2019-10-15-fitting-seasonality-IL-flu

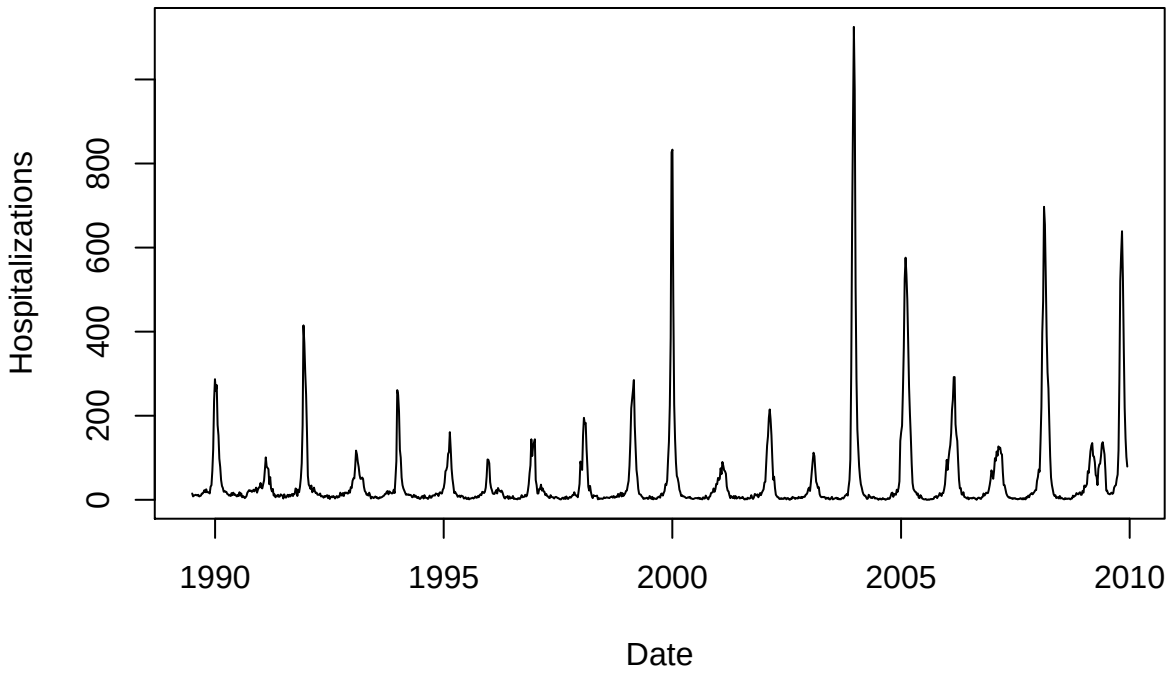
K. Bodie Weedop

10/15/2019

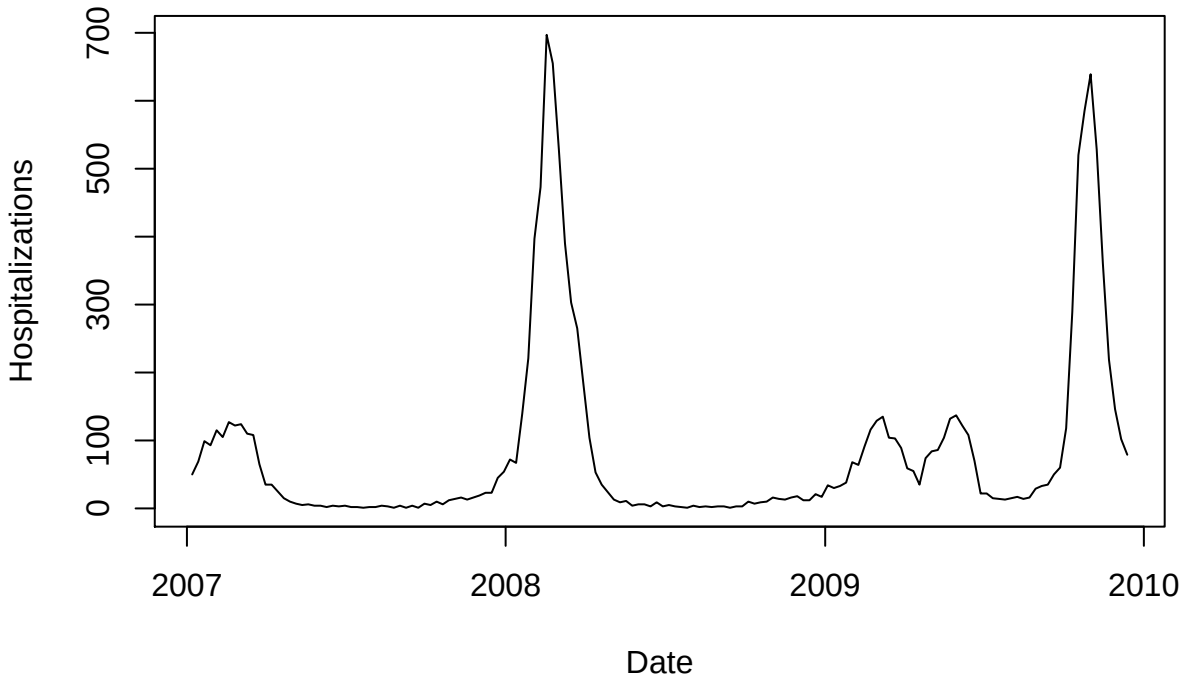
```
library(deSolve)
library(bbmle)
```

```
## Loading required package: stats4
```

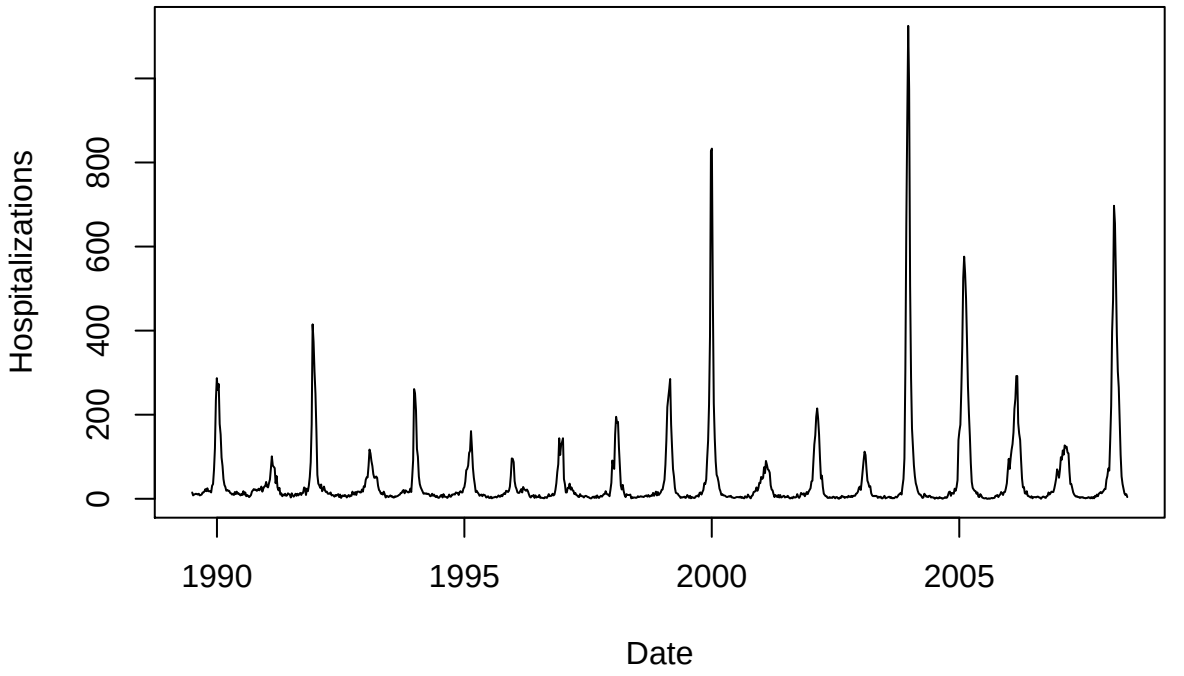
```
# Load data
flu.data <- read.csv("~/Documents/rotation-project/data/IL_hosp_cases_formatted.csv",
                     header = TRUE,
                     stringsAsFactors = FALSE)
# Using data for all ages only...
flu.all.ages <- flu.data[which(flu.data$age.class == 3),]
# Cumulative sum of flu hospitalizations over time series...
flu.all.ages$cumulative.cases <- cumsum(flu.all.ages$flu)
# Reformat date in data
flu.all.ages$date <- as.Date(flu.all.ages$date)
# Putting the week in the data frame if needed
flu.all.ages$week <- seq(1, nrow(flu.all.ages))
plot(flu.all.ages$flu ~ flu.all.ages$date,
     type = "l",
     xlab="Date",
     ylab="Hospitalizations")
```




```
# 2009 was a weird outbreak year that could be throwing off my fit. Take a closer, comparative look at
plot(flu.all.ages$flu[which(flu.all.ages$date > as.Date("2007-01-01"))] ~ flu.all.ages$date[which(flu.a
     type = "l",
     xlab="Date",
     ylab="Hospitalizations")
```




```
# For the time being I am removing data after 2008-06-01 to try and make things a bit easier on optimiz
flu.all.ages <- flu.all.ages[which(flu.all.ages$date < as.Date("2008-06-01")),]
plot(flu.all.ages$flu ~ flu.all.ages$date,
     type = "l",
     xlab="Date",
     ylab="Hospitalizations")
```




```
# beta(t)
betafx <- function (beta0, beta1, phi, t) {
  omega <- (2*pi)/52
```

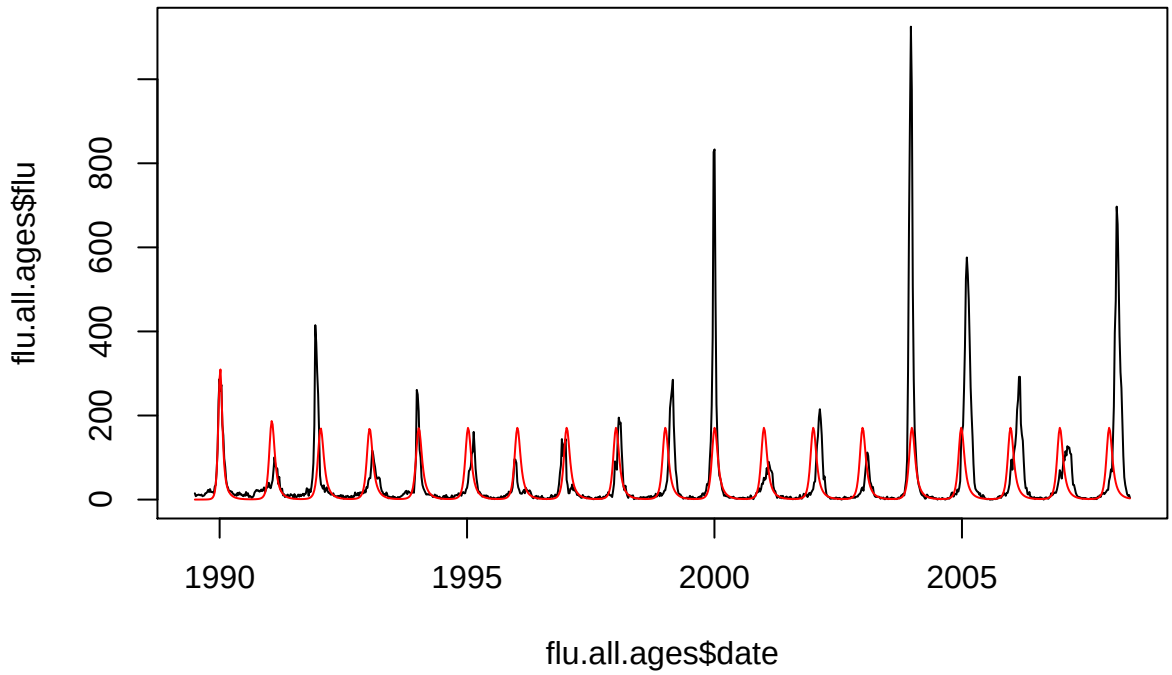
```r
  return(beta0 * (1 + beta1 * sin((omega*t) - phi)))
}

# Seasonally forced SIR with demographics
seasonal.sir <- function(t, vars, params) {
  N <- flu.all.ages$popsize[1]
  s0 <- vars[1]
  i0 <- vars[2]
  r0 <- vars[3]
  c0 <- vars[4]

  b_t <- betafx(beta0 = params[1],
                beta1 = params[2],
                phi = params[5],
                t)
  gamma <- params[3]
  xi <- params[4]
  mu <- 1/(70*52)

  dS <- mu * N - (b_t * s0 * i0) / N - mu * s0
  dI <- (b_t * s0 * i0) / N - gamma * i0 - mu * i0
  dR <- gamma * i0 - mu * r0
  dC <- ((b_t * s0 * i0) / N) * xi
  list(c(dS, dI, dR, dC))
}
# Simple function (no fitting or anything) which takes parameter values, simulates the model and plots
model.compare <- function(beta0, beta1, gamma, xi, phi) {
  # Population size; I realize this is only one value but for a rough go at fitting I am sacrificing a
  n.test <- flu.all.ages$popsize[1]
  # Calculate initial conditions
  s.test <- ((1/(70*52)+gamma)/betafx(beta0=beta0, beta1=beta1, phi=phi, t=1)) * n.test
  i.test <- 0.000001 * n.test
  r.test <- n.test - s.test - i.test

  out <- ode(y=c(S=s.test, I=i.test, R=r.test, C=0),
             times = seq(0, 987),
             seasonal.sir,
             parms = c(beta0, beta1, gamma, xi, phi))
  diff.cases <- diff(out[,5])
  # plot
  plot(flu.all.ages$flu ~ flu.all.ages$date, type = "l")
  lines(diff.cases ~ flu.all.ages$date, col = "red")
}
# Random parameters that I found by trial and error that seem to work well...
model.compare(65, 0.25, 0.45, 0.01, pi/1.5)
```




```
mle.sir <- function(b0, b1, x = 0.01) {
    t <- seq(0, 987)
    beta0 <- exp(b0)
    beta1 <- exp(b1)
    gamma <- 0.45 #exp(g)
    xi <- x
    phi <- pi/1.5

    N <- flu.all.ages$popsize[1]
    s0 <- ((1/(70*52) + gamma)/betafx(beta0=beta0, beta1=beta1, phi=phi, t=1)) * N
    i0 <- 0.00001 * N
    r0 <- N - s0 - i0

    results <- as.data.frame(ode(y=c(S=s0, I=i0, R=r0, C=0),
                                 times=t,
                                 seasonal.sir,
                                 parms=c(beta0, beta1, gamma, xi, phi),
                                 hmax=1/120))
    diff.cases <- diff(results$C)
    nll <- -sum(dpois(x=flu.all.ages$flu, lambda=diff.cases, log=TRUE))
    return(nll)
}

# Initialize parameter values
beta0 <- log(65)
beta1 <- log(0.25)

init.params <- list(b0=beta0,
                    b1=beta1)

fit0 <- mle2(mle.sir, start=init.params, optimizer = "nlm")
```

```
## Warning in nlm(f = objectivefunction, p = start, hessian = FALSE, ...): NA/
```

```
## Inf replaced by maximum positive value

## Warning in nlm(f = objectivefunction, p = start, hessian = FALSE, ...): NA/
## Inf replaced by maximum positive value

## Warning in nlm(f = objectivefunction, p = start, hessian = FALSE, ...): NA/
## Inf replaced by maximum positive value

## Warning in nlm(f = objectivefunction, p = start, hessian = FALSE, ...): NA/
## Inf replaced by maximum positive value
```

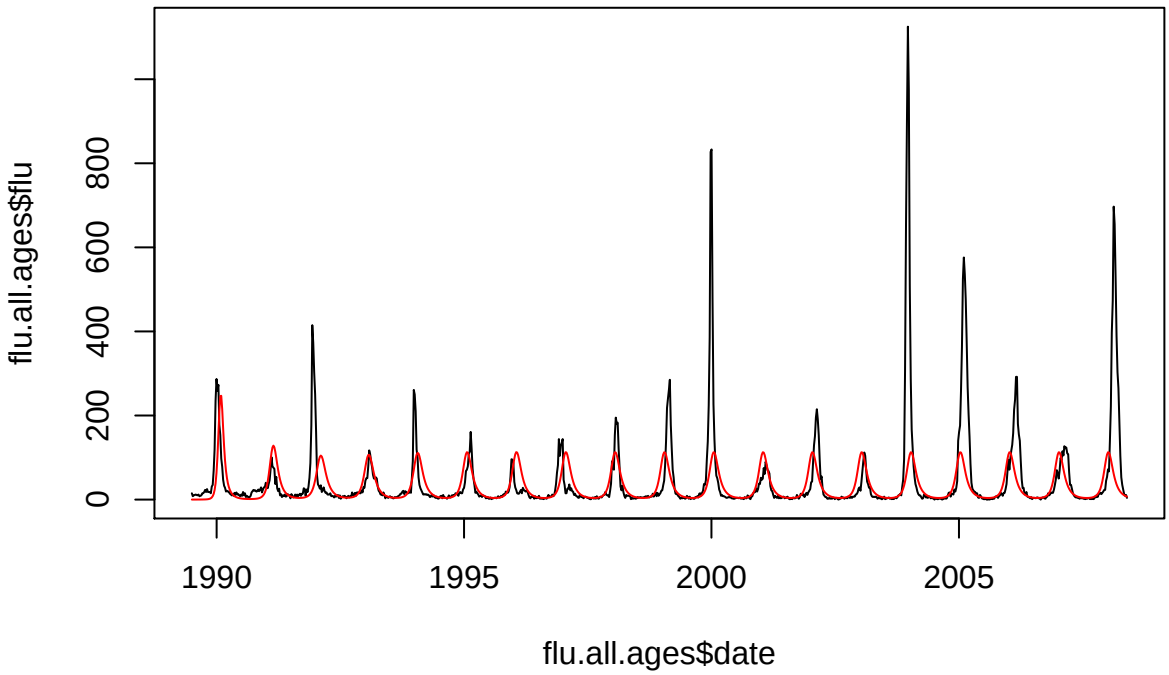
```
fit0
```

```
##
## Call:
## mle2(minuslogl = mle.sir, start = init.params, optimizer = "nlm")
##
## Coefficients:
##        b0        b1
##  3.998135 -1.925447
##
## Log-likelihood: -31771.58
##
## Warning: optimization did not converge (code 1: )
```

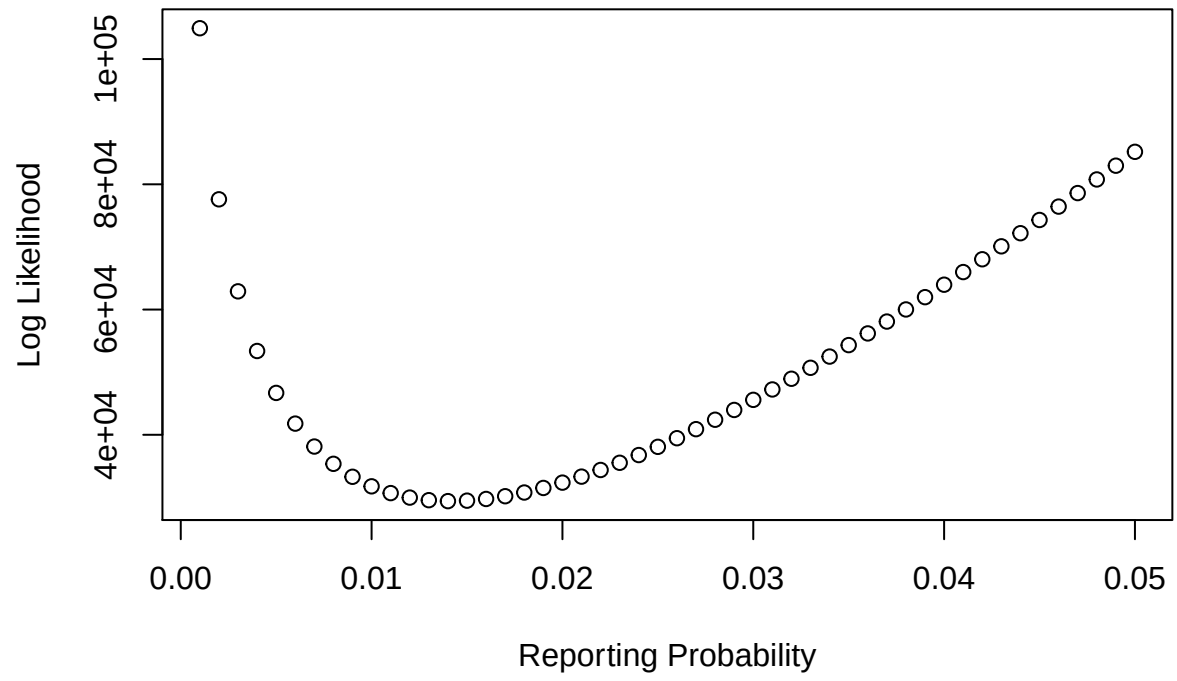
```
fit <- mle2(mle.sir, start=as.list(coef(fit0)), optimizer = "nlm")
fit
```

```
##
## Call:
## mle2(minuslogl = mle.sir, start = as.list(coef(fit0)), optimizer = "nlm")
##
## Coefficients:
##        b0        b1
##  3.998135 -1.925447
##
## Log-likelihood: -31771.58
##
## Warning: optimization did not converge (code 1: )
```

```
model.compare(exp(coef(fit0)[1]), exp(coef(fit0)[2]), 0.45, 0.01, pi/1.5)
```




```
compare.reporting <- data.frame(xi=seq(0.001, 0.05, length.out=50),
                                mle=NA)
for (i in 1:nrow(compare.reporting)) {
  compare.reporting$mle[i] <- mle.sir(log(exp(coef(fit)[1])), log(exp(coef(fit)[2])), compare.reporting$
}
plot(mle ~ xi,
     data=compare.reporting,
     xlab="Reporting Probability",
     ylab="Log Likelihood")
```



```
model.compare(exp(coef(fit0)[1]),
              exp(coef(fit0)[2]),
              0.45,
              compare.reporting$xi[which(compare.reporting $mle == min(compare.reporting$mle))],
              pi/1.5)
```