In Depth Analysis of Norovirus Data

Andrew Martin

June 6, 2019

1 Introduction

This report gives an in depth explanation and interpretation of a statistical analysis of norovirus. The analysis is based on a dataset containing information on independent outbreaks of norovirus from a series of different hospital wards. The wards use one of three control actions to attempt to manage the rate of infection. A Markov-Chain Monte Carlo method is used to predict the effective reproduction number for norovirus under normal circumstances, and two independent control strategies.

2 Background information

In general, epidemics are modelled with the assumption that a single person can transition between a series of states over the course of an infection. An individual will typically be (S)usceptible to the disease initially, until they are (E)xposed to the disease, not yet infectious, but will soon become (I)nfectious, where they expose others, until this individual will (R)ecover from the disease with some level of immunity.
It is possible for there to be other variety of states, and for there to be loops, but this is a simple linear SEIR model for a disease.

R_0 denotes the reproduction number, the expected number of secondary infections caused by a single infected individual in an otherwise entirely susceptible population.

$$R_0 = \frac{\beta}{\gamma}$$

Where β is the infection rate, and γ is the recovery rate for the norovirus. $R_0 > 1$ corresponds to a possibility for the disease to become a large-scale epidemic, while for a value $R_0 < 1$ the virus should die out.
The Metropolis-Hastings (MH) algorithm is a Markov Chain Monte Carlo (MCMC) method, which uses Markov chains to obtain the distribution of a random parameter vector Θ. Assuming the parameter comes from a known (or assumed) prior distribution, the stationary distribution of the CTMC produced will be the distribution of the parameter. Where the prior distribution: $P(\Theta)$ is the assumed distribution for the parameter(s) Θ. Posterior distribution: $P(D|\Theta)$ is the distribution of data (in this case known) given the parameter Θ.

3 Assumptions

To obtain a model, for norovirus some basic information on norovirus is considered.

Norovirus is a highly infectious disease. With a time-line of: 12-48 hours after exposure, symptoms appear, after an incubation period. After this incubation period, infection begins and typically lasts 24-48 hours. There is another asymptomatic infectious state for up to 24 hours after this [2].
This suggests some extension of the linear SEIR model, with no loops, and possibly with a secondary asymptomatic infectious state, I.e. the model where an individual can be:

$$\text{Full model: } S \to E \to I_1 \to I_2 \to R$$

Other analysis suggests that the second infectious state can be disregarded with equally valid results, reducing it to the $SEIR$ model [3].

$$\text{Reduced model: } S \to E \to I \to R$$

For the purposes of this report, since the given data is limited to the number of infected and total number of people, there is no possible way to identify a number of exposed, and hence no real value in the inclusion of the exposed state. The information we are seeking (namely the rate of reproduction and effectiveness of control strategies) is not affected by the number of infectious states. Hence the model is fully reduced to the SIR model.

$$\boxed{\text{Final model: } S \to I \to R}$$

The rate of reproduction R_0 has already been researched by the Chief Medical Officer, with the claim that

> R_0 for norovirus in typical hospital settings is between 2 and 3 with (approximately) 66% probability

No prior information has been given regarding either of the intervention strategies. With no background on their effectiveness, denoted α_1, α_2. The expectation is that the intervention strategies would be beneficial in reducing the effective R_0 (I.e. $\alpha_i < 1$) but it is possible that they could have adverse effects ($\alpha_i > 1$). Since no more can be inferred, a reasonable prior

$$\alpha_i \sim U(0, 2)$$

is chosen, noting that a value of 1 would correspond to no change, with a smaller value corresponding to a reduction in infectiousness, and a larger value would cause an increase. Given this prior, if the MH algorithm fails to converge to a value, and appears to increase, this may suggest that $\alpha_i > 2$ and hence α_i would be an extremely detrimental strategy and would need to be discarded immediately.

4 Method

4.1 Data

The dataset, `NorovirusDataA3.txt`, containing information on independent outbreaks across 125 hospital wards is used for all inference in this report. The text file contains comma separated values (CSV) data, where each row denotes an independent ward, and the columns

correspond to: the number of occupied beds, number of people succumbed to the virus, and the treatment action taken universally in the ward (T_0, T_1, T_2), respectively.
Denote the treatments as α_1, α_2 respectively, such that R_0, $\alpha_1 R_0$, $\alpha_2 R_0$ are the effective reproduction numbers for treatments T_0, T_1, and T_2 respectively. The assumption of independence between α_1, α_2, and R_0 is used. This assumption will significantly improve convergence. The assumption does not necessarily have to hold for the posteriors however.

4.2 Distributions

Chosen prior distributions (priors):

$$R_0 \sim N'(\mu, \sigma)$$

$$\alpha_1, \alpha_2 \sim U(0, 2)$$

Where N' is a Normal distribution, concatenated such that all values outside $[0, 5]$ are dropped. Parameters are $\mu = 2.5$ and σ is chosen such that $\approx 66\%$ of the density is contained in $(2, 3)$.
Assume α_i are each uniformly distributed between $0, 2$. This is chosen since $\alpha_i R_0 \geq 0$, and it is expected that α_i will not be significantly detrimental (so as to effectively double R_0). This has been centered around 1, which would correspond to the treatment supplying no change whatsoever.
Since the assumption of uniform priors is used for α_1 and α_2, they can be omitted from the likelihood calculations barring the check that they sit within the valid range $(0, 2)$.

4.3 Algorithm

The Metropolis-Hastings (MH) algorithm takes an initial guess x_0, and then iterates through (variable i): Propose a candidate state from the proposal density, r:

$$x' \sim r(x'|x_i)$$

And accept this state, x' with probability (based on the data):

$$a(x_i, x') = \frac{p(x')r(x_i|x')}{p(x_i)r(x'|x_i)}$$

If accepted, set $x_{i+1} = x'$ or if rejected, set $x_{i+1} = x_i$ and restart the loop.
The modified MH algorithm uses log likelihood for the a calculation instead:

$$\log(a(x_i, x')) = \log(p(x')) + \log(r(x_i|x')) - \log(p(x_i)) - \log(r(x'|x_i))$$

The likelihood calculations are made using an amended copy of the final size distribution code from Ross et al[1].
Using the modified MH algorithm, attempt to obtain parameters R_0, α_1, α_2 where α_i corresponds to the reduction of R_0 caused by treatment i. The code used for this is listed in the appendix. This is done in `MATLAB`, by running the script `R0Predict.m`

5 Analysis and Results

An effective treatment will have $\alpha < 1$, with a theoretically perfect treatment having $\alpha = 0$.
Thee assumption that the wards are *independent*, and there is a control set, where no treatment actions are taken are vital to the analysis. For the methods, with initial guesses for R_0, α_1, α_2: $(4, 1.5, 1.5)$ (overshoot), $(0.5, 0.5, 0.5)$ (undershoot) and $(2, 0.7, 0.7)$ (approximate guess), the MH algorithm is run.

Figure 1 and figure 2 plot the resulting guesses from the MH algorithm. Figure 2 only shows the first 1000 iterations to demonstrate the burn in time, and convergence of solutions. The plots show the solutions obtained for three different sets of parameter guesses: an overshoot, undershoot and approximate guess - all three of which clearly converge to the same values.
Figure 3 displays the dependency between the different parameters. Clearly there is some level of dependency. Particularly between R_0 and the different treatments. This is to be expected, as it is a shared parameter, and changes it it would be reflected in α_i, similarly it would be expected that a decrease in R_0 would correspond to an increase in α_i, to continue to fit with the data. Importantly, there is no correlation between α_1, α_2.
Figure 3 also shows that the expected values for R_0, α_1, α_2 taken assuming no covariances closely resemble where they would belong with covariances included.
Since it appears that all the assumptions for the model are reasonably satisfied and there is clear convergence in the MH algorithm, the values obtained should be close to the true values. Figure 4 plots the kernel-density curves for the parameters, which is the distribution of candidate values for the parameters. The accepted values correspond to the peaks on the density plots. The values obtained are

```
est =

    2.2883    0.8409    0.6781
```

I.e. $R_0 \approx 2.2883$, $\alpha_1 \approx 0.8409$ and $\alpha_2 \approx 0.6781$. Since a smaller value of α_i corresponds to a better means of infection control, α_2 appears to be a better treatment than α_1.
The value for R_0 appears reasonable, as many of the wards experienced quite large outbreaks of the virus.
The corresponding values for $\alpha_1 R_0$ and $\alpha_2 R_0$ obtained are, respectively, 1.9242 and 1.5518. Since both values are still larger than 1 they do not prevent the possibility of a major outbreak, and better alternatives should still be researched. However, the reduction is still significant, and if enforced early in an outbreak, could make a significant difference to the number of infections.

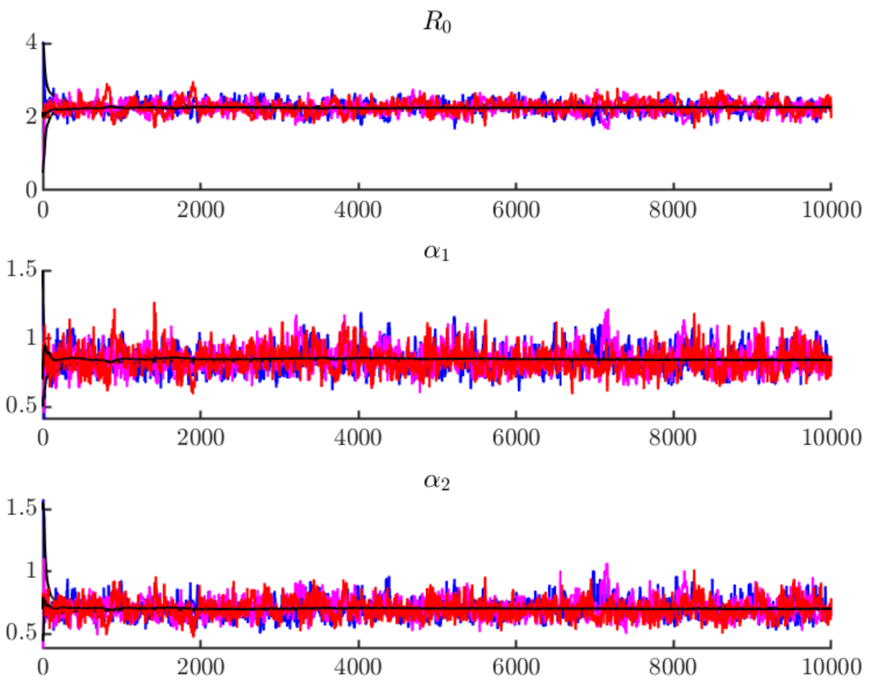


Figure 1: Superimposed trace plots for the estimations of the parameters from the modified MH algorithm

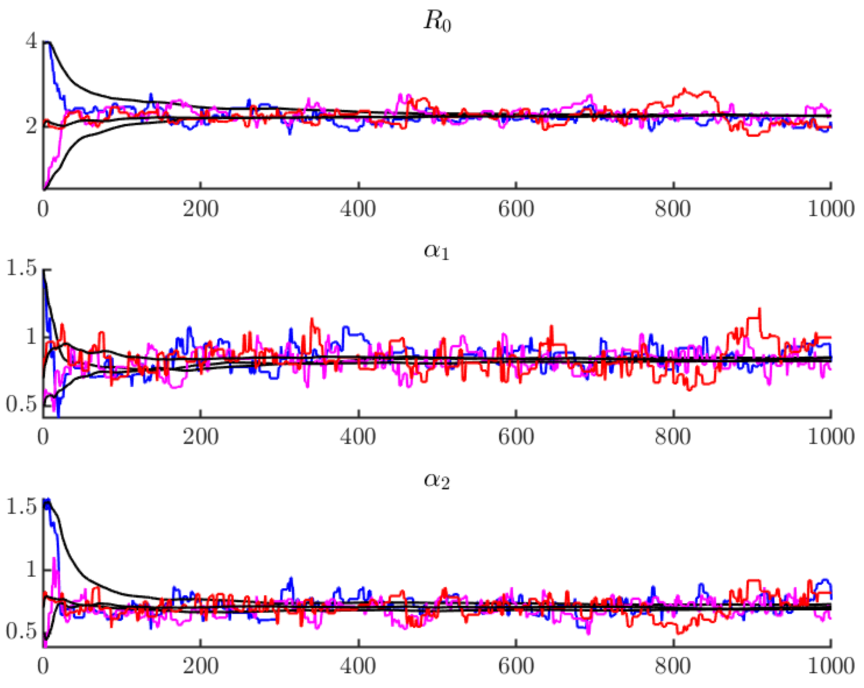
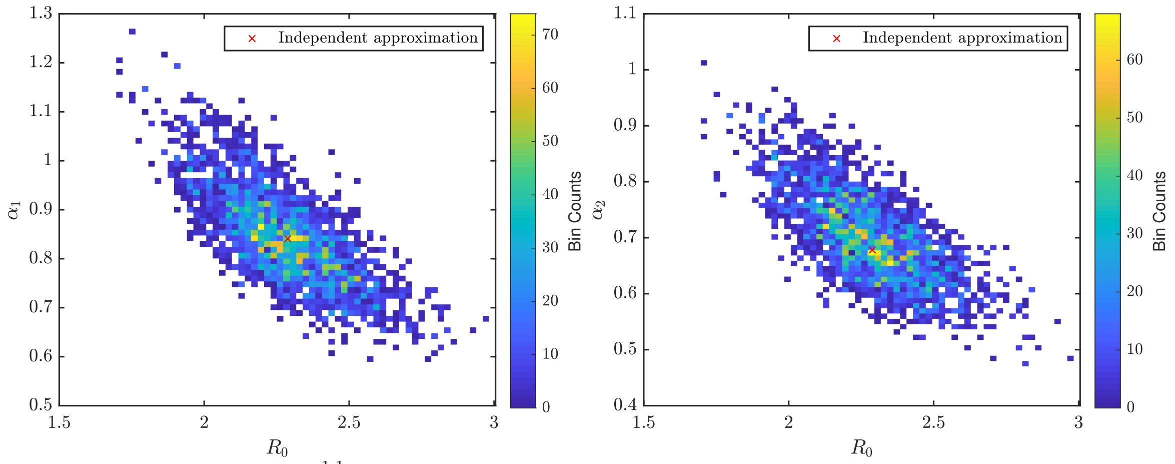


Figure 2: First 1000 iterations of the MH algorithm. Clearly the burn in has completed after approximately 500 iterations for the three parameters



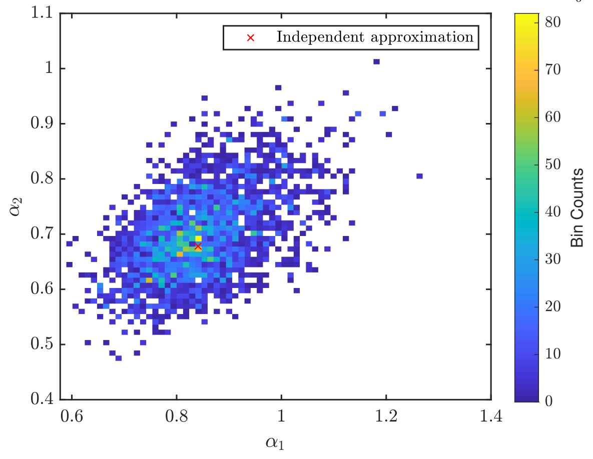


Figure 3: Bin scatter plots of the parameters identifying any correlations and their effect on the value. Top-left: (R_0, α_1), Top-right: (R_0, α_2), Bottom: (α_1, α_2)

6 Conclusion

The effective reproduction numbers, R_0 for the control and two treatment types are found to be, respectively, 2.2883, 1.9242 and 1.5518. The two independent treatments appear to have had positive effects on the reproduction number, with treatment 2, T_2 appearing to be the most effective.
The treatments still do not reduce R_0 below the threshold of $R_0 = 1$, meaning the treatments do not remove the possibility of major outbreaks. For now T_2 should still be implemented as it is a significant improvement on taking no action. However, more research should be taken to attempt to find a more effective treatment.

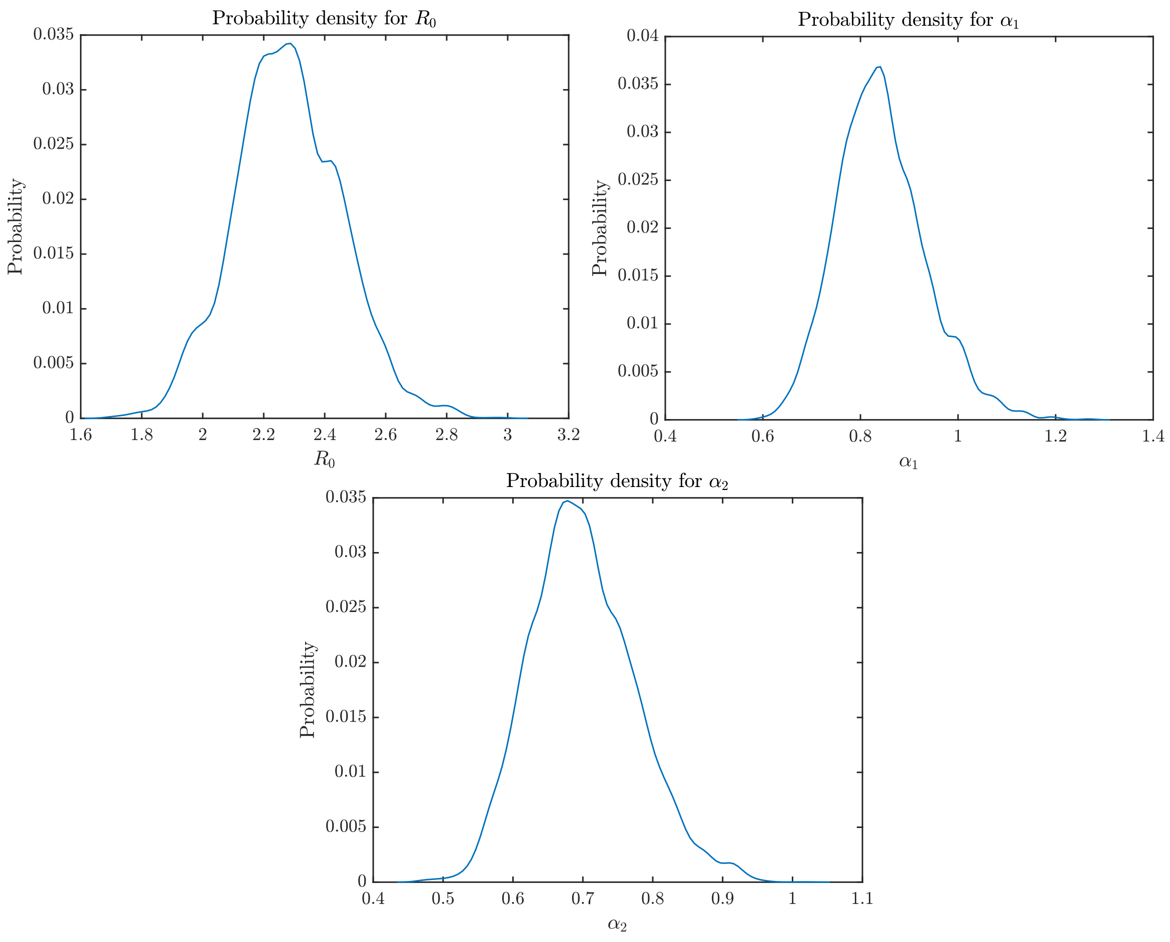


Figure 4: Probability density curves for the parameters - the horizontal axis is the parameter while the vertical is the probability

References

[1] Andrew J. Black and J.V. Ross. "Computation of epidemic final size distributions". In: *Journal of Theoretical Biology* 367 (2015), pp. 159–165. ISSN: 0022-5193. DOI: `https://doi.org/10.1016/j.jtbi.2014.11.029`. URL: `http://www.sciencedirect.com/science/article/pii/S0022519314006882`.

[2] South Australian Health Department. *Norovirus infection*. URL: `https://www.sahealth.sa.gov.au/wps/wcm/connect/public+content/sa+health+internet/health+topics/health+conditions+prevention+and+treatment/infectious+diseases/norovirus+infection`.

[3] K. A. M Gaythorpe et al. "Norovirus transmission dynamics: a modelling review". In: 146.2 (2018), pp. 147–158. ISSN: 0950-2688.

A Code

A.1 Main Script (R0Predict.m)

```
%Pretty plots
set(groot, 'DefaultLineLineWidth', 1, ...
    'DefaultAxesLineWidth', 1, ...
    'DefaultAxesFontSize', 12, ...
    'DefaultTextFontSize', 12, ...
    'DefaultTextInterpreter', 'latex', ...
    'DefaultLegendInterpreter', 'latex', ...
    'DefaultColorbarTickLabelInterpreter', 'latex', ...
    'DefaultAxesTickLabelInterpreter','latex');



%Speeds up the code for multiple runs
%get the dataset once only
if ~exist('dataMat','var')
    %dataMat has format [#people, #infected, #action]
    dataMat = readmatrix('NorovirusDataA3.txt');
    %print the first few rows
    dataMat(1:5,:)
end

%Observation of dataset
%how many of them correspond to the disease effectively dying out?
amountOfData = length(dataMat(:,1))
propDiedOut = sum(dataMat(:,2)<=0.1*dataMat(:,1))/amountOfData



%consistency
rng(1)


```

```
%since we are expecting approximately 66%
%getting the right prior
%we want the dist to be N'(2.5,sigma)
%where sigma is the variance to have approx 66% in (2-3).
%N' since we will drop values below 0 and above 5
%N' will still be symmetric
sigma = -(3-2.5)/norminv((1-0.66)/2);
variance = sigma* speye(3);

varR0 = 0.01;
distStruct.ProposalDist = @(x) mvnrnd(x,diag([varR0,0.01,0.01]));
distStruct.PriorPDF = @(x) normpdf(x,2.5,sigma);
distStruct.LogLikelihoodFunc =@(params) LogLikelihood(params,dataMat);
%constrain a_1,a_2 in [0,2]
distStruct.ProposalConstraints = @(vars) ...
    ProposalConstraints(vars,[0,0,0],[5,2,2]);
%overshoot, undershoot and approximately close to the true solutions
startVars = [4,1.5,1.5;
             0.5,0.5,0.5;
             2,0.7,0.7];

vars = zeros(10000,3,3);
for index = 1:3
    startPoint = startVars(index,:)
    exitflag = 1;
    while exitflag~=0
        [varsTemp,accrate,exitflag]= MetropolisHastingsPassLikelihood...
            (distStruct,startPoint,10000);
        %using exitflag change the variance to get a reasonaable acceptance
        %rate between 0.2, 0.27
        if exitflag==-1
            %acceptance was too low
            %variance is too high
            varR0 = varR0 *2/3;
            distStruct.ProposalDist = @(x) mvnrnd(x,diag([varR0,0.01,0.01]))
            warning('retrying with decreased variance')
        elseif exitflag==1
            %acceptance was too high
            %variance is too low
            varR0 = varR0*2;
            distStruct.ProposalDist = @(x) mvnrnd(x,diag([varR0,0.01,0.01]))
            warning('retrying with decreased variance')
        end
        %store all the solution sets
        vars(:,:,index) = varsTemp;
    end


end
```

```
%%
cumVars = cumsum(vars,1)./(1:length(vars))';
tstr = ["$$R_0$$","$$\alpha_1$$","$$\alpha_2$$"];
for burnin = 0:1
    figure
    for i=1:3
        hold on
        subplot(3,1,i)
        hold on
        %trace plots & trendlines
        plot(vars(:,i,1),'b')
        plot(cumVars(:,i,1),'k')
        plot(vars(:,i,2),'m')
        plot(cumVars(:,i,2),'k')
        plot(vars(:,i,3),'r')
        plot(cumVars(:,i,3),'k')
        title(tstr(i))

        if burnin
            axis([0,1000,-inf,inf])
        end
    end
        saveas(gcf,"MHplot"+num2str(burnin)+".eps","epsc")
end
    axis([0,1000,-inf,inf])
    xlabel('Iteration')

%%
%%Analysis
%just take one set since they all converged
%and omit the burnin
varsClean = vars(500:end,:,3);

%density plot
%and obtain estimates for R0, a1, a2
%axis labels for later
labs = ["$$R_0$$","$$\alpha_1$$","$$\alpha_2$$"];

%est is the estimated [R0,a1,a2]
est = [0,0,0];
for i=1:3
    figure
    [prob,val] = ksdensity(varsClean(:,i));
    prob = prob./sum(prob);
    plot(val,prob)
    xlabel(labs(i))
    ylabel("Probability")
    title("Probability density for "+labs(i))
    [~,ind] =max(prob);
    %assuming there is no dependence
    est(i) = val(ind);
    saveas(gcf,"Probdensity"+num2str(i),"epsc")
end
%print out R0,a1,a2
est
%print out R0a1, R0a2
est(2:3) * est(1)

%covariance-scatter plots
for i=1:2
    for j=i+1:3
        figure
        binscatter(varsClean(:,i),varsClean(:,j),60,'HandleVisibility','off'
        hold on
        scatter(est(i),est(j),'xr')
        xlabel(labs(i))
        ylabel(labs(j))
        legend("Independent approximation")
        colormap(gca,'parula')
        saveas(gcf,"BinScatter"+num2str(i)+num2str(j),"epsc")
    end
end
```

A.2 MetropolisHastingsPassLikelihood.m

```
function [vars,accRate,exitflag]= MetropolisHastingsPassLikelihood...
    (distStruct,startVars,numIterations)
%Most generic Log Likelihood MetropolisHastings
%%INPUTS
%distStruct is a struct containing:
%  the function handles
%   -PriorPDF - the PDF of the prior to obtain likelihood
%   -ProposalDist - the proposed distribution to pull new values
%   -ProposalConstraints - the constraints that the params must abide
%   -LogLikelihoodFunc - the function to obtain the log likelihood
%startVars - the initial values for the vars to estimate
%numIterations - the number of times to iterate through the algorithm
%%OUTPUTS
%vars - the full matrix of variables accepted [nxd] where n is
%      num iterations and d is the number of variables to predict
%accRate - the acceptance rate for the algorithm, a good value is
%           in 0.2 - 0.27
%exitflag - An enum to log the outcome of the algorithm
%          -1, acceptance rate was low (try decreasing proposed variance)
%           0, good acceptance rate
%           1, acceptance rate was high (try increase proposed variance)
numAccepted = 0;
%extract all the functions
PriorPDF = distStruct.PriorPDF;
ProposalDist = distStruct.ProposalDist;
BreaksProposalConstraints = distStruct.ProposalConstraints;
```

```
LogLikelihoodFunc = distStruct.LogLikelihoodFunc;
vars = zeros(numIterations,length(startVars));
vars(1,:) = startVars;

for i=2:numIterations
     %get x'
    proposal = ProposalDist(vars(i-1,:));
   %if breaks constraints
    if BreaksProposalConstraints(proposal)
        vars(i) = vars(i-1);
    else
        %log(a) = ...
         candidateProbTop = LogLikelihoodFunc(proposal)...
             + log(PriorPDF(proposal(1)));
         candidateProbBottom = LogLikelihoodFunc(vars(i-1,:))...
             + log(PriorPDF(vars(i-1,1)));
         candidateProb = candidateProbTop - candidateProbBottom;
         acceptProb= log(rand);
        %should we accept this state?
         if acceptProb< candidateProb
             vars(i,:) = proposal;
             numAccepted = numAccepted +1;
         else
             vars(i,:) = vars(i-1,:);
         end
    end
end
accRate = numAccepted/numIterations;

%Warn if the acceptance rate is suboptimal
%apparently 0.24 is roughly perfect
if  accRate < 0.2
    warning("Bad acceptance rate - too low, accRate = "+num2str(accRate));
    exitflag = -1;
elseif accRate >0.27
    exitflag = 1;
    warning("Bad acceptance rate - too high, accRate = "+num2str(accRate));
else
    exitflag = 0;

end


end
```

A.3 LogLikelihood.m

```
function logLikelihood = LogLikelihood(params,data)
%Amended SIR finalsize code from Ross
%Amended by Andrew Martin
%calculates log likelihood for a given R0,alpha1,alpha2
```

```
%Input:
%params-  [R0,a1,a2]
%data  - [N,I,T_i] where N is the total population, I is number infected
%and T_i is the treatment used
%Output:
%logLikelihood - the logged likelihood of the data given the params
logLikelihood = 0;
% R0      = params(1);
% alpha1  = params(2);
% alpha2  = params(3);
NVec = data(:,1);
%[R0, a1R0, a2R0]
paramsModified = [params(1),params(1)*params(2),params(1)*params(3)];
%yay matlab uses 1 based indexing
R0index = data(:,3)+1;


for iterator=1:length(data)
    N = NVec(iterator);
    relevantParam = paramsModified(R0index(iterator));
    q = zeros(N+1,1);
    q(2) = 1;
    %Proportions for each number of infection events
    %could vectorise, but this is more meaningful
    for Z2 = 0:N
        for Z1 = Z2+1:N-1
            %infection probability (jump prob) - modified from Ross
            infProb = 1 / ( 1 + ((N-1)/(relevantParam*(N-Z1))));
            q(Z1+2) = q(Z1+2) + q(Z1+1)*infProb;
            q(Z1+1) = q(Z1+1)*(1-infProb);
        end
    end
    %sum of the log likelihoods (product of likelihoods)
    logLikelihood= logLikelihood + log(q(data(iterator,2)+1));
end


end
```

A.4 ProposalConstraints.m

```
function boolean = ProposalConstraints(vals,min,max)
%
%OUTPUT:
%boolean - true if the boundary constraints are broken
boolean = any(vals < min | vals > max);

end
```