result

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

Bayes' theorem

- What is Bayes' theorem?
 - A simple formula that relates $p(B|A)$ to $p(A|B)$

$$p(B|A) = \frac{p(A|B)p(B)}{p(A)}$$

Bayes' theorem

- Bayes' theorem applied to determining the probability of disease given a positive test result
 - Usually we know test sensitivity, and we would like to know the probability that disease is present given test result

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

- $p(D^+|T^+)$: Positive predictive value
- $p(T^+|D^+)$: Test sensitivity
- $p(D^+)$: Disease prevalence
- $p(T^+)$: Probability of a positive test

Bayes' theorem

- Bayes' theorem applied to determining the probability of disease given a positive test result

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

Bayes' theorem

- Bayes' theorem applied to determining the probability of disease given a positive test result

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

$$p(D^+|T^+) = \frac{p(T^+|D^+)p(D^+)}{p(T^+|D^+)p(D^+) + p(T^+|D^-)p(D^-)}$$

Bayes' theorem

- Bayes' theorem applied to determining the probability of disease given a positive test result

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

$$p(D^+|T^+) = \frac{p(T^+|D^+)p(D^+)}{p(T^+|D^+)p(D^+) + p(T^+|D^-)p(D^-)}$$

$$p(D^+|T^+) = \frac{Se\pi}{Se\pi + (1 - Sp)(1 - \pi)}$$

Bayes' theorem

Bayes' theorem applied to statistical inference

- We have some data (y) and a model, we would like to know what is the probability of the model parameter (θ) values given these data

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)}$$

$p(\theta|y)$ Probability of parameters given data → **Posterior distribution**

$p(y|\theta)$ Probability of data given parameters → **Likelihood function**

$p(\theta)$ Parameter prior distributions → **priors**

$p(y)$ **Normalising constant**

Bayes' theorem

Bayes' theorem applied to statistical inference

- Bayesian inference is a way to estimate model parameters incorporating:
 - data
 - prior knowledge/hypotheses about the model parameters

Bayes' theorem

Bayes' theorem applied to statistical inference

- The normalising constant $p(y)$:
 - is an integral that cannot be readily computed, except in simple cases
 - makes the area under the posterior density curve sum to 1
 - is a constant

$$p(y) = \int p(y|\theta)p(\theta)d\theta$$

Bayes' theorem

Bayes' theorem applied to statistical inference

- Because in most cases the normalising constant cannot be computed, we need estimation methods that do not need to compute it for the estimation of the full posterior density

$$p(\theta|y) \propto p(y|\theta)p(\theta)$$

- Solution: draw many samples from likelihood \times prior distribution using Markov Chain Monte Carlo

Markov Chain Monte Carlo

- **Monte Carlo:** draw random samples (θ) from statistical distributions
- **Markov Chain:** the next random values drawn depend on the values of the current ones

$$p(\theta|y) \propto p(y|\theta)p(\theta)$$

Markov Chain Monte Carlo

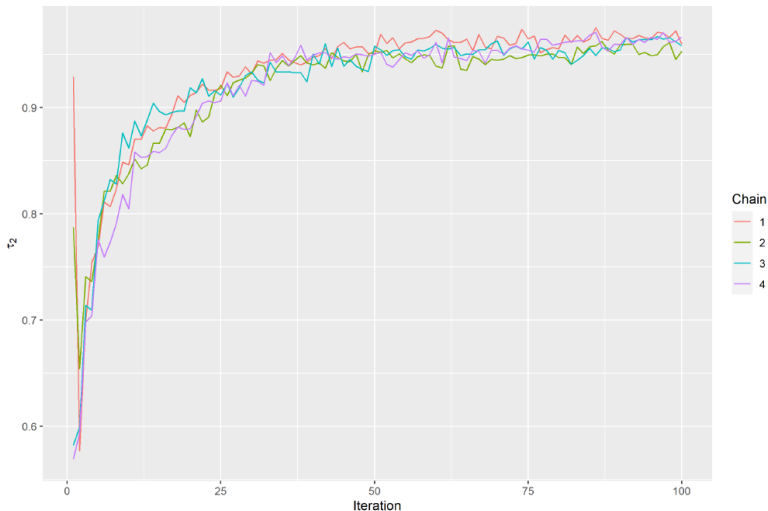
Principles of MCMC algorithms

- Start with some random initial values ($t = 1$)
- Use values at current iteration to sample values at next iteration (Markovian transition)
- The Markov Chain is constructed in such a way that it moves towards the target posterior probability distribution
- There is no way to be absolutely sure that the samples come from the target distribution
 - First iterations discarded \Rightarrow burn in or warmup
 - Different simulations are run in parallel \Rightarrow chains

Markov Chain Monte Carlo

Convergence

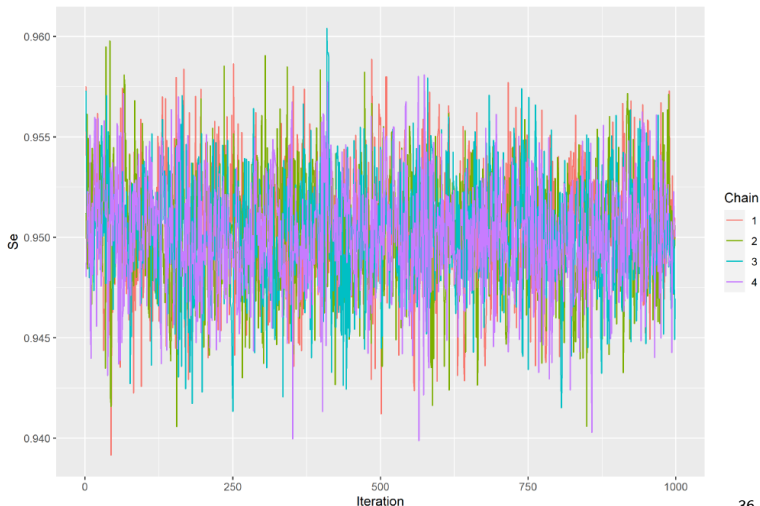
- Moving towards the target distribution



Markov Chain Monte Carlo

Convergence

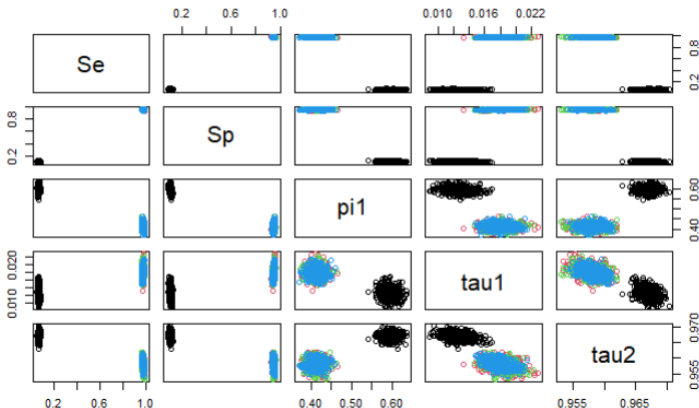
- All chains should converge to the same distribution → traceplot



Markov Chain Monte Carlo

Convergence

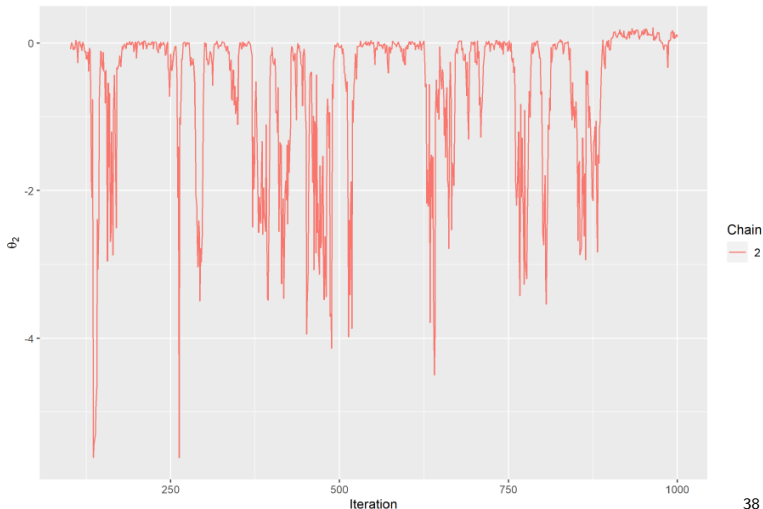
- Different chains can converge to different distributions
 - Model with 5 parameters / each color is a chain



Markov Chain Monte Carlo

Autocorrelation

- Autocorrelation: within a chain, high correlation between consecutive MCMC samples



Markov Chain Monte Carlo

in practice

- Run several chains (> 2): allows checking that the samples obtained do not come from different distributions = convergence (traceplots , Gelman Rubin statistic)
- Initialise each chain with different values
- Discard the first n iterations = burn in, warmup
- If autocorrelation, use 1 out of k iterations (depending on autocorrelation) = thinning

Markov Chain Monte Carlo

Algorithms

- There exist many different MCMC algorithms:
 - Metropolis: first invented (1953)
See an introduction here: [Ben Lambert - An introduction to the Random Walk Metropolis algorithm](#)
 - Metropolis Hastings
 - Gibbs sampling: BUGS, WinBUGS, JAGS
 - Hamiltonian Monte Carlo: Stan

Markov Chain Monte Carlo

Gibbs sampling

- First widely used algorithm for Bayesian inference
 - Not possible before the 1990s because computation intensive
 - First implementations:
 - BUGS = Bayesian Inference Using Gibbs Sampling
 - WinBUGS , OpenBUGS
 - Most recent implementations
 - **JAGS**: Just Another Gibbs Sampler
 - MultiBUGS
- ⇒ Same principles, more efficient

Markov Chain Monte Carlo

Gibbs sampling

- All the programmes use the same language to code statistical models
- Easy to write the code from model specification
- Straightforward to translate the STOC free model equations into code

Markov Chain Monte Carlo

Hamiltonian Monte Carlo

- Implemented in **Stan**
- Much more efficient than Gibbs sampling
 - Exploration of the posterior distribution much more efficient
 - Less autocorrelation → requires less iterations
- Does not support latent discrete parameters
 - Not possible to code the STOC free model as simply as in JAGS

Markov Chain Monte Carlo

- For a visual comparison of different MCMC algorithms, see: [The Markov-chain Monte Carlo Interactive Gallery by Chi-Feng](#)

Table of Contents

1 Context and objectives

2 Modelling framework

3 Parameter estimation and status prediction

4 Implementation

The STOC free model in JAGS

The STOC free model in Stan

Comparison of the JAGS and Stan implementations

STOCfree package

JAGS model

- Easy to go from model equations to JAGS code
- On the following slides:
 - Simplified version in which test results assumed available for all months
 - The *real* model allows for missing test results with a complicated system of loops. Same idea but harder to read
 - When no test available, the dynamics drive status evolution

JAGS model

First status

```
model{
  ## loop over all herds
  ## t1 is the vector of indices for first month of test in each herd
  ## t2 is the vector of indices for second month of test in each herd
  ## tf is the vector of indices for last month of test in each herd
  for(i in 1:N_herds){

    ### First monthly status of each herd
    ## probability of being latent status positive for herd i at t1
    logit_pi1[i] ~ dnorm(logit_pi1_mean, logit_pi1_prec)

    ## latent status for herd i at time = 1
    Status[t1[i]] ~ dbern(ilogit(logit_pi1[i]))

    ## probability of being test positive given herd status
    p_test_pos[t1[i]] <- Se * Status[t1[i]] +
                        (1 - Sp) * (1 - Status[t1[i]])

    ## test result associated with first Status => data
    test_res[t1[i]] ~ dbern(p_test_pos[t1[i]])

    ...
  }
}
```

JAGS model

Statuses 2 to last - 1

```
### Statuses 2 to 1 minus last
for(t in (t1[i] + 1):(tf[i] - 1)){

  # probability of new infection
  # logistic regression
  logit(tau1[t]) <- inprod(risk_factors[t,], theta)

  ## probability of being status positive given previous status,
  ## tau1 and tau2
  pi[t] <- (1 - Status[t - 1]) * tau1[t] +
           Status[t - 1] * tau2

  ## herd status at time t
  Status[t] ~ dbern(pi[t])

  ## probability of test positive at time t
  p_test_pos[t] <- Se * Status[t] + (1 - Sp) * (1 - Status[t])

  ## test result at time t => data
  test_res[t] ~ dbern(p_test_pos[t])

}
```

JAGS model

Predicted statuses

```
# probability of new infection
logit(tau1[tf[i]]) <- inprod(risk_factors[tf[i],], theta)

## Predicted probability of infection for herd i on last month
pi[tf[i]] <- tau1 * (1 - Status[tf[i] - 1]) +
              tau2 * Status[tf[i] - 1]

# probability of infection updated with test result
predicted_proba[tf[i]] <- test_res[tf[i]] * (
  Se * pi[tf[i]] / (Se * pi[tf[i]] + (1 - Sp) * (1 - pi[tf[i]]))
) + (1 - test_res[tf[i]]) * (
  (1 - Se) * test_res[tf[i]] /
  ((1 - Se) * pi[tf[i]] + Sp * (1 - pi[tf[i]]))
)

}
```

JAGS model

Priors

```
### Priors
## test characteristics
Se ~ dbeta(Se_beta_a, Se_beta_b)
Sp ~ dbeta(Sp_beta_a, Sp_beta_b)

## Status dynamics - sampling on the logit scale
logit_tau2 ~ dnorm(logit_tau2_mean, logit_tau2_prec)

## logit back to the probability scale
tau2 <- ilogit(logit_tau2)

## Logistic regression coefficients
for(i_rf in 1:n_risk_factors){

  theta[i_rf] ~ dnorm(theta_norm_mean[i_rf], theta_norm_prec[i_rf])

}

}
```

Stan model

- Stan implements Hamiltonian Monte Carlo which is expected to be more efficient at sampling from the full posterior distribution
- Stan does not support latent discrete parameters
- Translation of the model's equations not as easy as with JAGS
- Various HMM implementations in Stan described in a tutorial by [Damiano et al. \(2017\)](#)

Stan model

- Forward algorithm adapted from the tutorial
- Same model as the JAGS version, but estimation performed in a different way

Stan model

Declaration of variables

```
data{  
  int<lower=1> n_herds;  
  int<lower=1> herds_t1[n_herds];  
  int<lower=1> herds_t2[n_herds];  
  int<lower=1> herds_T[n_herds];  
  int<lower=1> N;  
  int<lower=0, upper=3> test_res[N];  
  real<lower = 0> Se_beta_a;  
  real<lower = 0> Se_beta_b;  
  real<lower = 0> Sp_beta_a;  
  real<lower = 0> Sp_beta_b;  
  real logit_pi1_mean;  
  real logit_pi1_sd;  
  real logit_tau2_mean;  
  real logit_tau2_sd;  
  int<lower = 0> n_risk_factors;  
  real theta_norm_mean[n_risk_factors];  
  real theta_norm_sd[n_risk_factors];  
  matrix[N, n_risk_factors] risk_factors;  
}
```

Stan model

Parameters

```
parameters{  
  
  real<lower = 0, upper = 1> Se;  
  real<lower = 0, upper = 1> Sp;  
  real<lower = 0, upper = 1> pi1;  
  real<lower = 0, upper = 1> tau2;  
  vector[n_risk_factors] theta;  
  
}
```


Stan model

Transformed parameters

```
transformed parameters{

  // logalpha needs to be accessible to other blocks
  matrix[N, 2] logalpha;

  {

    // accumulator used at each time step
    real tau1[N];
    real accumulator[2];

    // logistic regression for tau1
    for(n in 1:N){

      tau1[n] = inv_logit(risk_factors[n,] * theta);

    }

  }

}
```

Stan model

First status

```
// looping over all herds
for(h in 1:n_herds){

  // first test in sequence
  // negative status
  logalpha[herds_t1[h], 1] = log(1 - pi1) +
    bernoulli_lpmf(test_res[herds_t1[h]] | 1 - Sp);

  // positive status
  logalpha[herds_t1[h], 2] = log(pi1) +
    bernoulli_lpmf(test_res[herds_t1[h]] | Se);
```

Stan model

Status transition / no test result

```
// tests 2 in T in sequence
for(t in herds_t2[h]:herds_T[h]){

  // Missing test result
  if(test_res[t] == 3){

    // transition from status negative to status negative (j = 1; i = 1)
    accumulator[1] = logalpha[t-1, 1] + log(1 - tau1[t]);
    // transition from status positive to status negative (j = 1; i = 1)
    accumulator[2] = logalpha[t-1, 2] + log(1 - tau2);

    logalpha[t, 1] = log_sum_exp(accumulator);

    // transition from status negative to status negative (j = 1; i = 1)
    accumulator[1] = logalpha[t-1, 1] + log(tau1[t]);
    // transition from status positive to status positive (j = 1; i = 1)
    accumulator[2] = logalpha[t-1, 2] + log(tau2);

    logalpha[t, 2] = log_sum_exp(accumulator);

  } else {
```

Stan model

Status transition / test result

```
// transition from status negative to status negative (j = 1; i = 1)
  accumulator[1] = logalpha[t-1, 1] +
    log(1 - tau1[t]) +
    bernoulli_lpmf(test_res[t] | 1 - Sp);
// transition from status positive to status negative (j = 1; i = 1)
  accumulator[2] = logalpha[t-1, 2] +
    log(1 - tau2) +
    bernoulli_lpmf(test_res[t] | 1 - Sp);

logalpha[t, 1] = log_sum_exp(accumulator);
```

Stan model

Status transition / test result

```
// transition from status negative to status negative (j = 1; i = 1)
  accumulator[1] = logalpha[t-1, 1] +
    log(tau1[t]) +
    bernoulli_lpmf(test_res[t] | Se);
// transition from status positive to status positive (j = 1; i = 1)
  accumulator[2] = logalpha[t-1, 2] +
    log(tau2) +
    bernoulli_lpmf(test_res[t] | Se);

  logalpha[t, 2] = log_sum_exp(accumulator);

} // if

} // time sequence loop

} // herd loop

} //local

} // end of block
```

Stan model

Priors and likelihood

```
model{

  // priors for test characteristics
  Se ~ beta(Se_beta_a, Se_beta_b);
  Sp ~ beta(Sp_beta_a, Sp_beta_b);

  // priors for status dynamics
  logit(pi1) ~ normal(logit_pi1_mean, logit_pi1_sd);
  logit(tau2) ~ normal(logit_tau2_mean, logit_tau2_sd);

  // priors for the logistic regression coefficients
  for(k in 1:n_risk_factors){

    theta[k] ~ normal(theta_norm_mean[k], theta_norm_sd[k]);

  }

  // update based only on last logalpha of each herd
  for(i in 1:n_herds)
    target += log_sum_exp(logalpha[herds_T[i]]);

}
```

Stan model

Predicted status

```
generated quantities{

  // variable in which predictions are stored
  real pred[n_herds];

  {
    matrix[n_herds, 2] alpha;

    // loop in which the probabilities of infection are predicted
    for(i in 1:n_herds){
      alpha[i] = softmax(logalpha[herds_T[i],])';
      pred[i] = alpha[i, 2];
    }
  }
}
```

Comparison of the JAGS and Stan implementations

- The JAGS and Stan implementations of the model were compared using data collected as part of BVDV control programme in France
 - Work under review with **PCI Animal Science**, available as a **pre-print**.
- The Stan implementation:
 - gives the same parameter estimates
 - is much faster
 - converges much better
 - returns predicted probabilities of infection that are easier to interpret

The STOCfree R package

What is an R package?



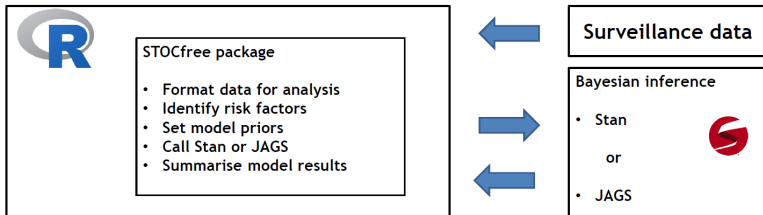
- Programming environment for data manipulation and analysis
- Widely used
- Free



package

- Set of functions gathered to perform specific tasks
- Users install a package and can use the functions they contain
- Packages are installed from the web (CRAN, GitHub...)

The STOCfree R package




The STOCfree R package on Github

- The package is hosted on Github
<https://github.com/AurMad/STOCfree>
- Github is a server hosting:
 - The package code
 - The package documentation
 - The history of development and different package versions, using the Git versioning programme













The STOCfree R package on Github

- All the code is in the R folder


[AurMad / STOCfree](#)
Unwatch

<> Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights

master ▾
2 branches
0 tags
Go to file
Add file ▾
Code ▾

 AurMad Summary method created for predicted probabilities of status po... cfaf611 6 days ago 126 commits		
	R	Summary method created for predicted probabilities of status p... 6 days ago
	README_files/figure-gfm	Summary method created for predicted probabilities of status p... 6 days ago
	data	Readme updated. 10 months ago
	man	Summary method created for predicted probabilities of status p... 6 days ago
	.Rbuildignore	logit and inverse logit functions added 16 months ago
	.gitignore	Initial commit. Functions to format the data for analysis created. 17 months ago
	DESCRIPTION	Rmd file modified with information on how to run the Stan versi... 29 days ago
	NAMESPACE	Summary method created for predicted probabilities of status p... 6 days ago
	README.Rmd	Summary method created for predicted probabilities of status p... 6 days ago
	README.md	Summary method created for predicted probabilities of status p... 6 days ago
	STOCfree.Rproj	Initial commit. Functions to format the data for analysis created. 17 months ago

The STOCfree R package on Github

- The documentation is at the bottom of the page

☰ README.md

STOCfree: prediction of probabilities of freedom from infection from longitudinal data

- [Overview](#)
- [Package installation and update](#)
- [Attaching packages](#)
- [Steps of the analysis](#)
- [Test data](#)
- [Priors for test characteristics](#)
- [Priors for the model parameters related to status dynamics](#)
- [Running the STOC free model in Stan](#)
- [Running the STOC free model in JAGS](#)
- [Model results](#)
- [Inclusion of risk factors](#)

Overview

The aim of the `stocfree` package is to predict herd level probabilities of freedom from infection from

Thank you for your attention



<http://www.stocfree.eu/>

This study was awarded a grant by EFSA and was co-financed by public organisations in the countries participating in the study.