Basic SIR fitting

March 15, 2017

1 Harbin

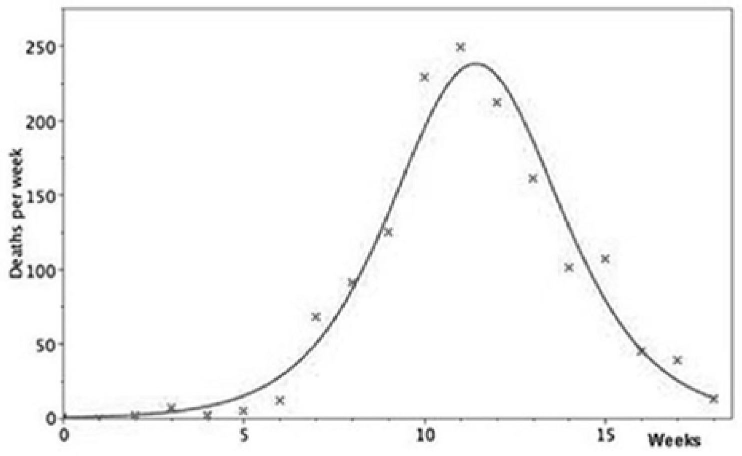


Figure 1: Unnumbered figure (p. 102) from Dietz (2009) showing the Harbin epidemic.

Figure 1 shows a Kermack-Mckendrick model fit to Harbin plague data. Based on the equations (1) and estimates ("$x_0 = 2985$, $\mathcal{R}_0 = 2.00$ and a mean infectious period of 11 days") that Dietz (2009) provides, we can compare how Kermack-Mckendrick model fit differs from SIR model fit based on maximum likelihood estimation.

$$\begin{aligned}\frac{dz}{dt} &= \frac{\gamma x_0}{2\mathcal{R}_0^2} c_1 \mathrm{sech}^2(c_1\gamma t - c_2),\\ c_1 &= \sqrt{(\mathcal{R}_0 - 1)^2 + \frac{2\mathcal{R}_0^2}{x_0}}\\ c_2 &= \tanh^{-1}\left(\frac{\mathcal{R}_0 - 1}{c_1}\right).\end{aligned} \tag{1}$$

We note that the original equation provided by Dietz (2009) contains a typo. $c_1\gamma t$ after sech^2 in the first equation should be corrected to $c_1\gamma t/2$ (Kermack and McKendrick, 1927).

First, load the pacakge:

```
library(fitsir)
```

Since `fitsir` package lazy loads harbin data, `data(harbin)` syntax is unnecessary.

```
head(harbin)

##   week Deaths
## 1    2      2
## 2    3      7
## 3    4      2
## 4    5      6
## 5    6     12
## 6    7     68
```

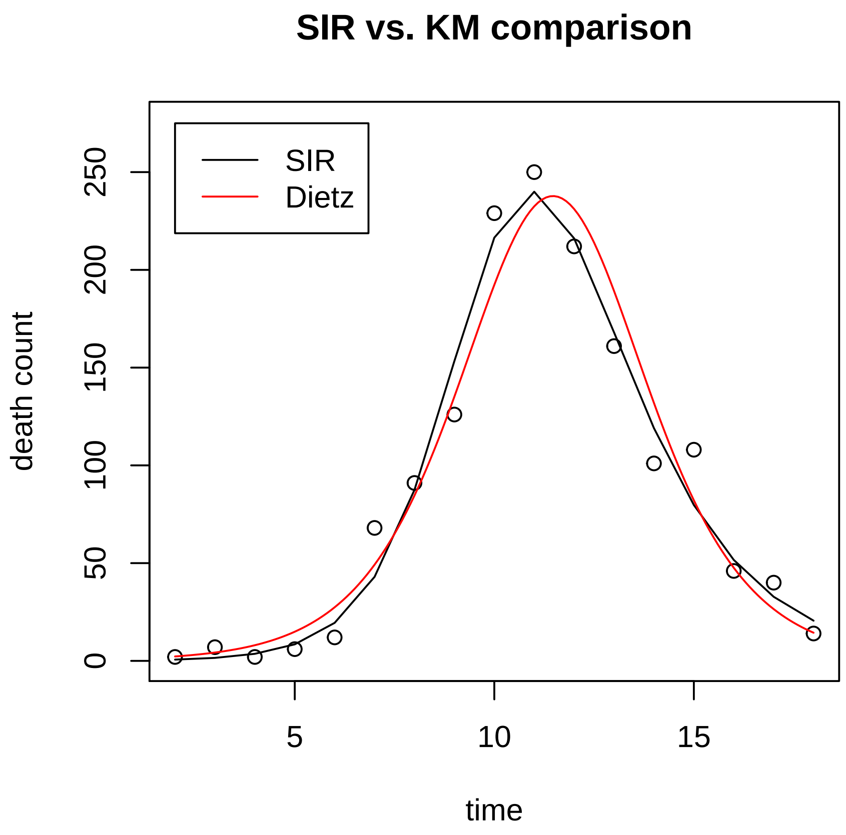
Then, we transform the parameters provided by Dietz (2009) into *unconstrained parameters* (`log.beta, log.gamma, log.N, logit.i`) so that they can be used as starting values for `fitsir`. Although `fitsir` expects a dataframe with column names `times` and `count`,

```
dietz_harbin <- c(x0=2985,rzero=2,gamma=7/11)
dietz_lpars <- with(as.list(dietz_harbin),
      c(log.beta=log(rzero*gamma),
        log.gamma=log(gamma),
        log.N=log(x0),
        logit.i=qlogis(1e-3)))
(ff <- fitsir(harbin, start=dietz_lpars, type="death",
              tcol="week", icol="Deaths", method="BFGS"))

##
## Call:
## mle2(minuslogl = objfun, start = start, method = method, data = dataarg,
##     vecpar = TRUE, gr = gradfun, control = control)
##
## Coefficients:
##   log.beta  log.gamma      log.N    logit.i
##  0.4868478 -0.2708639  7.4966582 -8.1274230
##
## Log-likelihood: -68.27
```

Plot it:

```
plot(ff, main="SIR vs. KM comparison")
times <- with(as.list(harbin), seq(min(week), max(week), by = 0.1))
dpKM <- with(as.list(dietz_harbin),
        {
           c1 <- sqrt((rzero-1)^2+2*rzero^2/x0)
           c2 <- atanh((rzero-1)/c1)
           gamma*x0/(2*rzero^2)*c1*
               (1/cosh(c1*gamma*times/2-c2))^2
        })
lines(times,dpKM, col = 2)
legend(x=2, y=275, legend=c("SIR","Dietz"), col=c("black", "red"), lty = 1)
```



Summarize it:

```
summary(ff)

## Maximum likelihood estimation
```

```
##
## Call:
## mle2(minuslogl = objfun, start = start, method = method, data = dataarg,
##     vecpar = TRUE, gr = gradfun, control = control)
##
## Coefficients:
##                    R0          r     infper          i0          I0
## Estimate   2.1334e+00 8.6446e-01 1.3111e+00 2.9524e-04 5.3203e-01
## Std. Error 5.4710e-01 1.0344e-01 4.9281e-01 1.2155e-04 2.6501e-01
##                    S0       N
## Estimate   1.8015e+03 1802.01
## Std. Error 2.6259e+02  262.77
##
## -2 log L: 136.5431
```

MLE returns higher $\mathcal{R}_0$ and longer infectious period but estimates lower population size.

Overdispersion?

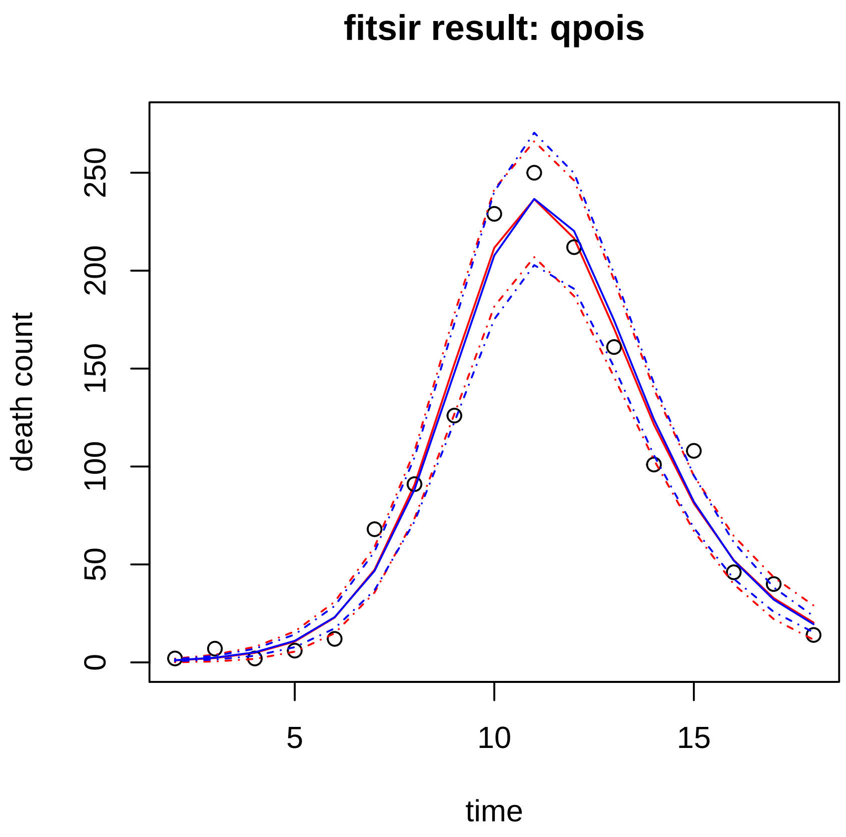
```
(ff2 <- fitsir(harbin, dist="qpois", type="death",
              tcol="week", icol="Deaths", method="BFGS"))

##
## Call:
## mle2(minuslogl = objfun, start = start, method = method, data = dataarg,
##     vecpar = TRUE, gr = gradfun, control = control)
##
## Coefficients:
##   log.beta  log.gamma      log.N    logit.i
##  0.5372979 -0.0919146  7.5927621 -7.9344532
##
## Log-likelihood: -71.74

(ff3 <- fitsir(harbin, dist="nbinom", type="death",
              tcol="week", icol="Deaths"))

##
## Call:
## mle2(minuslogl = objfun, start = start, method = method, data = dataarg,
##     vecpar = TRUE, gr = gradfun, control = control)
##
## Coefficients:
##    log.beta   log.gamma       log.N     logit.i
##  0.62002541  0.08260534  7.67718430 -8.06667616
##
## Log-likelihood: -67.67
```

```r
plot(ff2, level=0.95, col.traj="red", col.conf="red")
plot(ff3, level=0.95, add=TRUE, col.traj="blue", col.conf="blue")
```



Its also worth noting them negative binomial fits slightly better than Gaussian (we still need to deal with profiled parameters somehow...):

```r
AIC(ff3) - AIC(ff)

## [1] -1.202405

summary(ff3)

## Maximum likelihood estimation
##
## Call:
## mle2(minuslogl = objfun, start = start, method = method, data = dataarg,
##     vecpar = TRUE, gr = gradfun, control = control)
```

```
##
## Coefficients:
##                   R0          r     infper         i0          I0
## Estimate   1.7116e+00 7.7286e-01 9.2071e-01 3.1373e-04 6.7719e-01
## Std. Error 6.5332e-02 3.3701e-02 5.0720e-02 5.8763e-05 1.4405e-01
##                   S0       N
## Estimate   2.1579e+03 2158.53
## Std. Error 1.4400e+02  144.08
##
## -2 log L: 135.3407
```

2 Philadelphia

```
head(phila1918)

##         date pim
## 1 1918-09-01   3
## 2 1918-09-02   7
## 3 1918-09-03   5
## 4 1918-09-04   1
## 5 1918-09-05   3
## 6 1918-09-06   4
```

Transform...

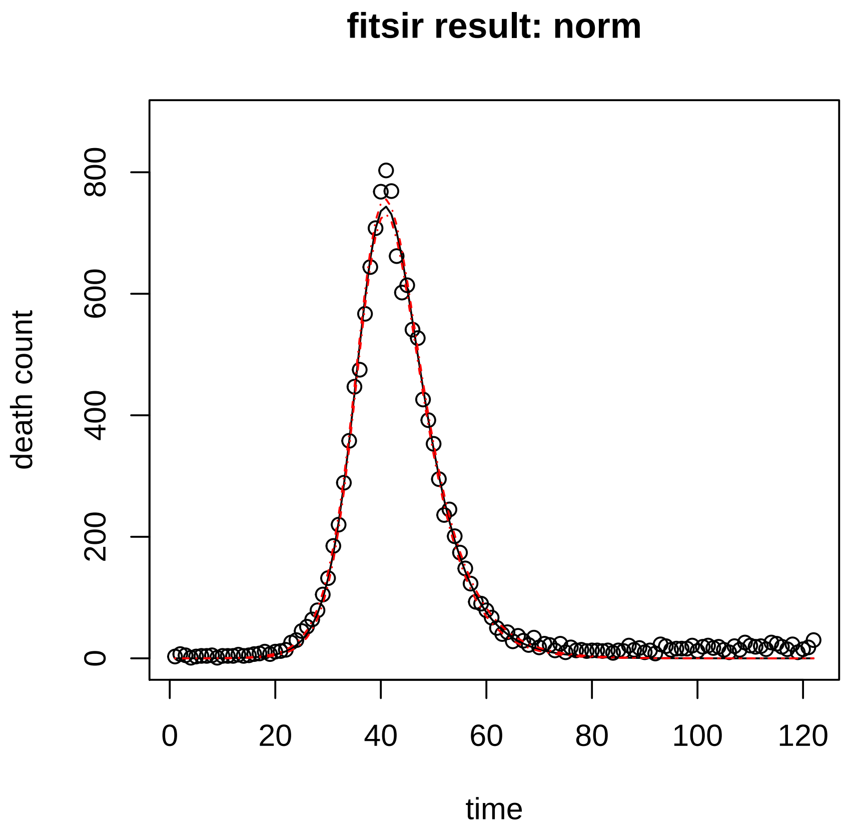
```
phila1918a <- data.frame(times=1:nrow(phila1918), count=phila1918$pim)
```

Naive fitting

```
(philafit <- fitsir(phila1918a, method="BFGS", type="death"))

##
## Call:
## mle2(minuslogl = objfun, start = start, method = method, data = dataarg,
##     vecpar = TRUE, gr = gradfun, control = control)
##
## Coefficients:
##    log.beta   log.gamma       log.N     logit.i
##  -0.5796543  -1.4135201   9.6186667 -12.5577537
##
## Log-likelihood: -512.18

plot(philafit, level=0.95)
```



Summarize it:

```
summary(philafit)

## Maximum likelihood estimation
##
## Call:
## mle2(minuslogl = objfun, start = start, method = method, data = dataarg,
##     vecpar = TRUE, gr = gradfun, control = control)
##
## Coefficients:
##                    R0           r      infper          i0          I0
## Estimate   2.3022e+00 3.1681e-01 4.1104e+00 3.5175e-06 5.2914e-02
## Std. Error 1.3169e-01 8.6596e-03 3.1472e-01 7.4630e-07 1.2466e-02
##                    S0        N
## Estimate   1.5043e+04 15042.98
## Std. Error 4.0145e+02   401.46
```

```
##
## -2 log L: 1024.357
```

Does this look reasonable?

We have to deal with overdispersion somehow...

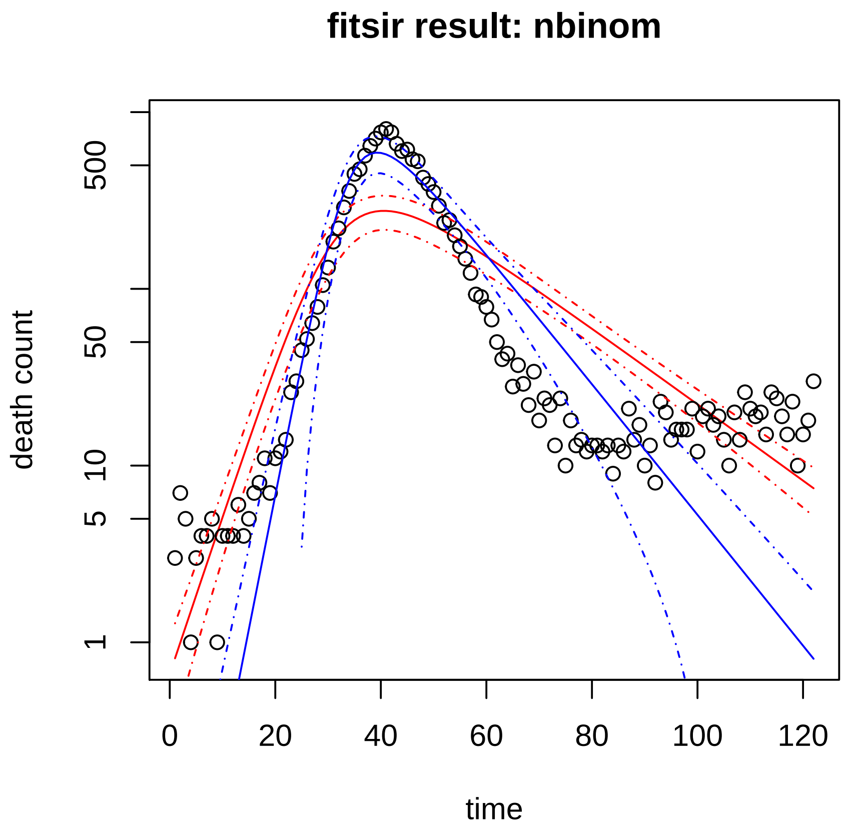
```
(philaQP <- fitsir(phila1918a, dist="qpois", method="BFGS", type="death"))

##
## Call:
## mle2(minuslogl = objfun, start = start, method = method, data = dataarg,
##     vecpar = TRUE, gr = gradfun, control = control)
##
## Coefficients:
##    log.beta   log.gamma       log.N     logit.i
##  -0.8246981  -2.4249433   9.5502483 -12.0299160
##
## Log-likelihood: -1887.65

(philaNB <- fitsir(phila1918a, dist="nbinom",  type="death"))

##
## Call:
## mle2(minuslogl = objfun, start = start, method = method, data = dataarg,
##     vecpar = TRUE, gr = gradfun, control = control)
##
## Coefficients:
##  log.beta log.gamma     log.N   logit.i
## -1.358728 -2.959870  9.324681 -6.676200
##
## Log-likelihood: -583.3

plot(philaNB, level=0.95, col.traj="red", col.conf="red", log="y")
plot(philaQP, level=0.95, add=TRUE, col.traj="blue", col.conf="blue")
```



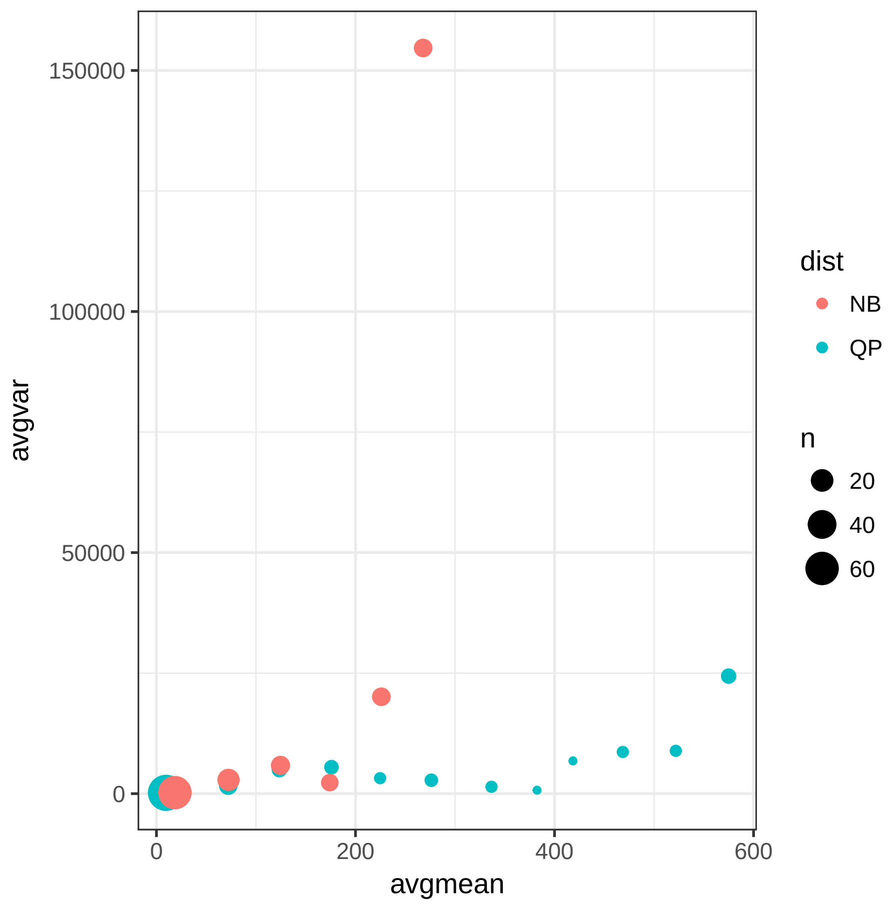
Negative binomial fits way better than Poisson. Can we look at mean-variance relationship?

```
library(dplyr)
library(ggplot2); theme_set(theme_bw())
mvrel <- function(fit) {
    mean <- SIR.detsim(phila1918a$times, coef(fit, "trans"), type="death")
    data.frame(
        mean=mean,
        var=(phila1918a$count-mean)^2
    )
}
level <- seq(0, 600, by = 50)

mvfun <- . %>%
    mvrel %>%
    mutate(group=cut(mean, breaks=level)) %>%
    group_by(group) %>%
    summarise(avgmean=mean(mean), avgvar=mean(var), n=length(var))

mvtot <- list(QP=mvfun(philaQP), NB=mvfun(philaNB)) %>%
    bind_rows(.id="dist")

ggplot(mvtot, aes(avgmean, avgvar, size=n, col=dist)) +
    geom_point()
```



I don't think this really is a problem with over dispersion... Long tails are definitely affecting both quasipoisson and negative binomial fits..

2.1 Bombay

I want to explain multiple local minima here..

```
bombay2 <- setNames(bombay, c("times", "count"))
(bb1 <- fitsir(bombay2, method="BFGS"))

##
## Call:
## mle2(minuslogl = objfun, start = start, method = method, data = dataarg,
##     vecpar = TRUE, gr = gradfun, control = control)
##
## Coefficients:
##   log.beta  log.gamma      log.N    logit.i
##   3.113400   3.096465  15.622618 -14.078296
##
## Log-likelihood: -167.75

(bb3 <- fitsir(bombay2, dist="nbinom"))

##
## Call:
## mle2(minuslogl = objfun, start = start, method = method, data = dataarg,
##     vecpar = TRUE, gr = gradfun, control = control)
##
## Coefficients:
##   log.beta  log.gamma      log.N    logit.i
##   4.519040   4.514817  18.398145 -16.872443
##
## Log-likelihood: -141.97
```

Also, negative binomial fits way better but gets stuck in a weird place again and gives us bunch of warnings... I definitely need to reconsider 5 parameter fitting...

References

Dietz, K. (2009, April). Epidemics: the fitting of the first dynamic models to data. *Journal of Contemporary Mathematical Analysis 44*(2), 97–104.

Kermack, W. O. and A. G. McKendrick (1927). A contribution to the mathematical theory of epidemics. In *Proceedings of the Royal Society of London A: mathematical, physical and engineering sciences*, Volume 115, pp. 700–721. The Royal Society.