Spring Temperature Triggers Analysis

Temperature Trigger Analysis

First, we load the data.

```
knitr::opts_chunk$set(fig.width=6, fig.height=4, fig.path='Figs/', warning=FALSE,
                      message=FALSE)
library(rEDM)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

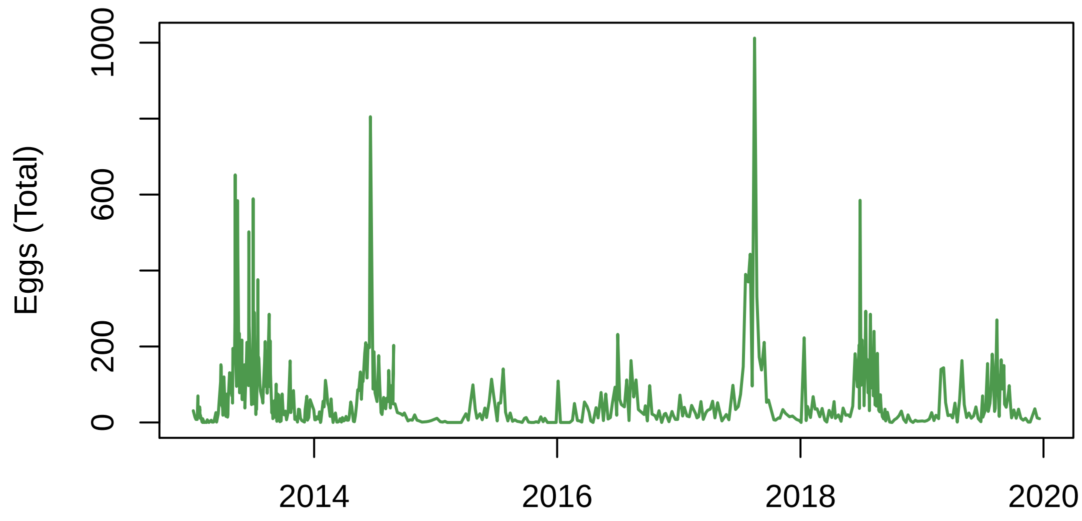
```
library(psych)
#Temperature Data
temps = read.csv("temp_data_01_2013-06_2020.csv")
temp_vals = temps$temperature
temp_dates = as_datetime(as.character(temps$time))
temp_vals=temp_vals[!is.na(temp_dates)]
temp_dates = temp_dates[!is.na(temp_dates)]

#Egg Data
d = read.csv("fish_eggs.csv")
eggs = as.numeric(as.character(d$Eggs))

days = as.character(d$Date)
eggs = cbind(days,eggs)
eggs = eggs[!is.na(eggs[,2]),]
eggs = eggs[-572,]

egg_dates =as.Date(eggs[,1], format = "%m/%d/%Y")
```

Next, we define a function that takes our very high resolution temperature timeseries (5 min sampling frequency) and makes it daily averages.

```
make_day <- function(){

  temperatures = array(NA, dim= c(365*10,1))
  times = array(NA, dim= c(365*10,1))
  count = 1
  for(d in as.character(unique(as_date(temp_dates)))){
    if(year(as_date(d)) >= 2012){
    temps_on_this_day = as.numeric(as.character(temp_vals[as_date(temp_dates) == d ]))


    temperatures[count] = mean(temps_on_this_day[!is.na(temps_on_this_day)])
    times[count] = as.character(d)
    count = count +1
    }
  }
  return(cbind(times[!is.na(temperatures)],temperatures[!is.na(temperatures)]))

}

day <- make_day()
```

We can check out what the raw data looks like plotting them below:

```
  plot(egg_dates[year(egg_dates)>=2013], eggs[,2][year(egg_dates)>=2013],
       type = 'l', xlab = "",
       ylab = "Eggs (Total)",
       lwd = 1.5, col = rgb(.3,.6,.3))
```



```
  plot(as.Date(day[,1])[year(as.Date(day[,1]))>=2013],
       day[,2][year(as.Date(day[,1]))>=2013], type = 'l',
       lwd = 1.5, ylab = "Temperature", xlab = "", col = rgb(.3,.3,.3))
```

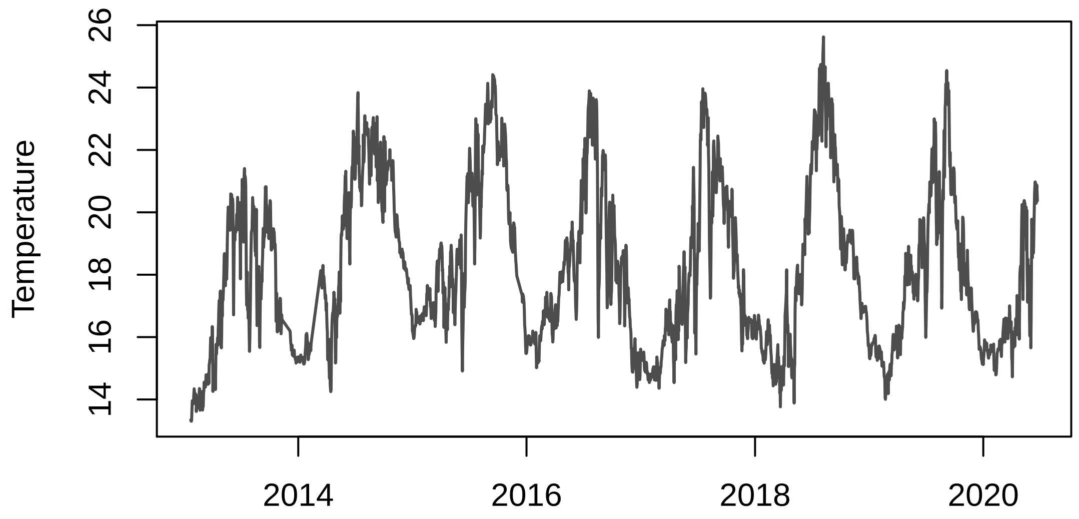
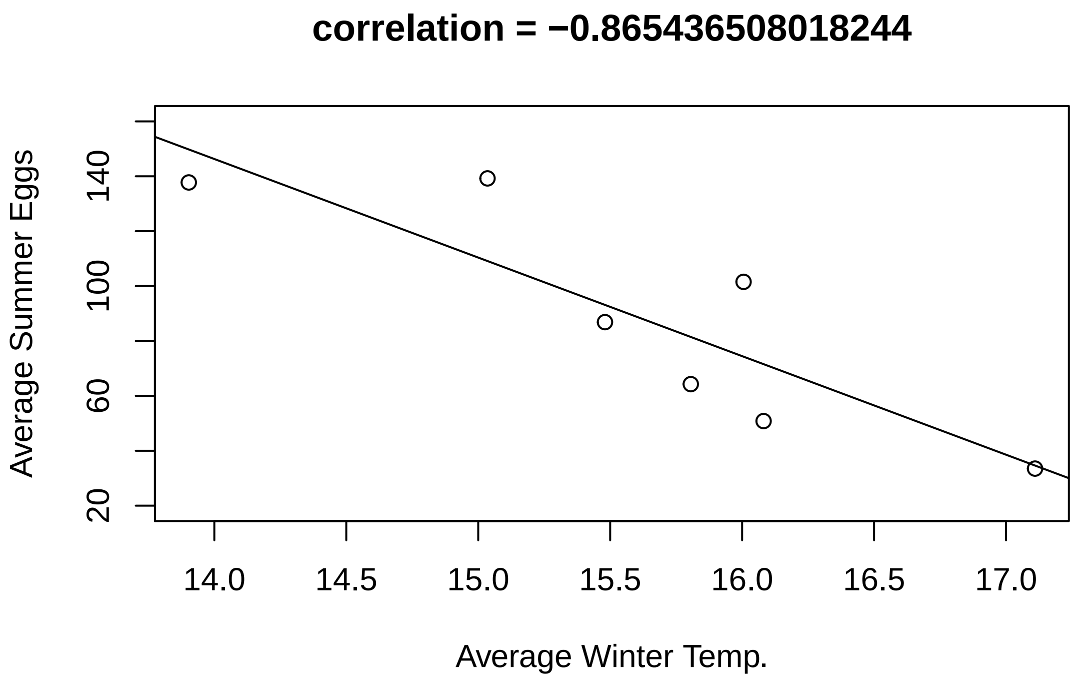


Figure 1C and 1D Correlations

In accordance with Duke et al. average winter temperature was defined as the average temperature from December through Februrary, and average Summer Eggs was defined as the average eggs abundance from March through August.

Here, we calculate these averages and plot them against each other.

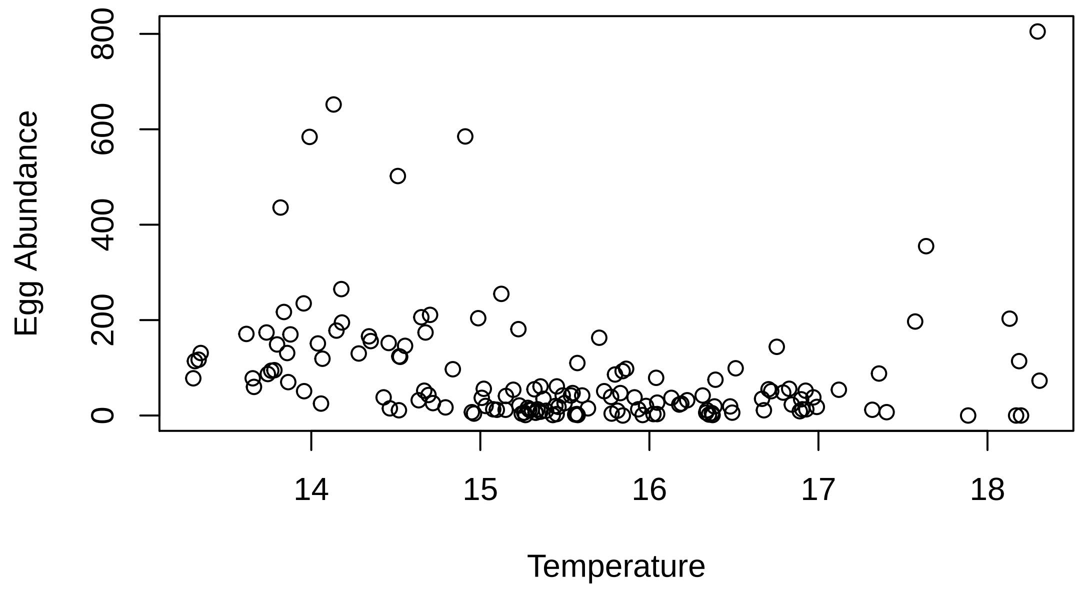
```
winter_temps = day[which( month(as.Date(day[,1])) >= 12 | month(as.Date(day[,1])) <=2 ), ]
average_temps = {}
for(y in c(2013:2020)){
  i = which(as.Date(winter_temps[,1]) == as.Date(paste(y,"-1-25", sep = "")))
  differences = as.Date(winter_temps[i,1])-as.Date(winter_temps[,1])
  indexes = which(abs(differences) < 180)
  average = mean(as.numeric(winter_temps[indexes,2]))
  average_temps = rbind(average_temps, c(y, average))
}

averages = {}
summer_eggs = eggs[which(month(as.Date(eggs[,1], format = "%m/%d/%Y")) >= 3
                         & month(as.Date(eggs[,1], format ="%m/%d/%Y")) <= 8), ]

for(y in c(2013:2019)){
  i = which.min(abs(as.Date(summer_eggs[,1],
                            format = "%m/%d/%Y") - as.Date(paste(y,"-6-05", sep = ""))))
  differences = as.Date(summer_eggs[i,1],
                        format = "%m/%d/%Y")-as.Date(summer_eggs[,1],
                                                       format = "%m/%d/%Y")
  indexes = which(abs(differences) < 140)
  average = mean(as.numeric(summer_eggs[indexes,2]))
  averages = rbind(averages, c(y+1, average))
}

{
plot(average_temps[1:7,2], averages[,2], ylim = c(20,160),
     xlab = "Average Winter Temp.",
     ylab = "Average Summer Eggs",
     main = paste("correlation =", cor(average_temps[1:7,2], averages[,2])))
abline(lm(averages[,2]~average_temps[1:7,2]))
}
```



Although a strong correlation exists between the average winter temperature and average summer egg abundance, look at the unaveraged data show little-to-no correlation.

Here, we take an analagous lag to that of Duke et al., and use temperature values from Dec 1st of each year and plot it against the corresponding egg value 90 days later, if it was sampled. This is done on a moving window through May (Dec 1st - May 31st). Note that this window does extend further than in the average above, because in order to maintain the 90 day lag between temperature and egg abundance, we must look further in the season (up to May) to capture the full summer season of eggs (to August).

```r
winter_temps = day[which( month(as.Date(day[,1])) >= 12 | month(as.Date(day[,1])) <=5 ), ]
correlation  = {}
for(y in c(2012:2019)){
  for(d in c(0:120)){
    i = which(as.Date(winter_temps[,1]) == as.Date(paste(y,"-12-01",
                                                         sep = ""))+d )

    i2 = which(as.Date(eggs[,1],
                       format = "%m/%d/%Y") ==
                 as.Date(paste(y,"-12-01",

                               sep = ""))+d+90 )

    if(length(i2) >0 & length(eggs[,2][i2]) >0
       & length(winter_temps[,2][i]) >0){
      correlation = rbind(correlation,
                          c(as.character(
                            as.Date(paste(y,"-12-01", sep = ""))+d),
                            as.character(as.Date(paste(y,"-12-01",
                                                       sep = ""))+d+90),
                                          winter_temps[,2][i], eggs[,2][i2]))
    }
  }
}

plot(as.numeric(correlation[,3]), as.numeric(correlation[,4]),
     xlab = "Temperature",
     ylab = "Egg Abundance")
```



Exploring this over a range of lags shows no correlation

```
winter_temps = day[which( month(as.Date(day[,1])) >= 12 | month(as.Date(day[,1])) <=5 ), ]
cs = {}
for(lag in c(121:180)){
  print(lag)
  correlation  = {}
for(y in c(2012:2019)){
  for(d in c(0:120)){
    i = which(as.Date(winter_temps[,1]) == as.Date(paste(y,"-12-01",
                                                        sep = ""))+d )
    if(length(i) > 1){i=i[1]}

    i2 = which(as.Date(eggs[,1],
                       format = "%m/%d/%Y") ==
                 as.Date(paste(y,"-12-01",

                               sep = ""))+d+lag )
    if(length(i2) > 1){i2=i2[1]}

    if(length(i2) >0 & length(eggs[,2][i2]) >0
       & length(winter_temps[,2][i]) >0){
      correlation = rbind(correlation,
                          c(as.character(
                            as.Date(paste(y,"-12-01", sep = ""))+d),
                            as.character(as.Date(paste(y,"-12-01",
                                                       sep = ""))+d+lag),
                                          winter_temps[,2][i], eggs[,2][i2]))
    }
  }
```

```r
}
  c = cor(as.numeric(correlation[,3]), as.numeric(correlation[,4]))
  cs = c(cs,c)
}
```

```
## [1] 121
## [1] 122
## [1] 123
## [1] 124
## [1] 125
## [1] 126
## [1] 127
## [1] 128
## [1] 129
## [1] 130
## [1] 131
## [1] 132
## [1] 133
## [1] 134
## [1] 135
## [1] 136
## [1] 137
## [1] 138
## [1] 139
## [1] 140
## [1] 141
## [1] 142
## [1] 143
## [1] 144
## [1] 145
## [1] 146
## [1] 147
## [1] 148
## [1] 149
## [1] 150
## [1] 151
## [1] 152
## [1] 153
## [1] 154
## [1] 155
## [1] 156
## [1] 157
## [1] 158
## [1] 159
## [1] 160
## [1] 161
## [1] 162
## [1] 163
## [1] 164
## [1] 165
## [1] 166
## [1] 167
## [1] 168
## [1] 169
## [1] 170
## [1] 171
## [1] 172
## [1] 173
## [1] 174
## [1] 175
## [1] 176
## [1] 177
## [1] 178
## [1] 179
## [1] 180
```

EDM Analysis (Figure 1E and S1)

Although temperature data existed for every day, eggs were collected at inconsistent intervals, typically ranging from one collection every 3-8 days. Thus, in order to make a proper embedding, we filtered both temperature and egg abundance time series to only include temperature values for which a collection occurred on a given day, 6-8 days prior, and 13-15 days prior as well. This gave us a 3-dimensional embedding for fish eggs, with time lags of about 1 week, with accompanying temperature values.

We define a function for taking lags of the egg abundance time series, with a default lag size (tau) is 1 week. Because eggs were not sampled at a consisten interval, a 1 week lag is taken to be 6-8 days.

```r
make_egg_block<- function(E){

  d = read.csv("fish_eggs.csv")
  eggs = as.numeric(as.character(d$Eggs))

  days = as.character(d$Date)
  eggs = cbind(days,eggs)
  eggs = eggs[!is.na(eggs[,2]),]
  eggs = eggs[-572,]
  egg_dates =as.Date(eggs[,1], format = "%m/%d/%Y")

  eggs = cbind(as.character(as.Date(egg_dates[year(egg_dates)>=2013])),
               eggs[,2][year(egg_dates)>=2013] )
  eggs = cbind(eggs, array(NA, dim = c(NROW(eggs),E-1)))

  for(i in c(10:NROW(eggs))){
    curr = eggs[i,]
    date = as.Date(curr[1])
    diffs = difftime(as.Date(eggs[,1]), date, units = "secs")

    point = {}
    for(l in c(1:(E-1))){
      r1 = 7*l*60*60*24
      indexes = which(abs(diffs+r1)  < 60*60*24  )
      if(length(indexes>0)){
        index = which.min(abs(diffs+r1))
        point = c(point, eggs[index, 2])
      }
    }
    if(length(point) == E-1){
      eggs[i, 3:4] <- point
    }
  }
  eggs = eggs[-which(is.na(eggs[,3])),]
  return(eggs)
}
```

First, we use the make_egg_block() function to embed our egg abundance time series in 3-dimensions with a lag of 7 days. Then, we append a 5th column that corresponds to the temperature value corresponding to the same date of the unlagged egg value.

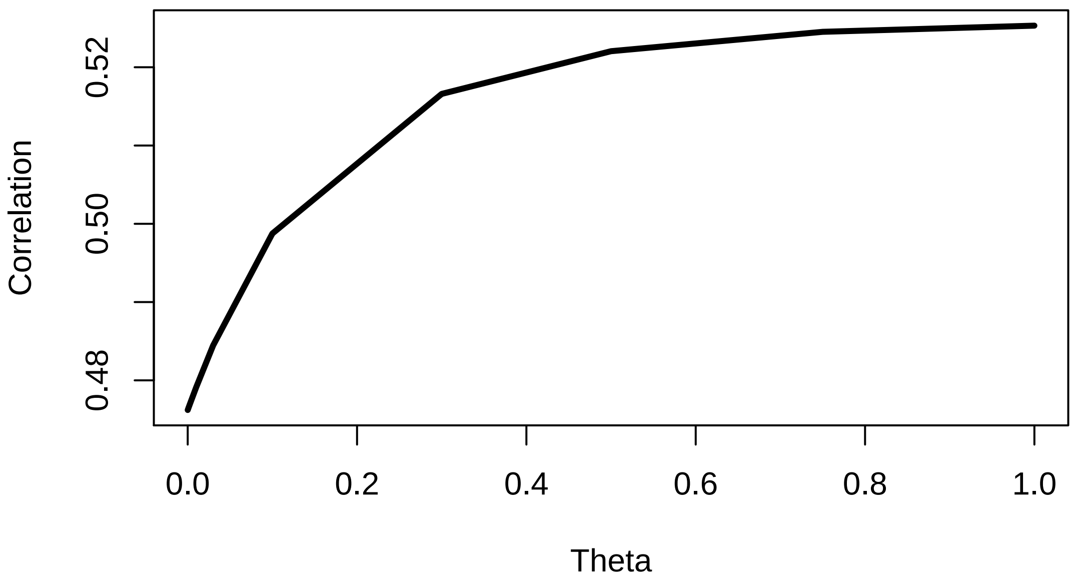
```
eggs_block<-make_egg_block(3)
eggs_block = cbind(eggs_block, array(NA, dim = c(NROW(eggs_block),1)))
for(i in c(1:NROW(eggs_block))){
  curr = eggs_block[i,]
  date = as.Date(curr[1])
  index = which(as.Date(day[,1]) == date)
  if(length(index)>0){
    eggs_block[i,NCOL(eggs_block)]<- day[index,2]
  }
}
eggs_block = eggs_block[-which(is.na(eggs_block[,5])),]

for(i in c(2:NROW(eggs_block))){ #remove the days there was > than 1 sample in a day
  if(eggs_block[(i-1),1] == eggs_block[(i),1]){
    eggs_block[(i-1),1] = NA
  }
}
eggs_block = eggs_block[-which(is.na(eggs_block[,1])),]
colnames(eggs_block) <- c("Date","Eggs_t","Eggs_t-7","Eggs_t-14", "Temp_t" )
print(head(eggs_block))
```

```
##      Date         Eggs_t Eggs_t-7 Eggs_t-14 Temp_t
## [1,] "2013-01-22" "41"   "8"      "14"      "13.3459409836066"
## [2,] "2013-01-29" "2"    "34"     "8"       "13.8637926388889"
## [3,] "2013-01-31" "10"   "11"     "70"      "13.9576729166667"
## [4,] "2013-02-05" "0"    "3"      "34"      "14.0181543055556"
## [5,] "2013-02-07" "1"    "0"      "11"      "14.1777979166667"
## [6,] "2013-02-12" "1"    "0"      "3"       "13.7818711111111"
```

S-Map Test for non-linearity First, we test if the egg abundance time series is non-linear. We do this by performing S-Map forecasts on our embedding generated above to see if predictions increase with increasing theta (1).

Note: we need only inspect the behavior of the egg abundance time series in the analysis because we are only interested in the dynamics driving egg abundance; not in the drivers of sea surface temperature. Sea surface temperature may be a largely linear signal (strongly seasonally driven), however, the spawning response to this linear driver may still be non-linear.

```
thetas = c(0, 1e-04, 3e-04, 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 0.5,
  0.75, 1)
m = data.matrix(eggs_block[,2:4])
m=apply(m, 2, as.numeric)
out<- block_lnlp(m, method = "s-map", target_column = 1,
                 columns = c(2,3), tp = 0,theta = thetas)
plot(thetas, out$rho, type = 'l', ylab = "Correlation", xlab = "Theta", lwd = 3)
```

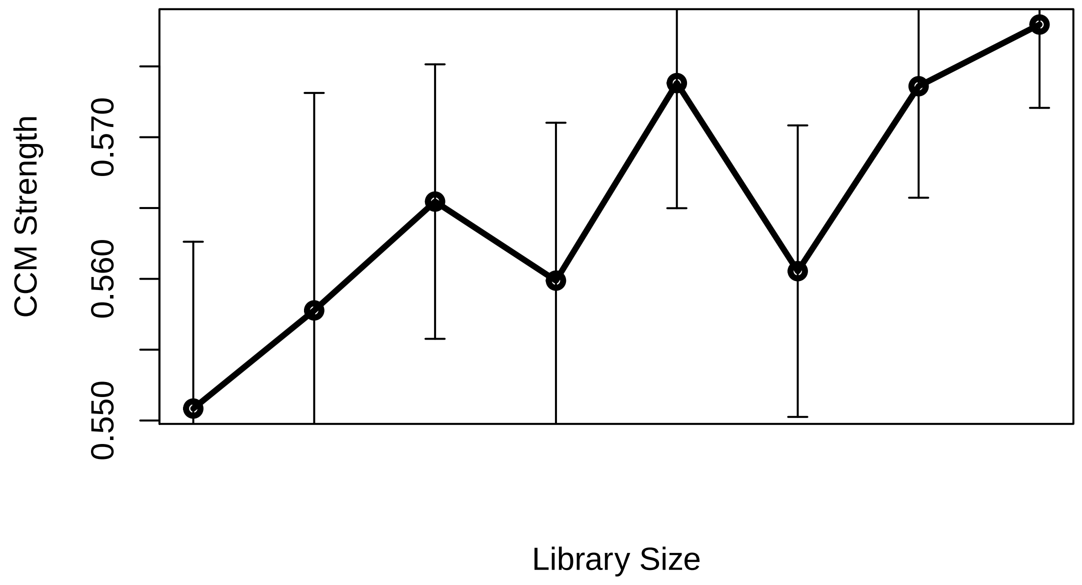


Indeed, our predictions do improve with increasing theta. Thus, egg abundance is likely driven by non-linear processes.

CCM Because both egg abundance and temperature are strongly seasonally driven, we needed to make sure we were not identifying shared information in the two variables driven by seasonality. To account for this, nearest neighbor selection only considered time points that were within 90 calendar days for our target prediction. Without doing this, increased library size will only increase the amount of seasonal information resolved in the embedding rather than actual causal inference.

Libraries of potential neighbors (points within 90 calendar days of the date of the target) were generated at random for each predicted point. Library sizes ranged from 5 - 40 points (increasing by increments of 5). Once the library was randomly generated, the nearest 4 neighbors (E+1, see Sugihara et al. 2012) in state space were selected and used to make a prediction. After a prediction was made on each temperature value, Pearson's correlation was calculated between observed and predicted values. This process was repeated 50 (adjusted by num_samples) times for each library size.

```
total = {}
num_samples = 25
for(libsize in seq(10,80,10)){
  rhos = {}
  for(trial in c(1:num_samples)){

    observations = {}
    predictions = {}
    for(i in c(1:NROW(eggs_block))){
      curr = eggs_block[i,]
      date = yday(as.Date(curr[1]))
      diffs = yday(as.Date(eggs_block[,1])) -  date

      indexes = which(abs(diffs)  < 90 )
      m <- eggs_block[indexes,]
      m <- m[,2:NCOL(m)]
      m<-apply(m,2, FUN = as.numeric)
      m = rbind(m, m[1,])
      len = NROW(m)
      if(len > 80){
        lib = sample(c(1:len), libsize)
        lib = cbind(lib,lib+1)


          out<- block_lnlp(m, columns = c(1:3), target_column = 4, stats_only = F,
                           lib =lib, pred = c(1, NROW(m)), tp = 0, num_neighbors = 4, silent = T)

          pred = out$model_output[[1]]
          obs = pred$obs
          pred = pred$pred

          index = which(obs == as.numeric(curr[5]))

          observations = c(observations, obs[index])
          predictions = c(predictions, pred[index])
      }
      }
    rhos= c(rhos, cor(observations, predictions))


  }
  total = cbind(total, rhos)
  }

plot(colMeans(total), lwd = 3, type = 'l', xaxt = "n", xlab = "Library Size",
     ylab = "CCM Strength")
error.bars(total, eyes = F, lwd = 3, add = T)
```



Because we have increasing predictability with increasing library size, there is likely non-linear coupling between the two variables at this scale. Next, we take a closer look at one of these non-linear relationships.

Spring Temperature Trigger Analysis

Here we define two functions: First is a function to find the largest temperature difference (i.e. STT) given a start date, end date, and window size (delay) for each year. The second is simply to find the maximum egg value for each year within a given time frame (Summer).

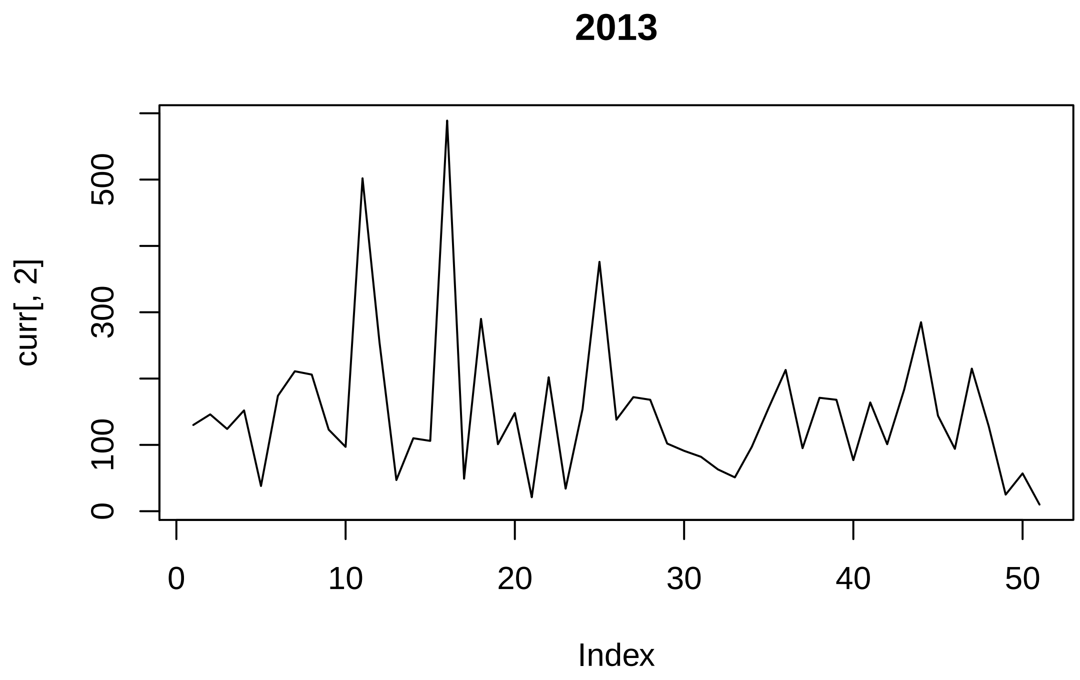
```r
find_biggest_diff<- function(ts, start = "01-01", end = "04-01",
                             delay = 24, positive = T){
  days = yday(ts[,1])
  if(is.character(start)){
    start = paste(start, "-2000", sep ="")
    end = paste(end, "-2000", sep ="")
    start = yday(as.Date(start, format = "%m-%d-%Y"))
    end = yday(as.Date(end, format = "%m-%d-%Y"))
  }
  data = ts[days>=start & days<=end,]
  years = year(ts[days>=start & days<=end,1])
  unique_years = c(2013:2020)

  differences = array(NA, dim = c(NROW(ts), 2))
  max_differences = array(NA, dim = c(length(unique_years), 2))
  count = 1
  yloop=1
  for(y in unique_years){
    current = data[years == y,]
    dates = current[,1]
    temps = as.numeric(current[,2])
    for(i in c(1:length(dates))){
      d = dates[i]
      if(yday(d)>delay+1){
        diffs= difftime(dates, d, units = "secs") + delay*60*60*24
        diff = temps[i]-temps[i:which.min(as.numeric(abs(diffs)))]
        differences[count,]<- c(d, max(diff))
        count = count+1
      }
    }

    i = which.max(as.numeric(differences[,2][!is.na(differences[,2])
                                             & year(differences[,1])==y]))

    if(positive == F){
      i = which.min(as.numeric(differences[,2][!is.na(differences[,2])
                                               & year(differences[,1])==y]))
    }
    v = differences[!is.na(differences[,2]) & year(differences[,1])==y,]
    v=v[i,]
    max_differences[yloop,]<- v

    yloop = yloop+1
  }
  return(max_differences)
}
```

```r
find_egg_max <- function(start, end){

  if(is.character(start)){
  start = paste(start, "-2000", sep ="")
  end = paste(end, "-2000", sep ="")
  start = yday(as.Date(start, format = "%m-%d-%Y"))
  end = yday(as.Date(end, format = "%m-%d-%Y"))
  }

  vals = eggs[yday(as.Date(eggs[,1], format = "%m/%d/%Y"))>start &
         yday(as.Date(eggs[,1], format = "%m/%d/%Y"))<end, ]
  vals = vals[!is.na(vals[,1]),]

  days = yday(as.Date(eggs[,1], format = "%m/%d/%Y"))


  years = year(as.Date(vals[,1], format = "%m/%d/%Y"))
  unique_years = unique(year(as.Date(vals[,1], format = "%m/%d/%Y")))
  out<- array(NA, dim = c(7,2))
  loop = 1
  for(y in c(2013:2019)){
    curr = vals[years == y,]
    print((curr))
    plot(curr[,2], main = y, type = 'l')
    # out[loop,]<- c(y, max(as.numeric(curr[,2])))
    i =which.max(as.numeric(curr[,2]))
    out[loop,]<- curr[i,]
    loop = loop+1
  }

  return(out)

}
```

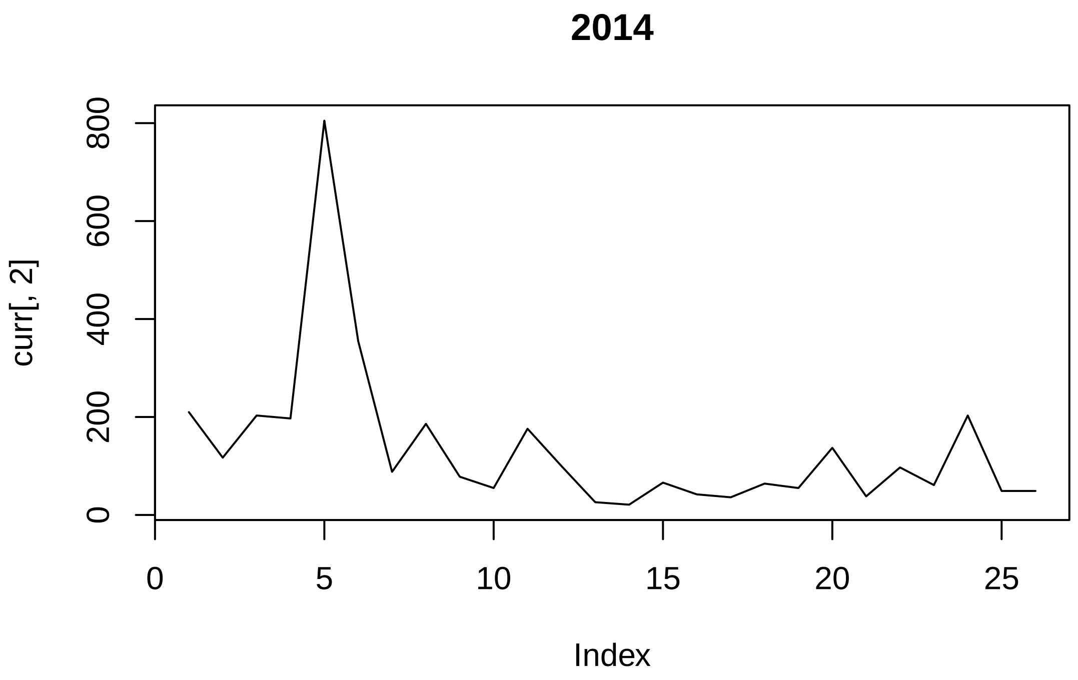
Now we use these functions to find the STT for each year, and peak eggs.

```
tdiffs = find_biggest_diff(day, start = "4-1", end = "6-17", delay = 27)
egg_max <- find_egg_max("6-1", "9-1")
```

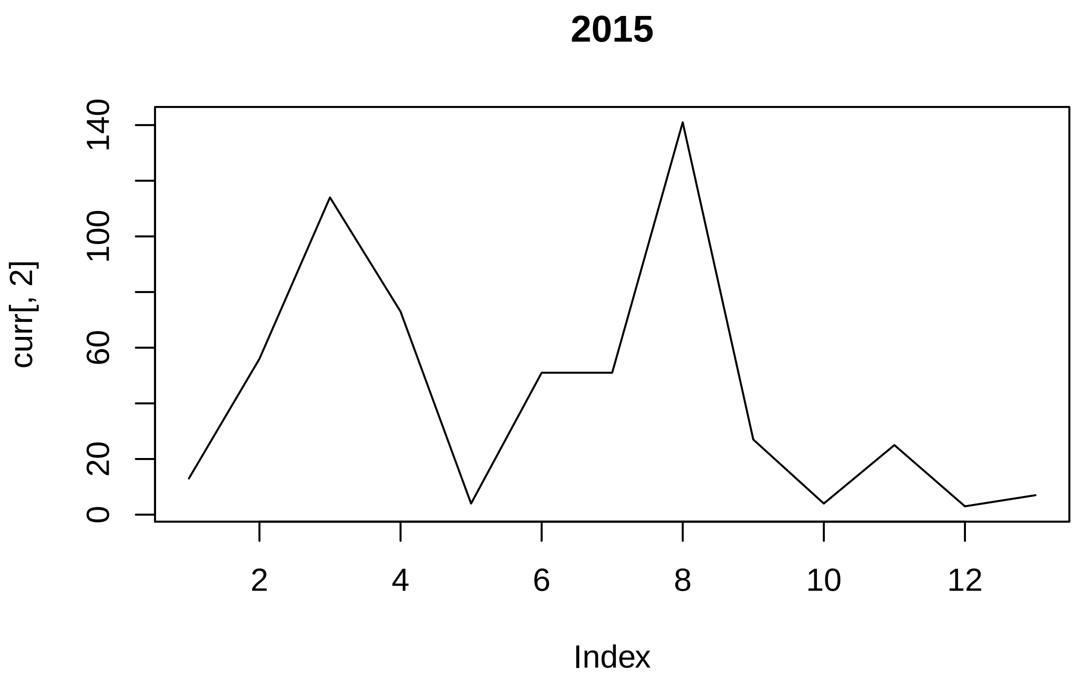
```
##       days        eggs
##  [1,] "6/3/2013"  "130"
##  [2,] "6/4/2013"  "146"
##  [3,] "6/5/2013"  "124"
##  [4,] "6/6/2013"  "152"
##  [5,] "6/7/2013"  "38"
##  [6,] "6/12/2013" "174"
##  [7,] "6/13/2013" "211"
##  [8,] "6/14/2013" "206"
##  [9,] "6/17/2013" "123"
## [10,] "6/18/2013" "97"
## [11,] "6/19/2013" "502"
## [12,] "6/20/2013" "255"
## [13,] "6/27/2013" "47"
## [14,] "6/28/2013" "110"
## [15,] "7/1/2013"  "106"
## [16,] "7/2/2013"  "589"
## [17,] "7/3/2013"  "49"
## [18,] "7/5/2013"  "290"
## [19,] "7/8/2013"  "101"
## [20,] "7/9/2013"  "148"
## [21,] "7/10/2013" "21"
## [22,] "7/11/2013" "202"
## [23,] "7/12/2013" "34"
## [24,] "7/15/2013" "154"
## [25,] "7/16/2013" "376"
## [26,] "7/17/2013" "138"
## [27,] "7/18/2013" "172"
## [28,] "7/19/2013" "168"
## [29,] "7/22/2013" "102"
## [30,] "7/23/2013" "91"
## [31,] "7/24/2013" "82"
## [32,] "7/29/2013" "63"
## [33,] "7/31/2013" "51"
## [34,] "8/2/2013"  "97"
## [35,] "8/5/2013"  "156"
## [36,] "8/7/2013"  "213"
## [37,] "8/8/2013"  "95"
## [38,] "8/9/2013"  "171"
## [39,] "8/12/2013" "168"
## [40,] "8/13/2013" "77"
## [41,] "8/14/2013" "164"
## [42,] "8/15/2013" "101"
## [43,] "8/16/2013" "183"
## [44,] "8/19/2013" "285"
## [45,] "8/20/2013" "144"
## [46,] "8/21/2013" "94"
## [47,] "8/22/2013" "215"
## [48,] "8/23/2013" "128"
## [49,] "8/27/2013" "25"
## [50,] "8/28/2013" "57"
## [51,] "8/30/2013" "10"
```



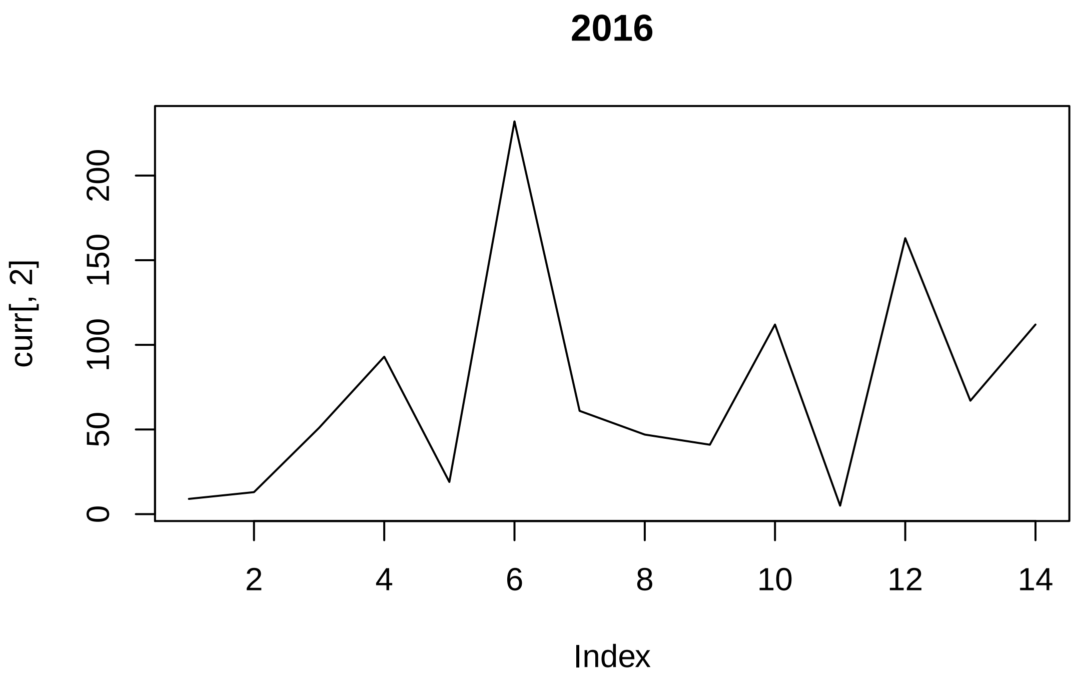
```
##        days         eggs
##  [1,] "6/5/2014"  "210"
##  [2,] "6/9/2014"  "117"
##  [3,] "6/12/2014" "203"
##  [4,] "6/16/2014" "197"
##  [5,] "6/19/2014" "805"
##  [6,] "6/24/2014" "355"
##  [7,] "6/27/2014" "88"
##  [8,] "6/30/2014" "186"
##  [9,] "7/3/2014"  "78"
## [10,] "7/9/2014"  "55"
## [11,] "7/14/2014" "176"
## [12,] "7/17/2014" "100"
## [13,] "7/21/2014" "26"
## [14,] "7/24/2014" "21"
## [15,] "7/29/2014" "66"
## [16,] "8/1/2014"  "42"
## [17,] "8/4/2014"  "36"
## [18,] "8/7/2014"  "64"
## [19,] "8/11/2014" "55"
## [20,] "8/13/2014" "137"
## [21,] "8/18/2014" "38"
## [22,] "8/21/2014" "97"
## [23,] "8/25/2014" "61"
## [24,] "8/28/2014" "203"
## [25,] "8/26/2014" "49"
## [26,] "9/1/2014"  "49"
```



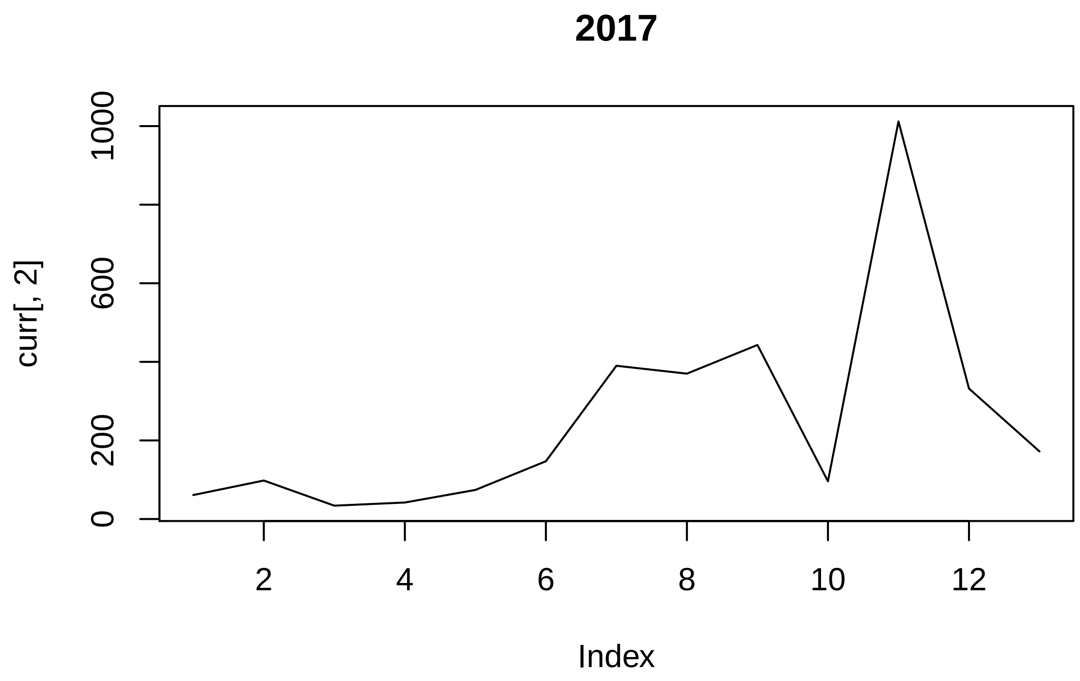
```
##       days        eggs
##  [1,] "6/3/2015"  "13"
##  [2,] "6/11/2015" "56"
##  [3,] "6/18/2015" "114"
##  [4,] "6/24/2015" "73"
##  [5,] "7/5/2015"  "4"
##  [6,] "7/9/2015"  "51"
##  [7,] "7/15/2015" "51"
##  [8,] "7/23/2015" "141"
##  [9,] "7/30/2015" "27"
## [10,] "8/6/2015"  "4"
## [11,] "8/13/2015" "25"
## [12,] "8/20/2015" "3"
## [13,] "8/27/2015" "7"
```



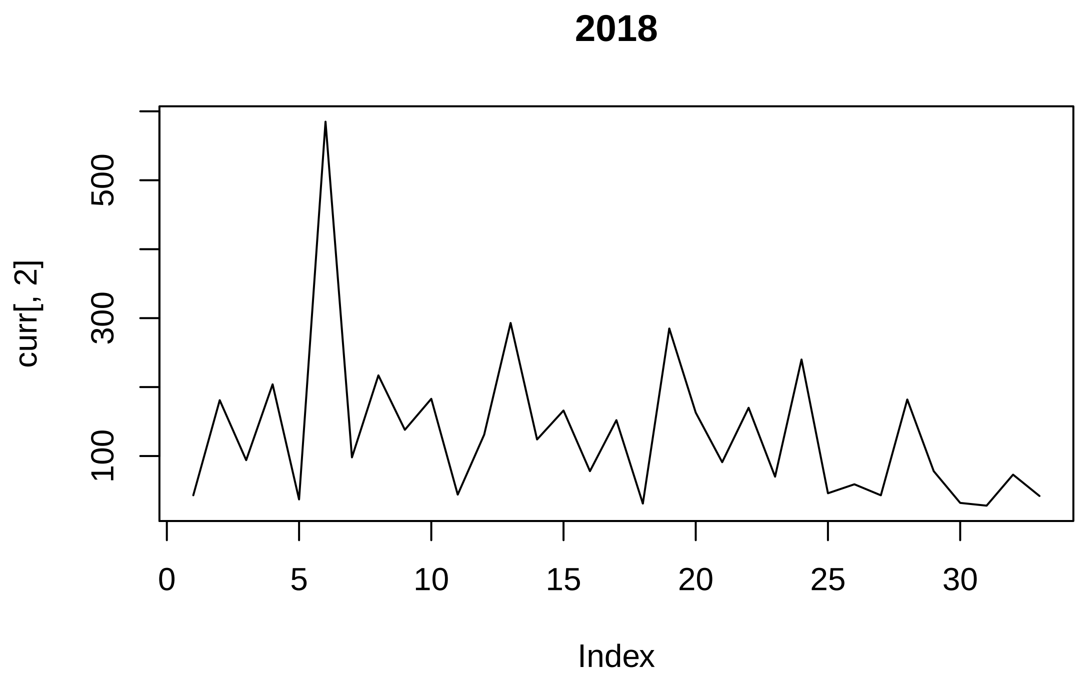
```
##       days        eggs
##  [1,] "6/2/2016"  "9"
##  [2,] "6/9/2016"  "13"
##  [3,] "6/15/2016" "51"
##  [4,] "6/23/2016" "93"
##  [5,] "6/28/2016" "19"
##  [6,] "7/1/2016"  "232"
##  [7,] "7/7/2016"  "61"
##  [8,] "7/12/2016" "47"
##  [9,] "7/21/2016" "41"
## [10,] "7/28/2016" "112"
## [11,] "8/4/2016"  "5"
## [12,] "8/10/2016" "163"
## [13,] "8/18/2016" "67"
## [14,] "8/25/2016" "112"
```



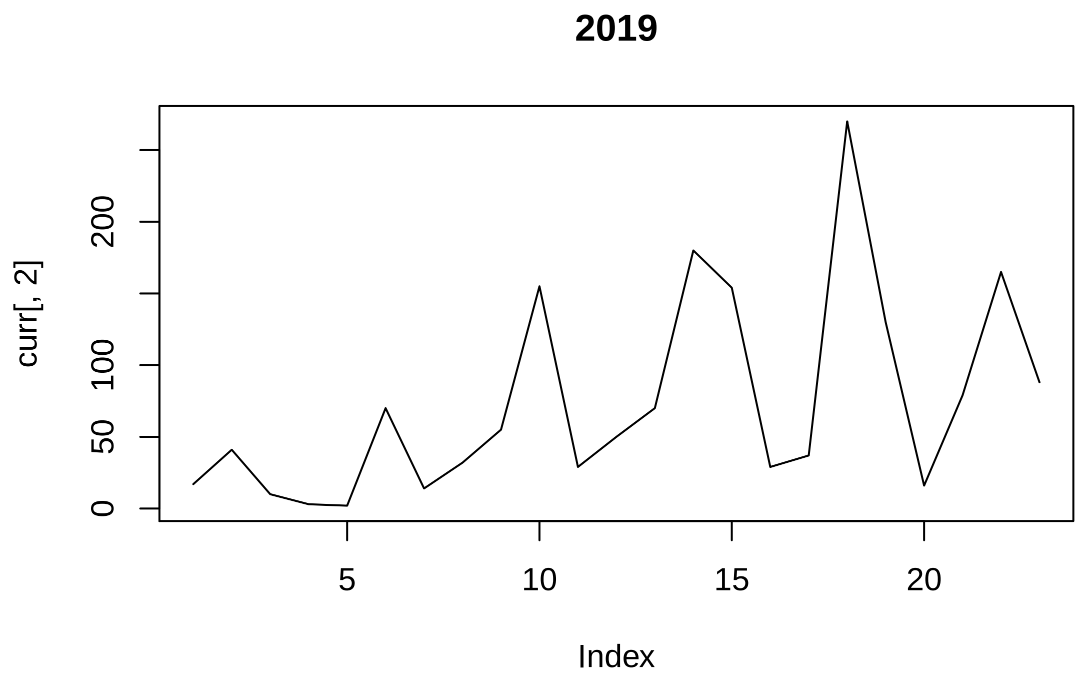
```
##       days        eggs
##  [1,] "6/7/2017"  "61"
##  [2,] "6/12/2017" "98"
##  [3,] "6/20/2017" "34"
##  [4,] "6/28/2017" "42"
##  [5,] "7/5/2017"  "74"
##  [6,] "7/13/2017" "147"
##  [7,] "7/20/2017" "390"
##  [8,] "7/28/2017" "370"
##  [9,] "8/3/2017"  "443"
## [10,] "8/9/2017"  "96"
## [11,] "8/16/2017" "1012"
## [12,] "8/23/2017" "332"
## [13,] "8/30/2017" "172"
```



```
##       days        eggs
##  [1,] "6/7/2018"  "43"
##  [2,] "6/14/2018" "181"
##  [3,] "6/21/2018" "94"
##  [4,] "6/26/2018" "204"
##  [5,] "6/27/2018" "37"
##  [6,] "6/29/2018" "585"
##  [7,] "7/2/2018"  "98"
##  [8,] "7/5/2018"  "217"
##  [9,] "7/6/2018"  "138"
## [10,] "7/9/2018"  "183"
## [11,] "7/11/2018" "44"
## [12,] "7/13/2018" "131"
## [13,] "7/16/2018" "293"
## [14,] "7/18/2018" "124"
## [15,] "7/20/2018" "166"
## [16,] "7/23/2018" "78"
## [17,] "7/25/2018" "152"
## [18,] "7/27/2018" "31"
## [19,] "7/30/2018" "285"
## [20,] "8/1/2018"  "163"
## [21,] "8/3/2018"  "91"
## [22,] "8/6/2018"  "170"
## [23,] "8/8/2018"  "70"
## [24,] "8/10/2018" "240"
## [25,] "8/13/2018" "46"
## [26,] "8/15/2018" "59"
## [27,] "8/17/2018" "43"
## [28,] "8/20/2018" "182"
## [29,] "8/22/2018" "78"
## [30,] "8/24/2018" "32"
## [31,] "8/27/2018" "28"
## [32,] "8/29/2018" "73"
## [33,] "8/31/2018" "42"
```



```
##       days        eggs
##  [1,] "6/5/2019"  "17"
##  [2,] "6/12/2019" "41"
##  [3,] "6/19/2019" "10"
##  [4,] "6/26/2019" "3"
##  [5,] "6/27/2019" "2"
##  [6,] "7/2/2019"  "70"
##  [7,] "7/3/2019"  "14"
##  [8,] "7/10/2019" "32"
##  [9,] "7/12/2019" "55"
## [10,] "7/17/2019" "155"
## [11,] "7/19/2019" "29"
## [12,] "7/24/2019" "50"
## [13,] "7/26/2019" "70"
## [14,] "7/31/2019" "180"
## [15,] "8/2/2019"  "154"
## [16,] "8/7/2019"  "29"
## [17,] "8/9/2019"  "37"
## [18,] "8/14/2019" "270"
## [19,] "8/16/2019" "130"
## [20,] "8/21/2019" "16"
## [21,] "8/23/2019" "79"
## [22,] "8/27/2019" "165"
## [23,] "8/30/2019" "88"
```

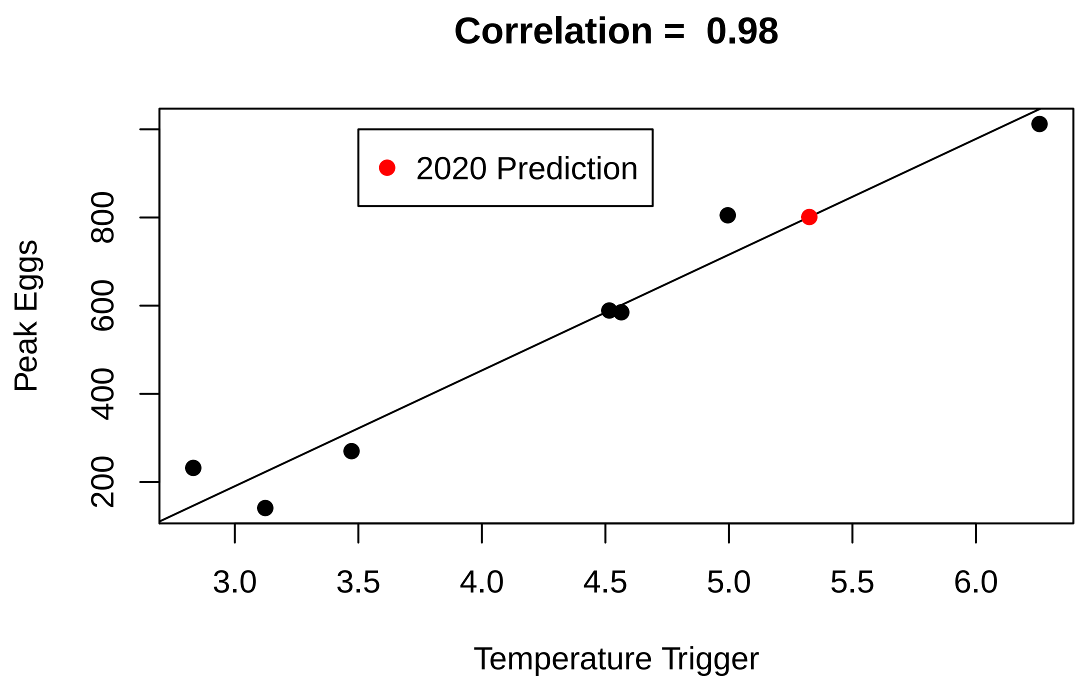


```
colnames(tdiffs)<- c("Date", "STT")
colnames(egg_max) <- c("Date", "Peak Eggs")
```

```
starts = {}
ends = {}
val1 = {}
val2 = {}
diffs = {}
for(t in c(1:8)){
  date = tdiffs[t,1]
  index = which(day[,1] == date)
  dts = day[c((index-27):index),1]

  vls = day[c((index-27):index),2]
  s = dts[which.min(as.numeric(day[c((index-27):index),2]))]

  v1 = as.numeric(vls[which.min(as.numeric(day[c((index-27):index),2]))])
  e = date
  v2 = as.numeric(day[index,2])

  starts = c(starts, s)
  ends = c(ends, e)
  val1 = c(val1, v1)
  val2 = c(val2, v2)
  diffs = c(diffs, v2-v1)
}
out <- cbind(starts,ends,val1,val2,diffs)
```

Plotting these against eachother gives us figure 2C

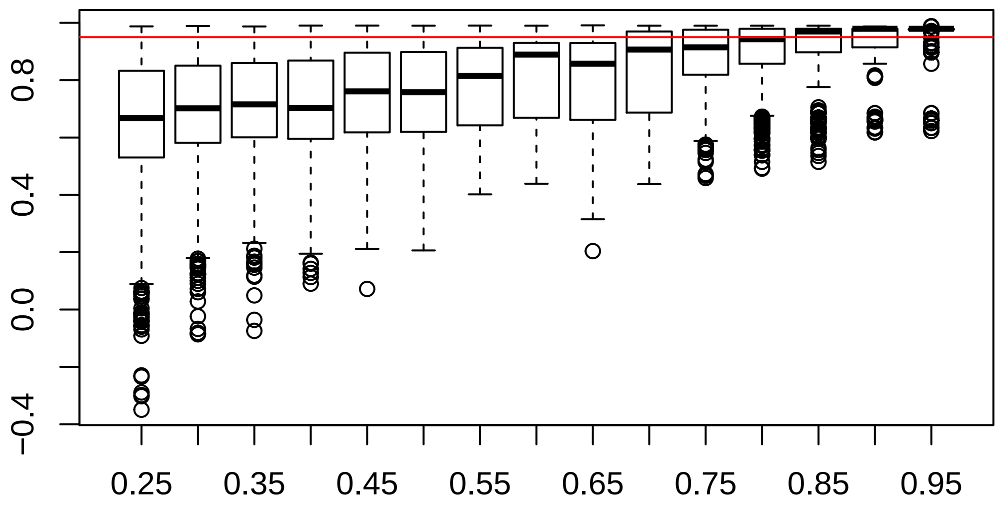
```
{
plot(tdiffs[1:7,2], egg_max[1:7,2],
     main = paste("Correlation = ",
                  round(100*cor(as.numeric(tdiffs[1:7,2]),
                                as.numeric(egg_max[1:7,2])))/100),pch = 19,
     ylab = "Peak Eggs", xlab = "Temperature Trigger")
abline(lm(as.numeric(egg_max[1:7,2])~as.numeric(tdiffs[1:7,2])))
points(5.3255688888889, 5.3255688888889*262-594.2, col = 'red', pch = 19)
legend(3.5,1000,legend =  "2020 Prediction", col = "red", pch = 19)
}
```



Here I do random subsamples

```
find_egg_max <- function(start, end, sample_size = 0){

  if(is.character(start)){
  start = paste(start, "-2000", sep ="")
  end = paste(end, "-2000", sep ="")
  start = yday(as.Date(start, format = "%m-%d-%Y"))
  end = yday(as.Date(end, format = "%m-%d-%Y"))
  }

  vals = eggs[yday(as.Date(eggs[,1], format = "%m/%d/%Y"))>start &
         yday(as.Date(eggs[,1], format = "%m/%d/%Y"))<end, ]
  vals = vals[!is.na(vals[,1]),]

  days = yday(as.Date(eggs[,1], format = "%m/%d/%Y"))


  years = year(as.Date(vals[,1], format = "%m/%d/%Y"))
  unique_years = unique(year(as.Date(vals[,1], format = "%m/%d/%Y")))
  out<- array(NA, dim = c(7,2))
  loop = 1
  for(y in c(2013:2019)){
    curr = vals[years == y,]
    if(sample_size != 0 ){
      curr = curr[sample(1:nrow(curr),round(nrow(curr)*sample_size)),]
```

```r
    }
    i =which.max(as.numeric(curr[,2]))
    out[loop,]<- curr[i,]
    loop = loop+1
  }
  return(out)

}

tdiffs = find_biggest_diff(day, start = "4-1", end = "6-17", delay = 27)

cors = {}
for(s in seq(.25,.95,.05)){
  cors_curr = {}
  for(trial in c(1:1000)){
    egg_max2 <- find_egg_max("6-1", "9-1", sample_size = s)
    c= cor(as.numeric(tdiffs[1:7,2]), as.numeric(egg_max2[1:7,2]))
    cors_curr=c(cors_curr,c)
  }
  cors = cbind(cors, cors_curr)
}
boxplot(cors, names = seq(.25,.95,.05))
abline(h = .95, col = 'red')
```

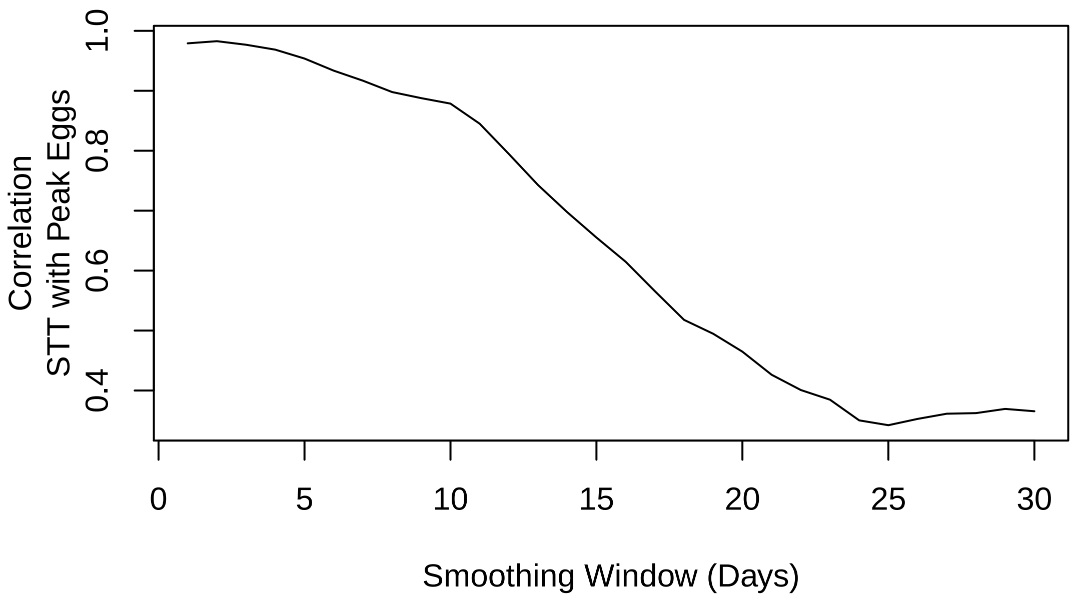


```r
# colnames(tdiffs)<- c("Date", "STT")
# colnames(egg_max) <- c("Date", "Peak Eggs")
# plot(tdiffs[1:7,2], egg_max[1:7,2],
#      main = paste("Correlation = ",
#                   round(100*cor(as.numeric(tdiffs[1:7,2]),
#                                 as.numeric(egg_max[1:7,2])))/100),pch = 19,
#      ylab = "Peak Eggs", xlab = "Temperature Trigger")
# abline(lm(as.numeric(egg_max[1:7,2])~as.numeric(tdiffs[1:7,2])))
```

Smoothing Analysis (Figure 3)

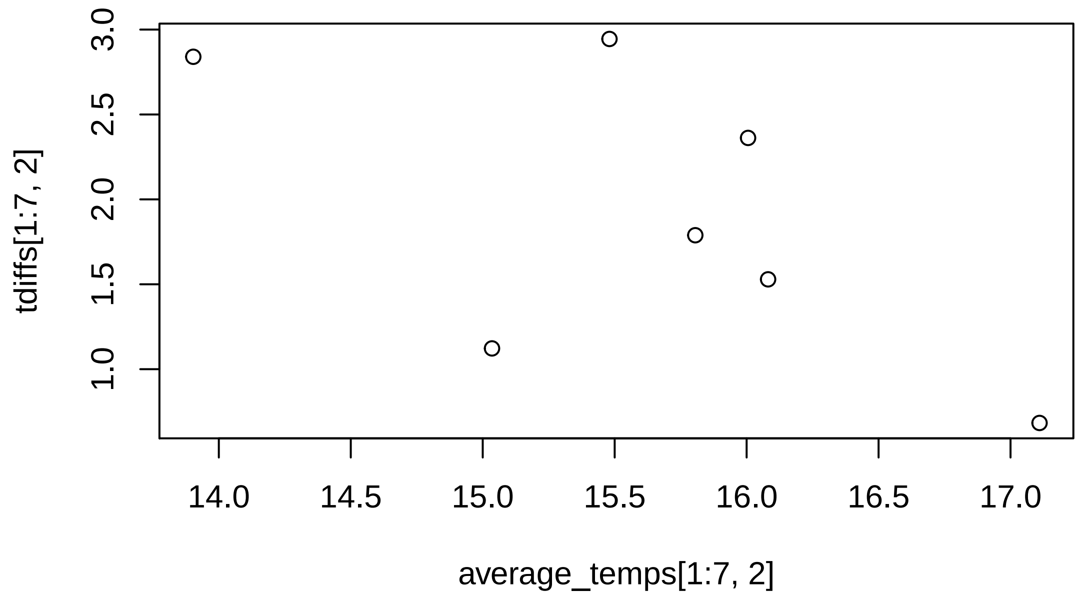
First we define our smoothing function, then recalculate the STT, and subsequently the correlation between STT and peak egg abundance for smoothing windows from 1 - 30 days.

```r
smooth <- function(ts, m){
  avg = array(mean(ts), dim = c(m-1,1))
  for(sm in c(m:length(ts))){
    curr = mean(ts[(sm-m+1):sm])
    avg = c(avg, curr)
  }
  return(avg)
}

cors = {}
for(m in c(1:30)){
  newday = cbind(day[,1],smooth(as.numeric(day[,2]), m)) #The smoothed timeseries

  tdiffs = find_biggest_diff(newday, start = "4-1", end = "6-17", delay = 27) #Find new STT
  cr = cor(as.numeric(tdiffs[year(as.Date(tdiffs[,1]))>2012
                            & year(as.Date(tdiffs[,1]))<2020,2]),
           as.numeric(egg_max[year(as.Date(egg_max[,1],
                                           format = "%m/%d/%Y"))>2012 &
                               year(as.Date(egg_max[,1],
                                            format = "%m/%d/%Y"))<2020,2]))
  cors = c(cors,cr )
}

{
plot(c(1:30), cors, type = 'l',xlab = "Smoothing Window (Days)", ylab = "" )
title(ylab = c("Correlation", "STT with Peak Eggs"))
}
```



Here we do a bunch of correlations

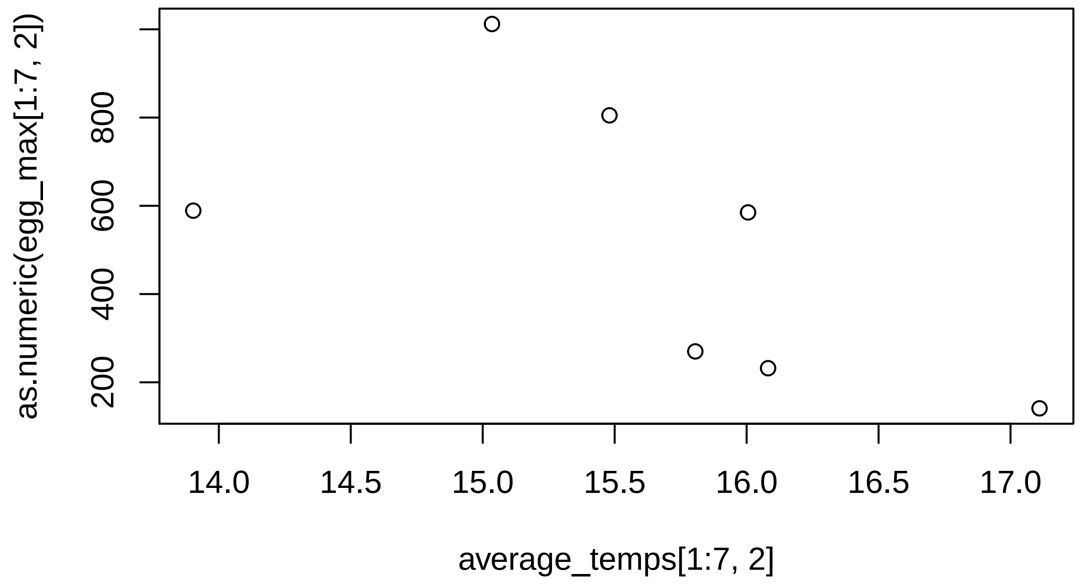
```r
plot(average_temps[1:7,2], tdiffs[1:7,2])
```



```
cor.test(as.numeric(average_temps[1:7,2]), as.numeric(tdiffs[1:7,2]))
```

```
##
##  Pearson's product-moment correlation
##
## data:  as.numeric(average_temps[1:7, 2]) and as.numeric(tdiffs[1:7, 2])
## t = -1.7287, df = 5, p-value = 0.1444
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.9343378  0.2621962
## sample estimates:
##        cor
## -0.6116271
```

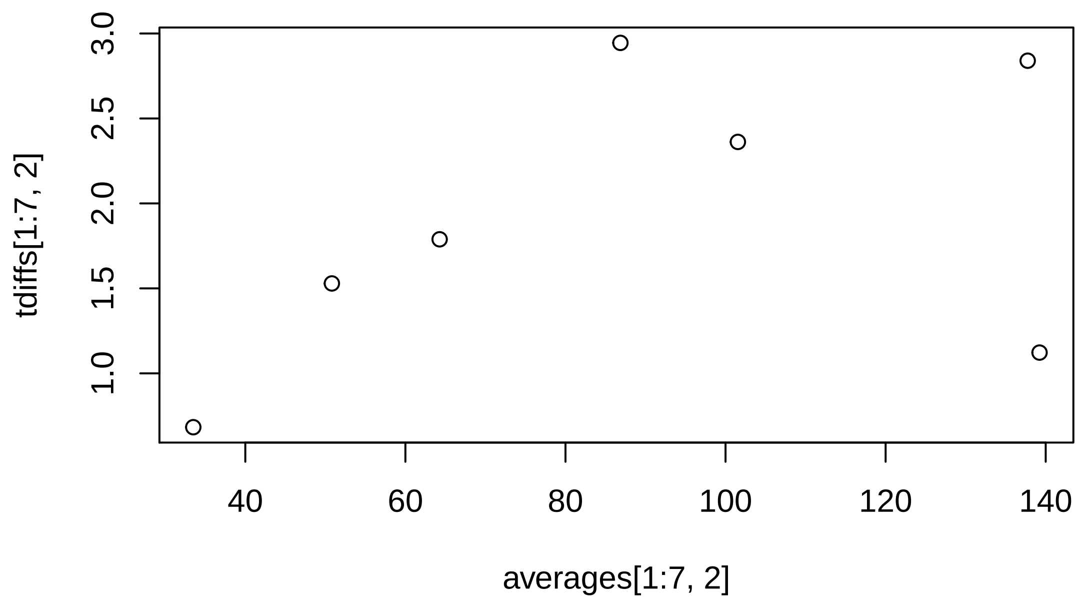
```
plot(average_temps[1:7,2], as.numeric(egg_max[1:7,2]))
```



```r
cor.test(as.numeric(average_temps[1:7,2]), as.numeric(egg_max[1:7,2]))
```

```
##
##  Pearson's product-moment correlation
##
## data:  as.numeric(average_temps[1:7, 2]) and as.numeric(egg_max[1:7, 2])
## t = -1.7086, df = 5, p-value = 0.1482
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.9334258  0.2688262
## sample estimates:
##       cor
## -0.607143
```

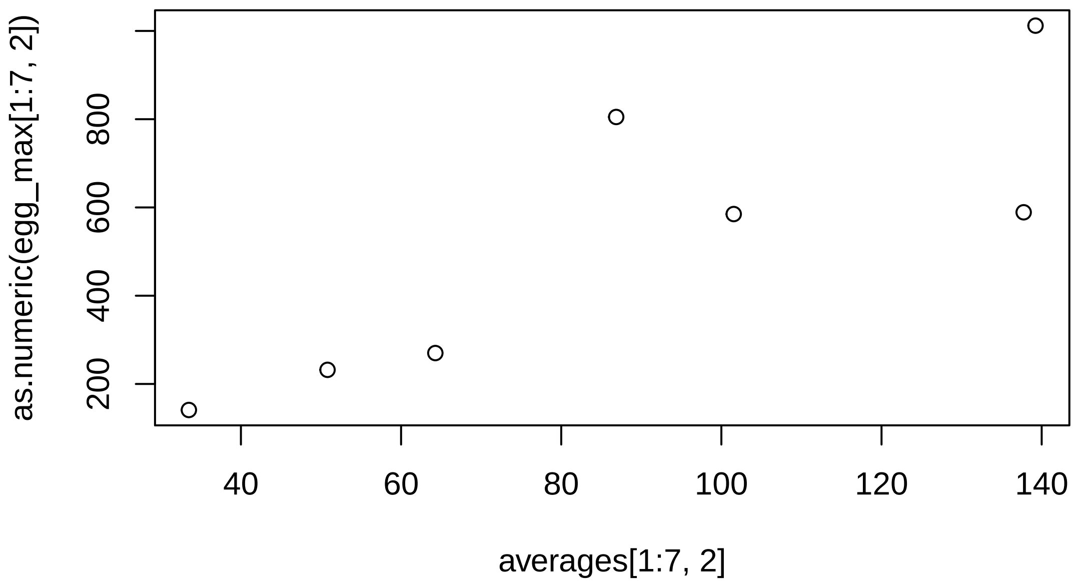
```r
plot(averages[1:7,2], tdiffs[1:7,2])
```



```r
cor.test(as.numeric(averages[1:7,2]), as.numeric(tdiffs[1:7,2]))
```

```
##
##  Pearson's product-moment correlation
##
## data:  as.numeric(averages[1:7, 2]) and as.numeric(tdiffs[1:7, 2])
## t = 1.114, df = 5, p-value = 0.3159
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.4624060  0.8975742
## sample estimates:
##      cor
## 0.445935
```

```r
plot(averages[1:7,2], as.numeric(egg_max[1:7,2]))
```



```
cor.test(as.numeric(average_temps[1:7,2]), as.numeric(egg_max[1:7,2]))
```

```
##
##  Pearson's product-moment correlation
##
## data:  as.numeric(average_temps[1:7, 2]) and as.numeric(egg_max[1:7, 2])
## t = -1.7086, df = 5, p-value = 0.1482
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.9334258  0.2688262
## sample estimates:
##       cor
## -0.607143
```