Using outbreak data to estimate the relative case fatality ratio

Nicholas G. Reich

December 6, 2019

1 Introduction to the data

The goal of this vignette is to provide a tutorial for implementing the case fatality ratio estimation methods available in the coarseDataTools R package. To illustrate the methods, we will estimate a relative case fatality ratio from a simulated outbreak dataset.

2 Loading the data and the code

We begin by loading the package and a set of simulated data.

```
> library(coarseDataTools)
> data(simulated.outbreak.deaths)
```

The data that we have loaded is a simulated dataset with five columns. Let's take a peek at the dataset.

```
> simulated.outbreak.deaths[15:20,]
```

```
   time grp    R D    N
15   15   1   53 0   53
16   16   1  121 0  121
17   17   1  293 0  293
18   18   1  731 0  731
19   19   1 1613 0 1613
20   20   1 3400 0 3400
```

The rows index observations at a particular time and group. The columns are defined as follows:

- time = the specified unit of time, t, for this observation
- grp = the covariate group, j, for this observation
- R = the number of recovered cases observed
- D = the number of dead cases observed
- N = the total number of cases observed, or D + R

We must perform a little bit of pre-processing on the data so that it is ready for the analysis. Specifically, we need to define the T time periods that we want to use for analysis. To avoid (for the moment) some convergence issues associated with small sample sizes, we will only include times where both groups have at least 10 total cases observed.

```
> ## set minimum number of observed cases for inclusion
> min.cases <- 10
> ## observed cases
> N.1 <- simulated.outbreak.deaths[1:60,"N"]
> N.2 <- simulated.outbreak.deaths[61:120,"N"]
> ## subset to run analyis on times with greater than min.cases
> first.t <-  min(which(N.1 > min.cases & N.2 > min.cases))
> last.t <-  max(which(N.1 > min.cases & N.2 > min.cases))
> idx.for.Estep <- first.t:last.t
> ## find and label the subset of times to be used for estimation routine
> new.times <- 1:length(idx.for.Estep)
> simulated.outbreak.deaths <- cbind(simulated.outbreak.deaths, new.times=NA)
> simulated.outbreak.deaths[c(idx.for.Estep, idx.for.Estep+60),"new.times"] <- rep(new.times, 2)
```

If we look at the new data matrix, we can see the new variable that we've made and that will be used in the analysis:

```
> simulated.outbreak.deaths[15:20,]

   time grp    R D    N new.times
15   15   1   53 0   53        NA
16   16   1  121 0  121        NA
17   17   1  293 0  293         1
18   18   1  731 0  731         2
19   19   1 1613 0 1613         3
20   20   1 3400 0 3400         4
```

3 Running an analysis

We wish to fit a model to the simulated dataset that adjusts for changing reporting rates over time and for a lag between disease onset and death. We assume that the reporting rates for dead and recovered cases vary by time but not by covariate j. We also assume that the case fatality ratio does not vary by time but is different for each of the j covariate groups. The model formula that adjusts for reporting rates but not the lag is

$$\log E(D_{tj}) \quad = \quad \log N_{tj} + \beta_0 + \alpha_t \cdot X_t + \gamma_j \cdot Y_j \tag{1}$$

where X_t and Y_j are indicator variables for $t = 2, \ldots, T$ and $j = 2, \ldots, J$. Then, γ_j is the log relative case fatality ratio. The EM algorithm described in the main paper imputes the values of D_{tj} for the E-step and uses the above equation as the M-step.

Before running an analysis, we need to fix a few parameters. Specifically, we need to fix the assumed η, or the vector of probabilities that define the survival distribution. We will assume that it is (0, .3, .4, .3). For example, given that a case will end up dying, there is a 40% chance that the death will occur on the third day after disease onset. Also, we need to set starting values for the α parameters in the model. We will assume that $\alpha_t = 0$ for all t.

```
> assumed.nu = c(0, .3, .4, .3)
> alpha.start <- rep(0, 22)
```

That is all the set up required. We can now call the central function, `EMforCFR`(), which generates three estimates of the CFR: the naïve, the reporting-rate-adjusted and the lag-adjusted estimators.

```
> cfr.ests <- EMforCFR(assumed.nu=assumed.nu,
+ alpha.start.values=alpha.start, full.data=simulated.outbreak.deaths, verb=FALSE,
+ SEM.var=TRUE, max.iter=500, tol=1e-5)
```

[Note that running cfr.ests will generate warnings (many of them, likely) but not to worry because this is due to the fact that the likelihood uses non-integer outcomes in calculating the Poisson density function. This is an expected behavior and is not an indication of a problem with the routine.]

The function `EMforCFR`() returns a list with the following components:

- naive.rel.cfr = the naïve estimator for the relative CFR
- glm.rel.cfr = the reporting-rate-adjusted estimator for the relative CFR
- EM.rel.cfr = the lag-adjusted estimator for the relative CFR
- EM.rel.cfr.var = the variance for the log-scale lag-adjusted estimator taken from the final M-step
- EM.rel.cfr.var.SEM = the Supplemented EM algorithm variance for the log-scale lag-adjusted estimator
- EM.rel.cfr.chain = a vector of the EM algorithm iterates of the lag-adjusted relative CFR estimates
- ests = the coefficient estimates for the model
- ests.chain.EM = a matrix with all of the coefficient estimates, at each EM iteration
- DM = the DM matrix from the SEM algorithm
- DMiter = a vector showing how many iterations it took for the variance component to converge in the SEM algorithm

We are particularly interested in the following components:

```
> cfr.ests$naive.rel.cfr

[1] 1.11048

> cfr.ests$glm.rel.cfr

factor(dat[, "grp"])2
           0.06035187

> cfr.ests$EM.rel.cfr

[1] 0.3370342

> cfr.ests$EM.rel.cfr.var.SEM

[1] 0.008049604
```