Dengue Dynamics

Krzysztof Sakrejda

June 17, 2016

Abstract

/noindent There's lots of important stuff I address. It is important. Evidence evidence, conclusion.

(Ch. 1) . . .

I will address these questions using ¡statement of methods goes here¿.

```
library(knitr)
read_chunk(file.path(R_dir,'shared_data.R'))
read_chunk(file.path(R_dir,'approximation-plot.R'))
```

```
library(grid)
library(ggplot2)
library(dplyr)
library(RPostgreSQL)
library(integrator)
library(cruftery)
```

```
link <- db_connector('~/credentials/pgsql-pass-dengue-local-db.rds')

## Warning:  Was not able to allocate a new connection, returned
## connection has lost its previously pending result.
```

1 The SIR model can be restated to ignore susceptibles.

The SI model is a simplification of the SIR model for a disease where infectious individuals leave the infectious pool after one time step.

1.1 SI case

The SI model can be stated as a pair of equations describing the size of the susceptible (S) and infected (I) population at time t in terms of the system state at time $t-1$ and three groups of parameters: 1) r_t, which describes the infectiousness of the disease at time t; 2) α_1, and α_2 which describe how effectively the infected interact with the susceptible population and how effectively the susceptible population mixes with the infected population; and 3) the error terms ϵ_t and u_t. The model is stated in equation 1.

$$I_t = r_t I_{t-1}^{\alpha_1} S_{t-1}^{\alpha_2} \epsilon_t \tag{1}$$

$$S_t = B_{t-d} + S_{t-1} - I_t + u_t \tag{2}$$

One of the difficulties of applying this model is that the state variables are not observed directly. Only a case count (C) is observed which is related to the number infected at time t through a reporting fraction ρ_t as $I_t = \rho_t C_t$. For estimation and to understand the confounding which results from these limited observations it would be useful to state the model in terms of infected individuals only, leaving aside the dynamics of the susceptible population. According to CITE CITE CITE this can be accomplished. First we begin by shifting the susceptible equation one time step back and using it to replace the S_{t-1} term in the infectious equation:

$$I_t = r_t I_{t-1}^{\alpha_1} \left[B_{t-1-d} + S_{t-2} - I_{t-1} + u_{t-1} \right]^{\alpha_2} \epsilon_t \tag{3}$$

$$\tag{4}$$

Now the only term left for susceptibles is S_{t-2} which we can replace by shifting the infectious equation back one time step and isolating S on one side.

$$S_{t-1}^{\alpha_2} = \frac{I_t}{r_t I_{t-1}^{\alpha_1} \epsilon_t} \tag{5}$$

$$\alpha_2 \log S_{t-1} = \log I_t - \log r_t - \alpha_1 \log I_{t-1} - \log \epsilon_t \tag{6}$$

$$\log S_{t-1} = \frac{\log I_t - \log r_t - \alpha_1 \log I_{t-1} - \log \epsilon_t}{\alpha_2} \tag{7}$$

$$\log S_{t-2} = \frac{\log I_{t-1} - \log r_{t-1} - \alpha_1 \log I_{t-2} - \log \epsilon_{t-1}}{\alpha_2} \tag{8}$$

This new representation of S refers only to lagged terms of I and parameters. It can be exponentiated and used to replace S_{t-2} in the infectious equation.

$$I_t = r_t I_{t-1}^{\alpha_1} \left[B_{t-1-d} + \exp\left(\frac{\log I_{t-1} - \log r_{t-1} - \alpha_1 \log I_{t-2} - \log \epsilon_{t-1}}{\alpha_2} \right) - I_{t-1} + u_{t-1} \right]^{\alpha_2} \epsilon_t \tag{9}$$

$$\tag{10}$$

Using the log of this equation can clarify the role of different terms:

$$\log I_t = \log r_t + \alpha_1 \log I_{t-1} + \tag{11}$$

$$\alpha_2 \log \left[B_{t-1-d} + \exp\left(\frac{\log I_{t-1} - \log r_{t-1} - \alpha_1 \log I_{t-2} - \log \epsilon_{t-1}}{\alpha_2} \right) - I_{t-1} + u_{t-1} \right] + \log \epsilon_t \tag{12}$$

The multiplicative error terms appear in log form everywhere so their expectation will be zero. In the exponentiated term the (log) ratio of infected individuals in two previous time steps appears. Though the equation is stated in terms of infected counts it could equally be stated using the reporting equation and case counts rather than true counts.

1.2 Multi-strain non-interacting SI

A similar transformation is also possible for a multi-strain SI model with non-interacting strains. In this case we need to keep track of strain-specific susceptibility and infection, which we do by subscripting with one bit per strain. An individual susceptible to all strains is denoted as S_{0000} and an individual immune to all but the third strain is denoted as S_{1101}. An individual susceptible to to the third strain but with unknown status with regards to all others is denoted as S_{xx0x}. This last notation refers to a sum:

$$S_{(0xxx),t} = S_{(0000),t} + S_{(0100),t} + S_{(0010),t} + \tag{13}$$
$$+ S_{(0001),t} + S_{(0110),t} + S_{(0011),t} + \tag{14}$$
$$+ S_{(0101),t} + S_{(0111),t} \tag{15}$$

With this notation in hand we can write the dynamics equations following the example of the SI model.

$$I_{(1xxx),t} = r_{1,t} I^{\alpha_1}_{(1xxx),t-1} S^{\alpha_2}_{(0xxx),t-1} \epsilon_t \tag{16}$$
$$S_{(0xxx),t} = B_{t-d} + S_{(0xxx),t-1} - I_{(1xxx),t} + u_t \tag{17}$$

As before, I_t is observed—at least in serotype-specfic data—and the goal is to substitute out S_t in all equations. Immediately we notice that indexing by strain susceptibility in the case of non-interacting strains does not affect equation structure and we can recycle the solution from original single-strain case.

$$\log I_{(1xxx),t} = \log r_{1,t} + \alpha_1 \log I_{(1xxx),t-1} + \tag{18}$$
$$\alpha_2 \log \left[B_{t-1-d} + \exp\left(\frac{\log I_{(1xxx),t-1} - \log r_{1,t-1} - \alpha_1 \log I_{(1xxx),t-2} - \log \epsilon_{t-1}}{\alpha_2} \right) - I_{(1xxx),t-1} + u_{t-1} \right] \tag{19}$$

Ultimately the change is trivial because there are no interactions and we are monitoring an important state variable.

1.3 Multi-strain interacting SI

As a next step, we introduce interactions into the multi-strain SI model. In this model the immune history of the individualis allowed to affect the infection rate. Due to this change we can no longer lump all susceptible individuals together in a sum term which complicates the process of removing susceptibles from the equation. Conveniently, the infected individuals still function only as a group regardless of their previous infection status.

$$I_t = r_t I_{t-1}^{\alpha_1} S_{t-1}^{\alpha_2} \epsilon_t \tag{20}$$

$$S_t = B_{t-d} + S_{t-1} - I_t + u_t \tag{21}$$

2 A generic error structure for continuous-time dynamics.

Real-time data commonly arrives with a variety of associated time scales. Working with continuous time provides two key advantages for model building. Decisions about how to align varying time frames can be deferred until later so that they do not drive the choice of model structure.

2.1 Poisson process basics

When describing a dynamical system we start with an arbitrary but smooth intensity function which describes the instantenous rate of events (e.g.- cases per day). We are interested in producing a model for the intensity function, written $\lambda(t)$, which describes this rate as a function of time.

One obstacle we face is that we do not measure the intensity directly, instead we measure the number of cases, Y, arriving between time s and time t. To be exact we choose to use a half-open interval, $(s, t]$, to describe the period of time. Using the half-open interval simplifies future calculations and data preparation by making it possible to combine intervals without calculating overlap and to assign data unambiguously to intervals. The complete notation for a measurement is then $Y_{s,t,i}$ which indicates the count (Y), the ends of the interval (s, t), as well as an index for additioal covariates measured at the batch level.

Homogenous Poisson process approximation

In the standard application of the homogenous Poisson process, the parameter, m, for the Poisson distribution can be calculated as the product of the fixed rate, λ_i for batch i, and the duration of time $t - s$. This is can be thought of as a Poisson regression where $t - s$ functions as the exposure adjustment. If λ is allowed to vary by batch, this can describe a time-varying process, but at the cost of discontinuities.

```r
time <- 1:10
rate <- seq(from=11, to=20, length.out=length(time))
n_per_batch <- 5
data <- data.frame(
        time = rep(time,n_per_batch),
        rate = rep(rate,n_per_batch)
)
data[['count']] <- rpois(n=nrow(data), lambda=data[['rate']])
data <- data[order(data[['time']]),]

f <- function(lambdas) {
        data[['lambda']] <- lambdas[data[['time']]]
        ll <- sum(dpois(x=data[['count']], lambda=data[['lambda']], log=TRUE))
        return(-ll)
}

optzd <- optim(
        par = 12:21,
        fn = f, gr=NULL,
        method = c("L-BFGS-B"),
        lower=10^-10, upper=50
)

g <- function(t) optzd[['par']][floor(t)+1]

ptzd <- data.frame(
        x = 0:9+.5,
        y = g(0:9+0.5)
)

library(ggplot2)
pl <- ggplot(
        data=data,
        aes(x=time-0.5, y=count)
) + geom_point() +
                geom_bar( data=ptzd, aes(x=x,y=y), alpha=0.4, stat='identity')
```

```
saveRDS(object=pl, file=file.path(figure_dir,'discrete-homo-poisson-process.rds'))
```

```
pl <- readRDS(file=file.path(figure_dir, 'discrete-homo-poisson-process.rds'))
print(pl)
```

As long as data is collected over well defined periods which match the biological process being observed, the discontinuities are unimportant. Sometimes, collection times vary and they may not be timed to changes in the evolution of the biological process or the biological process may be very dynamic as well as continuous which makes it difficult to time data collection such that per-batch rate estimates are biologically meaningful.

Inhomogenous Poisson process (IHPP)

One way to avoid introducing the ambiguities that come with discrete parameter batches is to use a continuous formulation of the Poisson process which allows for a time-varying rate. For any particular count, $Y_{s,t,i}$, there is still a single paramter m, but now the rate over the interval $(s, t]$ varies so we can no longer find the area, m, by multiplying the rate by the interval duration. Instead we define the intensity (or rate) function $\lambda(t)$ over the course of the experiment. To find the area $m_i(s, t)$ we use integration:

$$m_i(s,t) = \int_s^t \lambda_i(x)dx \tag{22}$$

Each count from a batch is still modeled as a Poisson distributed random variable with parameter $m_i(s, t)$.

```
library(cruftery)
data <- data.frame(
        x=seq(from=0, to=10, length.out=10^3),
        y=gaussian_spline(x=seq(from=0, to=10, length.out=10^3),
        knot_points=2:8, knot_weights=exp(rnorm(7))*2+c(0,2,2,2,rep(0,3)), knot_scale=0.5)
)
shade <- data[data[['x']] >= 3 & data[['x']] < 5,]
shade <- rbind(
        data.frame(x=3,y=0),
        shade,
```

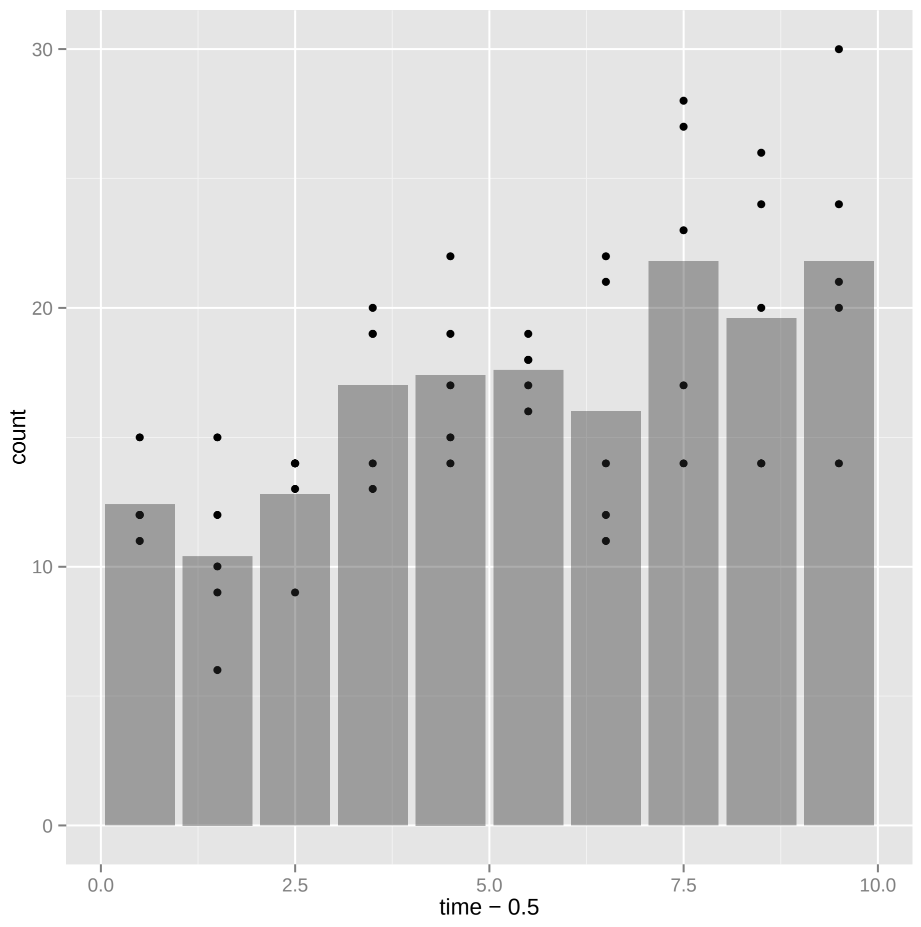

Figure 1: Counts drawn from a Poisson distribution with time-varying rate. Shaded areas describe the estimated probability mass function. When boundaries between batches are biologically relevant this model is appropriate, but if boundaries are fluid the interpretability of the parameter at the boundary is limited.

```
        data.frame(x=5,y=0)
)


pl <- ggplot(data=data, aes(x=x, y=y)) + geom_line() +
                        geom_polygon(data=shade, aes(x=x, y=y), alpha=0.3) +
                        xlab('Time (days)') + ylab('Intensity (Cases per day)') +
                        annotate("text", x=3, y=-.2, label="s") +
                        annotate("text", x=5, y=-.2, label="t") +
                        annotate("text", x=4, y= .8, label=
                                "m(s,t)==integral(lambda(x)*dx,s,t)", parse=TRUE)
```

```
saveRDS(object=pl, file=file.path(figure_dir,'cont-inhomo-poisson-process.rds'))
```

```
pl <- readRDS(file=file.path(figure_dir, 'cont-inhomo-poisson-process.rds'))
print(pl)
```

The inhomogenous poisson process makes an effective bridge between a process conceptualized as continuous and real data collected on varying time scales in a discrete fashion. The focus of any further effort can be on the function $\lambda(t)$ which is the rate of cases per unit time. The remaining difficulty is choosing a form of $\lambda(t)$ which is easy to integrate and allows us to apply the usual ameneties of statistics—design matrices and linear models.

2.2 Poisson process implementation

The two conditions on $\lambda(t)$ are that it should be easy to integrate and that is should be amenable to further modeling. For this second reason, we choose to use a semiparameteric radial spline. We spread K knot points over the time period covered by the data and use the normal distribution as a basis function. The basis set is then the set of normals centered on each knot point. The function $\lambda(t)$ then becomes a weighted sum of K normals.

$$\lambda(x) = \sum_{k=1}^{K} \beta_k \frac{1}{\sqrt{2\pi\sigma^2}} \exp(\frac{1}{2\sigma^2}\,(x-\mu_k)^2) \tag{23}$$

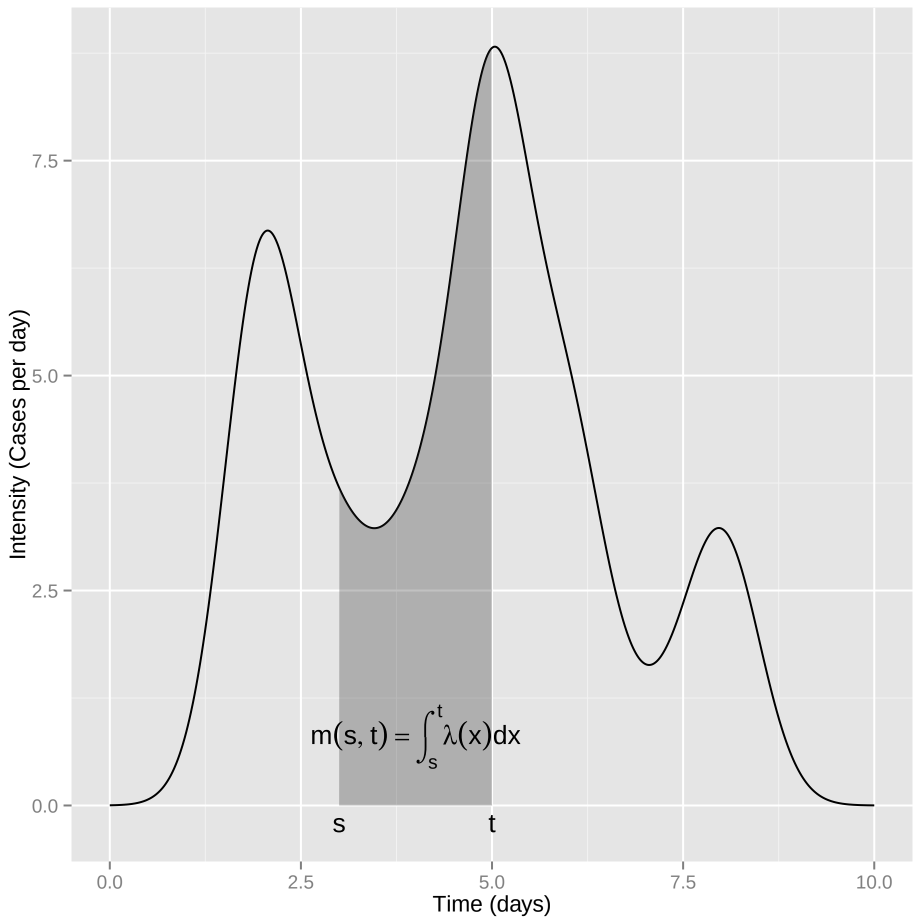

Figure 2: Counts drawn from a Poisson distribution with time-varying rate. The curve is the intensity function in units of cases per day, which is integrated to give the Poisson distribution parameter.

This formulation allows us to further model the weights β_k to describe the effects relevant to a specific portion of time without introducing discontinuities in the intensity function while still making the function very flexible. Formulating the intensity function as a sum of normal densities also makes analytical integration possible as long as the function has a known CDF. Abusing notation slightly, the calculation of $m(s,t)$ becomes:

$$m(s,t) = \int_s^t \sum_{k=1}^{K} \beta_k PDF_N(x, \mu_k, \sigma^2)dx \tag{24}$$

$$= \sum_{k=1}^{K} \beta_k \left(CDF_N(t, \mu_k, \sigma^2) - CDF_N(s, \mu_k, \sigma^2) \right) \tag{25}$$

Since calculations of many CDF's, including the normal, are efficiently coded in many libraries this is a generic strategy for applying the IHPP to a variety of problems.

3 Count data simulation for a continuous-time dynamics.

For a simulation where the timing of events wihin batches is not important but the number of events in a batch is important, it is possible to avoid dealing with timing entirely and simply simulate from the Poisson process conditional on $m(s,t)$ for each batch. The key point for such a simulation becomes the calculation of $m(s,t)$.

To simulate event timing a thinning (rejection) algorithm is available which and its efficiency depends on how well the proposed event distribution matches the function $\lambda(t)$ as used for simulation or estimation.

3.1 Splines for generating a smooth function

To evaluate models constructed based on this Poisson process formulation it will be important to have a way of simulating from the Poisson process as described above and evaluating our ability to recover parameters. The appropriate splines and their integrals can be constructed from a small set of R functions which implement the calculations described in section **??**.

```
read_chunk('~/packages/cruftery/package_dir/R/spline_functions.R')
```

For any spline the function value at point x can be described as a function of the basis function. The rest of the calculations are generic.

```
# Generic radial spline:
radial_spline <- function(x, f, knot_points, knot_weights, knot_scale) {
  if (length(x) == 1) {
    knot_contributions <- f(x=x, center=knot_points, scale=knot_scale)*knot_weights
    return(sum(knot_contributions))
  } else {
    return(sapply(x,radial_spline, f=f, knot_points=knot_points, knot_weights=knot_weights,
  }
}
```

The generic spline function can be specialized for a gaussian spline as follows.

```
# Generic radial spline specialized to Gaussian.
gaussian_spline <- function(x, knot_points, knot_weights, knot_scale) {
  f <- function(x, center, scale) dnorm(x=x, mean=center, sd=scale)
  return(radial_spline(x=x, f=f, knot_points=knot_points, knot_weights=knot_weights, knot_sc
}
```

The integral can be coded in terms of R's CDF functions.

```
# Integral of Gaussian radial spline.
gaussian_spline_integral <- function(knot_points, knot_weights, knot_scale, a, b) {
  contributions <- mapply(
    FUN=function(point, weight, scale, a, b) {
      weight * (pnorm(q=b, mean=point, sd=scale) - pnorm(q=a, mean=point, sd=scale))
    },
    point = knot_points, weight = knot_weights,
    MoreArgs = list(scale=knot_scale, a=a, b=b)
  )
  return(sum(contributions))
}
```

Given the spline integral function, it is simple to calculate $m(s,t)$ and simulate the number of events in a batch as:

$$X \sim Poi(m(s,t)) \tag{26}$$

4 Reporting in the Poisson process context.

4.1 Modeling observed reporting data.

The reporting delays, t_d, are observed. These can be modelled directly as, for example, having a weibull distribution. To estimate such a model we would:

1. load all data with the columns:
    - t_r, the date of the report delivery.
    - t_s, the date of illness.
    - $t_d = t_r - t_s$, the reporting delay.
    - j, the province.

   Weibull model in Stan..., etc...
   This lets us estimate parameter for a posterior density $p(t_d|\theta_d)$.

4.2 The reporting problem in the Poisson process model.

The question we want to answer to integrate the reporting model with a disease dynamics model is:

Q1: What proportion of cases which occur between time point t_a and t_b are reported by t^*.

Knowing this fraction, $f(t^*, t_a, t_b)$, would let us convert y, the case count observed between time points t_a and t_b, into y^*, the true (or reporting-adjusted) case count.

$$y = y^* \times f(t^*, t_a, t_b) \tag{27}$$

It is likely that this is some kind of CDF for t^*, but it's form is unclear. To get there we start by asserting that we will specify two densities: one for the reporting/delay process and one for the disease process:

$$p_d(t_d|\theta_d) = \dots \tag{28}$$

$$p_s(t_s|\theta_s) = \dots \tag{29}$$

The disease process is a primary focus of this work and the poisson process model can be used to estimate an intensity function as described. The density function for the dates of illness (t_s) is (the same?) a transformation of the intensity function.

The reporting delays are directly observed and standard time-to-event modeling techniques can be applied to estimate the parameters of the density of t_d. Due to the deterministic relationship between t_r and t_s, we can specify a distribution for t_r conditional on t_s.

$$p_r(t_r|\theta_d, t_s) = p_d(t_r - t_s|\theta_d) \tag{30}$$

To remove the conditionality on t_s, we can construct the joint distribution for $p_{r,s}$ and integrate over the possible values. In the poisson process, counts are reported in batches which begin at a and end at b restricting $t_a < t \leq t_b$, and these are the relevant bounds for integration:

$$p_{r,s}(t_r, t_s|\theta_d, \theta_s) = p_s(t_s|\theta_s) p_r(t_r|\theta_d, t_s) \tag{31}$$

$$p_r(t_r|\theta_d, \theta_s, a < t \leq b) = \int_a^b p_s(t_s|\theta_s) p_r(t_r|\theta_d, t_s) \tag{32}$$

The resulting density is depends only on the parameters of the reporting model (θ_d), the disease model (θ_s), and the boundaries of the batch ($t \in [a, b)$). This density defines the distribution of reporting times for a batch of cases. For cases to be available for a model, they must be reported prior to, t^*, the time of the analysis. The proportion of available cases can be stated in terms of the CDF:

$$Pr[t_r < t^*|\theta_d, \theta_s, a < t \leq b] = \int_a^{t^*} p_r(t_r|\theta_d, \theta_s, a < t <= b) dt_r \tag{33}$$

Which lets us relate y and y^*.

$$y = y^* \times Pr[t_r < t^*|\theta_d, \theta_s, a < t \leq b] \tag{34}$$