Web-based Supplementary Materials for "Assessing the impact of a movement network on the spatio-temporal spread of infectious diseases"

by Birgit Schrödle, Leonhard Held and Håvard Rue

Web Appendix A

1 Stationarity region for process (3) of the main manuscript

A vector autoregressive AR(1) process is stationary, if all eigenvalues of $\mathbf{\Omega}$ are less than one in absolute value (Lütkepohl, 2005). As $\mathbf{\Omega}$ depends only on λ, ρ and the known weights $\boldsymbol{W} = (w)_{ji}$, a grid search can reveal the a priori stationarity region of the process (3) for pairs (λ, ρ). The matrix $\mathbf{\Omega}$ can be written as $\mathbf{\Omega} = \lambda \cdot \boldsymbol{I} + \rho \cdot \boldsymbol{W}$. Hence, the R eigenvalues τ for fixed λ, ρ and $\boldsymbol{W}$ can be calculated as $\tau_r^{\Omega} = \lambda + \rho \cdot \tau_r^{W},\ r = 1, ..., R$. A separate grid search must be performed for each weight scheme listed in Table 2 of the main manuscript. The results for all six weight schemes are shown in Figure 1.

The (transformed) cattle trade weights were standardized in a way that they are 1 on average for all pairs of regions with cattle trade. The idea was to obtain model properties similar to the adjacency-driven weights and guarantee direct comparability of the $\hat{\rho}$ estimates. However, the a priori stationarity region for model PM_2 (w_{ji}: $i \sim j$) is very different from the stationarity regions of the cattle trade weights; note the different range of the ρ-axes in Figure 1. Hence, despite the standardization the obtained posterior estimates for $\hat{\rho}$ cannot be directly compared, as each model has different (stationarity) properties depending on the chosen weight matrix.

The question is now, how this knowledge about the prior stationarity region influences the choice of hyperpriors for λ and ρ. On the one hand, it might be desirable to implement a check of stationarity directly within the INLA routine for the computation of $\hat{\lambda}$ and $\hat{\rho}$, in order to adapt the estimates for the diamond shaped stationarity regions in Figure 1. On the other hand, infectious diseases are a highly dynamic phenomenon and long-term stationarity is not expected. Furthermore, in our case study we are only interested in one-step-ahead predictions and, hence, stationarity seems a minor issue. So we used unrestricted $N(0, 0.25)$ distributions as priors for λ and ρ in all PM models.

Note that the posterior mean estimates $(\hat{\lambda}, \hat{\rho})$ for models PM_2-PM_7 are also shown in Figure 1.

2 Supplementary `R INLA` code

2.1 Multivariate time series model without spatio-temporal disease spread

This section describes how to implement the multivariate time series model (2) in the main manuscript, i.e. a model without ρ component, in `R INLA`. As an example, we show the code to fit model PM_1 to the Coxiellosis data, 2004-2009. To reproduce the corresponding line in Table 3 of the main manuscript, the model has to be fitted to the period 2004-2008, excluding the observations from 2009.

The important feature is the `replicate`-option, which fits identical copies of a temporal AR(1) process for all regions.

```
#
# Coxiellosis data are stored in a data.frame
```

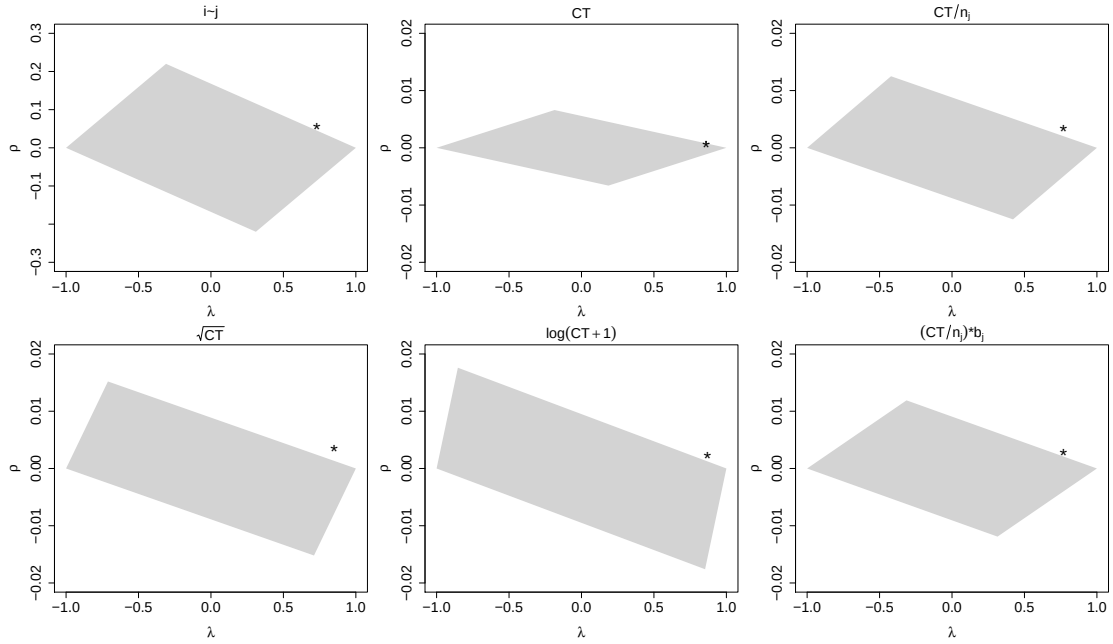

Figure 1: Prior stationarity region (grey color) as a function of λ and ρ for models PM_2 (w_{ji}: $i \sim j$), PM_3 (w_{ji}: CT_{ji}), PM_4 (w_{ji}: CT_{ji}/n_j), PM_5 (w_{ji}: $\sqrt{CT_{ji}}$), PM_6 (w_{ji}: $\log(CT_{ji}+1)$) and PM_7 (w_{ji}: $(CT_{ji}/n_j) \cdot b_j$). Furthermore, the posterior mean estimate $(\hat{\lambda}, \hat{\rho})$ for each model is shown (*), see Table 3 of the main manuscript.

```
# the order of the data is (i,t)
#
load("data_web_appendix_A.RData")
Y <- data_cox[,1]  # vector of cases
m <- data_cox[,2]  # offset
I <- 185  # number of regions
T <- 6    # number of timepoints
zeta <- rep(1:T,I)
id <- rep(1:I,each=T)
# define model formula for PM_1
f_PM_1 <- Y~f(zeta, model="ar1", replicate=id, param=c(0.1,0.001,0,0.2))
# call model
library(INLA)
PM_1 <- inla(f_PM_1, family="Poisson", E=m, data=data.frame(Y,m,zeta,id))
```

2.2 Multivariate time series model with spatio-temporal disease spread

This section describes how to fit model (4) of the main manuscript, i.e. a model including a ρ component, using R INLA.

WARNING: Fitting such a model with INLA may take several hours!

Pre-processing If we want to include a ρ component as in equation (4), an augmented model containing pseudo-observations must be implemented. A detailed description of the available toolbox is given in Ruiz-Cardenas et al. (2010) and Section 3 of the main

manuscript. To code the index and weight vectors for the first and second stage of model (4), the `coding()` function can be used.

```
#
# coding(): function to code the index and weight vectors/matrices
#           of the components of the first and second stage of model (4)
# wm: matrix containing the weights (w_ji)^T
#
coding <- function(I, T, wm){
# function to transform the (i,t) coordinates of an observation in a vector coordinate j
  idx <- function(t, i, T){
    j <- (i-1)*T+t
    if (t<1) return(NA)
    else return(j)
  }

# initialize the second part of the index vector for zeta and zetatm1 (lambda)
  zeta.p2 <- rep(NA,I*T)
  zetatm1.p2 <- rep(NA,I*T)

# initialize the second part of the rho component matrix and the respective weights
  rho.p2 <- matrix(NA,nrow=I*T,ncol=I)
  w.rho.p2 <- matrix(NA,nrow=I*T,ncol=I)

# go through the vector with dimension I*T=J
  j <- 1
  for (i in 1:I){
    for (t in 1:T){
# define the coordinates of zeta and zetatm1 for each vector node j
      zeta.p2[j] <- idx(t,i,T)
      zetatm1.p2[j] <- idx(t-1,i,T)

# define the rho coordinate and weight of each region k for vector node j
      for (k in 1:I){
        rho.p2[j,k] <- ifelse(wm[i,k]==0,NA,idx(t-1,k,T))
        w.rho.p2[j,k] <- ifelse(wm[i,k]==0,NA,-wm[i,k])
      }
      j <- j+1
    }
  }

# put together the first and second part of the index/weight vectors and matrices
  zeta <- c(zeta.p2,zeta.p2)
  zetatm1 <- c(rep(NA,I*T),zetatm1.p2)
  rho <- rbind(matrix(NA,nrow=(I*T),ncol=I),rho.p2)
  w.rho <- rbind(matrix(NA,nrow=(I*T),ncol=I),w.rho.p2)

  return(list(zeta,zetatm1,rho,w.rho))
}
```

Running R INLA for an augmented model The augmented model can be fitted using two different likelihoods for the real (Poisson) and pseudo-observations (Gaussian). The important feature here is the `same.as`-option, which allows to have the same ρ for each region.

```
#
# runPM(): function to fit an augmented model
# Y: matrix containing observations y_it and pseudo-observations, see Section 3
# E: offset m
# wm: matrix containing the weights (w_ji)^T
```

```
# p1: mean of the prior for \lambda and \rho
# p2: precision of the prior for \lambda and \rho
# start: initial values for hyperparameters
#
runPM <- function(Y, E, I, T, wm, p1, p2, start){

# code the index vector for components on the first stage - here: alpha
  alpha <- c(rep(1,T*I),rep(NA,T*I))

# code the index/weight vectors for components on the (first and) second stage
#      - here: zeta, zetatm1, rho
  codes <- coding(I,T,wm)
# index vector for zeta
  zeta <- codes[[1]]
# index vector for zetatm1
  zetatm1 <- codes[[2]]
# weights for zetatm1
  w.zetatm1 <- ifelse(is.na(zetatm1)==TRUE,NA,-1)
# index matrix for rho
  rho <- codes[[3]]
# weight matrix for rho
  w.rho <- codes[[4]]

# initialize the model formula
# use the "copy" feature to obtain an identical copy of the zeta process
# to estimate scaling parameters lambda (for zetatm1) and rho (for rho.1, rho.2,...)
# set the option fixed=FALSE
  f <- paste(deparse(substitute(Y)),"~f(zeta,model=\"iid\",fixed=TRUE,initial=-10)+
       +f(zetatm1,w.zetatm1,copy=\"zeta\",fixed=FALSE,initial=start[1],param=c(p1,p2))+
       +f(rho.1,w.rho.1,copy=\"zeta\",fixed=FALSE,initial=start[2],param=c(p1,p2))+alpha-1", sep="")

# split the weight matrix for rho in separate vectors for each region - here: region 1
  vn <- paste("rho", 1, sep=".")
  assign(vn, rho[,1])
  vn.w <- paste("w.rho", 1, sep=".")
  assign(vn.w, w.rho[,1])

# run a split loop over all columns (regions) of the index and weight matrix of rho
  for (i in 2:I){
    vn <- paste("rho", i, sep=".")
    assign(vn, rho[,i])
    vn.w <- paste("w.rho",i, sep=".")
    assign(vn.w, w.rho[,i])

# add a rho term for each region to the formula object
# to make sure that it's the same rho for each region, use option "same.as"
    f <- paste(f,"+f(rho.",i,",w.rho.",i,",copy=\"zeta\",same.as=\"rho.1\")",sep="")}

# convert the formula in a formula object
  f_final <- as.formula(f)

# call model - with two different likelihoods for the data and the pseudo-observations
  model <- inla(f_final, family=c("Poisson","Gaussian"), data=data.frame(alpha),
               E=E, control.data=list(list(),list(initial=start[3],param=c(0.1,0.001))),
               control.predictor=list(compute=1))

# return the result
  return(model)
}
```

An example The function `runPM()` can now be used to run a PM model for specific data and a known weight matrix. As an example, we show how model PM_2 can be fitted to the Coxiellosis data, 2004-2009, using the functions `coding()` and `runPM()`.

```
#
# wm.neighbourhood: matrix containing the neighbourhood information
# set up the Y2 matrix ((2*I*T)x2) containing observations and pseudo-observations
#
Y2 <- matrix(NA,nrow=2*T*I,2)
Y2[1:(I*T),1] <- data_cox[,1]
Y2[((I*T)+1):(2*T*I),2] <- 0
m2 <- c(data_cox[,2],rep(NA,I*T))
# fit model PM_2
PM_2_full <- runPM(Y2, m2, I, T, wm=wm.neighbourhood, p1=0, p2=4, start=c(1.6,0.03,0.2))
```

If a multivariate one-step-ahead prediction for time $T = 6$ (2009) should be made, the last timepoint for each region must be coded as `NA`.

```
# define a time vector
time <- c(rep(1:T,I),rep(0,I*T))
# set up Y3 matrix and code the last timepoint as NA for each region
Y3  <-  matrix(NA,nrow=(T+T)*I,2)
Y3[1:(I*T),1] <- data_cox[,1]
Y3[time==T,1] <- NA # important!
Y3[((I*T)+1):(2*T*I),2] <- 0
m3 <- m2
PM_2 <- runPM(Y3, m3, I, T, wm=wm.neighbourhood, p1=0, p2=4, start=c(1.6,0.03,0.2))
```

2.3 Calculation of scores for the PM models

The code in this section can be used to compute the scores discussed in Section 2 from the `R INLA` output. As mentioned above, multivariate one-step-ahead forecasts of the predictive distribution can be obtained by coding the observations at time T as `NA`.

In the following, the term y_{it} denotes the truly observed count at time t in region i. The vector $\boldsymbol{y}_{-T} = (y_{11}, \ldots, y_{1,T-1}, \ldots, y_{I1}, \ldots, y_{I,T-1})^T$ contains all observations in all regions i up to time $T-1$, i.e. the history at time T.

Calculating the predictive density $P(y_{iT}|\boldsymbol{y}_{-T})$ from INLA approximations of $\pi(\kappa_{iT}|\boldsymbol{y}_{-T})$ Suppose, $y_{it} \sim \text{Po}(m_{it} \cdot \exp(\eta_{it})) = \text{Po}(m_{it} \cdot \kappa_{it})$. We use the output obtained from a model, where the observations at time T, $\boldsymbol{y}_T = (y_{1T}, ..., y_{IT})^T$, were coded as `NA`. The predictive distribution of y_{iT} can always be written as

$$P(y_{iT}|\boldsymbol{y}_{-T}) = \int \pi(y_{iT}|m_{iT} \cdot \kappa_{iT}, \boldsymbol{y}_{-T})\ \pi(\kappa_{iT}|\boldsymbol{y}_{-T})\ d\kappa_{iT}.$$

A discretized approximation of the marginal of the linear predictor $\pi(\eta_{iT}|\boldsymbol{y}_{-T})$ at J nodes is obtained from INLA, when setting the option `control.predictor=list(compute=TRUE)`. This marginal can be transformed into a marginal of $\kappa_{iT} = \exp(\eta_{iT}|y_{-T})$ using the function
`inla.tmarginal()`. Using this output, we can approximate $P(y_{iT}|\boldsymbol{y}_{-T})$ by a finite sum

$$P(y_{iT}|\boldsymbol{y}_{-T}) \approx \sum_{j=1}^{J} \pi(y_{iT}|m_{iT} \cdot \kappa_{iT}^{(j)}, \boldsymbol{y}_{-T})\ \pi(\kappa_{iT}^{(j)}|\boldsymbol{y}_{-T})\ \Delta_j. \tag{1}$$

Calculating μ_P from INLA approximations of $\pi(\kappa_{iT}|\boldsymbol{y}_{-T})$ Furthermore, using the function `inla.emarginal()` we can obtain the mean of $\pi(\kappa_{iT}|\boldsymbol{y}_{-T})$. Applying the law of iterated expectations (*e.g.* Billingsley, 1986), we can then calculate μ_P as

$$\mu_P = \mathrm{E}(y_{iT}|\boldsymbol{y}_{-T}) = \mathrm{E}(\mathrm{E}(y_{iT}|m_{iT}\cdot\kappa_{iT},\boldsymbol{y}_{-T})) = \mathrm{E}(m_{iT}\cdot\kappa_{iT}|\boldsymbol{y}_{-T}). \tag{2}$$

```
#
# scoresPM(): function to obtain scores from INLA output
# m: the output of an INLA model, with the last year coded as NA
# data: a dataframe data <- data.frame(Y,m)
#
scoresPM <- function(m, T, I, data){

# load the required libraries
  library(INLA)
  library(caTools)

# define a time vector
  time <- c(rep(1:T,I),rep(0,I*T))

# obtain marginals of \kappa at times T via transformation of the output for \eta
  index <- which(time==T)
  tmarginals <- list(0)

  z <- 1
  for (i in index){
# important: function inla.tmarginal()
    tmarginals[[z]] <- inla.tmarginal(function(x)exp(x), m$marginals.fitted.values[[i]])
    z <- z+1
  }

# calculate mean of predictive distribution using formula (2)
  muP <- rep(0,I)

  for (i in 1:I){
# important: function inla.emarginal()
    muP[i] <- inla.emarginal(function(x) x, tmarginals[[i]])
  }

# calculate ses
  ses <- (data[time==T,1]-muP*data[time==T,2])^2

# calculate logs
  logs <- rep(0,I)

# run a loop for the last timepoint T in each region i
  for(i in 1:I){
# extract the respective marginals of kappa
    x <- tmarginals[[i]][,1]
    y <- tmarginals[[i]][,2]

# extract the number of observed cases and the offset
    Y <- data[time==T,1][i]
    E <- data[time==T,2][i]

# approximate sum (1) using the trapezoidal rule
    f <- 0
    for (j in 1:(length(x)-1)){
      f <- f+dpois(Y,((x[j]+x[j+1])/2)*E)*trapz(x[j:(j+1)],y[j:(j+1)])}
    logs[i] <- -log(f)
  }
```

```
# calculate rps
  rps <- rep(0,I)

# run a loop for the last timepoint T in each region i
  for(i in 1:I){

# extract the respective marginals of \kappa
    x <- tmarginals[[i]][,1]
    y <- tmarginals[[i]][,2]

# extract the number of observed cases and the offset
    Y <- data[time==T,1][i]
    E <- data[time==T,2][i]

# initialize a vector for P(Y=k)
    kmax <- 30
    vec.kk <- rep(0,kmax+1)

# run a loop over all k
    for (k in 0:kmax){
      f <- 0
# calculate P(Y=k) using formula (1)
      for (j in 1:(length(x)-1)){
        f <- f+dpois(k,((x[j]+x[j+1])/2)*E)*trapz(x[j:(j+1)],y[j:(j+1)])}
      vec.kk[k+1] <- f
    }

# evaluate RPS
    for(k in 0:kmax){
      rps[i] <- rps[i]+(sum(vec.kk[1:(k+1)])-(as.numeric(Y<=k)))^2
    }
  }

# return the mean scores
  return(list(mean(ses),mean(logs),mean(rps)))
}
# compute scores
scores <- scoresPM(PM_2, T, I, data_cox)
```

2.4 Calculation of scores for the H^3 models

As mentioned in Section 1 of the main manuscript, the software to perform maximum likelihood inference for H^3 models is available from `https://r-forge.r-project.org/projects/surveillance`. This webpage also offers a general introduction to the usage of the software (`vignette_hhh4.pdf`). Methodological details on the inference are described in Paul and Held (2011).

In the following, we show the R code to fit model H_2^3.

```
#
# hhh4(): function to run (penalized) maximum likelihood inference for H^3 models
# oneStepAhead(): function to obtain successive one-step-ahead predictions
# scores(): function to calculate predictive scores
#
library(surveillance)
source("algo_hhh4.R")
source("scores.R")
# create an object of class disProg
```

```
dP <- create.disProg(week=1:T, observed=matrix(data_cox[,1],nrow=T),
                  state= matrix(0,T,I), neighbourhood=wm.neighbourhood)
# convert disProg object to sts object
sts <- disProg2sts(dP)
# run model H^3_2
# use the adjacency-based weights (wm.neighbourhood)
H3_2 <- hhh4(sts,control=list(ar = list(f = ~ 1), ne = list(f = ~ 1, weights=wm.neighbourhood),
                end = list(f = ~ 1, offset=matrix(data_cox[,2],ncol=I,nrow=T)),
                family = "NegBin1",verbose = 1))
# compute one-step-ahead predictions for each region using oneStepAhead()
pred.H3_2 <- oneStepAhead(H3_2,tp=T-1)
# calculate the mean scores using scores()
scores.H3_2 <- scores(pred.H3_2, individual=T)
meanscores.H3_2 <- apply(scores.H3_2, 2, mean)
```

References

Billingsley, P. (1986). *Probability and Measure*. John Wiley & Sons, New York.

Lütkepohl, H. (2005). *New Introduction to Multiple Time Series Analysis*. Springer, Berlin.

Paul, M. and Held, L. (2011). Predictive assessment of a non-linear random effects model for multivariate time series of infectious disease counts. *Statistics in Medicine* **30,** 1118–1136.

Ruiz-Cardenas, R., Krainski, E. T., and Rue, H. (2010). Fitting dynamic models using integrated nested Laplace approximations - INLA. Technical report, NTNU Trondheim, Norway.