Workshop 2_Nonlinear Models

Katie Stansbury

1/17/2020

Objectives

The primary objectives of this analysis is to fit monthly light response curves for Harvard forest to understand annual patterns ecosystem photosynthetic potential and respiration rates in temperate mixed forests…

Methods

In order to fit a curve to the termperature data from Harvard Forest, we used data taken from the Environmental Measurement Station Eddy Flux Tower (EMS). This data was then loaded into R Studio, where the following code was used to attempt to fit the temperature data to the Arrhenius Curve.

```
library(nlstools)
```

```
##
## 'nlstools' has been loaded.
```

```
## IMPORTANT NOTICE: Most nonlinear regression models and data set examples
```

```
## related to predictive microbiolgy have been moved to the package 'nlsMicrobio'
```

```r
load("~/Downloads/NLM_Workshop.RData")
View(night)
####Create a dataframe to store month parameter values (parms.Month.night).
#### Selfstart for the trc:
trcModel <- function(TA, a, b) {
  y=a * exp(b*TA)
  return(y)
}

#### Create a function to find initial values for the selfstart function:
trc.int <- function (mCall, LHS, data){
  x <- data$TA
  y <- data$NEE

  a <-1.00703982 + -0.08089044* (min(na.omit(y)))
  b <- 0.051654 + 0.001400 * (min(na.omit(y)))

  value = list(a, b)
  names(value) <- mCall[c("a", "b")]
  return(value)
}

#### Selfstart Function
SS.trc <- selfStart(model=trcModel,initial= trc.int)

iv <- getInitial(NEE ~ SS.trc('TA', "a", "b"),
                 data = night[which(night$MONTH == c(1,2,3,11,12)),])
```

```
## Warning in night$MONTH == c(1, 2, 3, 11, 12): longer object length is not a
## multiple of shorter object length
```

```r
iv <- getInitial(NEE ~ SS.trc('TA', "a", "b"),
                 data = night[which(night$MONTH == c(1,2,3,11,12)),])
```

```
## Warning in night$MONTH == c(1, 2, 3, 11, 12): longer object length is not a
## multiple of shorter object length
```

```r
iv
```

```
## $a
## [1] 1.42767
##
## $b
## [1] 0.044374
```

```
####nls for new data set
y = nls( NEE ~ a * exp(b*TA), night[which(night$MONTH ==c(12,11,1,2,3)),],
         start=list(a= iv$a , b= iv$b),
         na.action=na.exclude, trace=F, control=nls.control(warnOnly=T))
```

```
## Warning in night$MONTH == c(12, 11, 1, 2, 3): longer object length is not a
## multiple of shorter object length
```

```
## Warning in night$MONTH == c(12, 11, 1, 2, 3): longer object length is not a
## multiple of shorter object length
```

```
summary(y)
```

```
##
## Formula: NEE ~ a * exp(b * TA)
##
## Parameters:
##   Estimate Std. Error t value Pr(>|t|)
## a 1.445838   0.031334   46.14   <2e-16 ***
## b 0.046737   0.003163   14.78   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.402 on 2208 degrees of freedom
##
## Number of iterations to convergence: 4
## Achieved convergence tolerance: 2.661e-06
##   (951 observations deleted due to missingness)
```

```
#### Create Dataframe to store the data: Creates a blank data frame for data to be en
tered into

parms.Month <- data.frame(

  MONTH=numeric(),

  a=numeric(),

  b=numeric(),

  a.pvalue=numeric(),

  b.pvalue=numeric(), stringsAsFactors=FALSE, row.names=NULL)

parms.Month[1:12, 1] <- seq(1,12,1) # Creates time file to merge with parm file

####Functions: based off of initial values
nee.night <- function(dataframe){y.df = nls(NEE ~ a * exp(b*TA),
                                            dataframe, start=list(a= iv$a , b=iv$b ),
                                            na.action=na.exclude, trace=F,
                                            control=nls.control(warnOnly=T))
y.df <- as.data.frame(cbind(t(coef(summary(y.df))[1:2, 1]), t(coef(summary(y.df)) [1:
2, 4])))
names(y.df) <- c("a", "b", "a.pvalue", "b.pvalue")
return(y.df)}

#### This loop fits monthly models (1:12):
####Bootstraps to create more data based off of function created by initial values to
see if created data still fits the model

try(for(j in unique(night$MONTH)){
  print(j)
  iv <- getInitial(NEE ~ SS.trc('TA', "a", "b"), data = night[which(night$MONTH == j)
,])
  y4 <- try(nee.night(night[which(night$MONTH == j),]), silent=T) # Fit night model
  try(parms.Month[c(parms.Month$MONTH == j ), 2:5 ] <- cbind(y4), silent=T)
  rm(y4)
}, silent=T)
```

```
## [1] 11
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 12
## [1] 7
## [1] 8
## [1] 9
## [1] 1
## [1] 10
```

```
####bunch of warnings because code broke at a=iv$a, meaning loop based on initial val
ues and bootstrapping didn't work
####created data doesn't fit the model

#### Create file to store parms and se: maybe making dataframe to store data created
by bootstrapping?
####File name is boot.NEE
####Created a file with the column Month and 4 other columns
boot.NEE <- data.frame(parms.Month[, c("MONTH")]); names (boot.NEE) <- "MONTH"

boot.NEE$a.est<- 0

boot.NEE$b.est<- 0

boot.NEE$a.se<- 0

boot.NEE$b.se<- 0

#### Night Model:

for ( j in unique(boot.NEE$MONTH)){
  print(j)
  y1 <-night[which(night$MONTH == j),]
  iv <- getInitial(NEE ~ SS.trc('TA',"a", "b"), data = y1)
  night.fit <- nls(NEE ~ a * exp(b-TA),
                   data=y1, start=list(a= iv$a , b=iv$b ),
                   na.action=na.exclude, trace=F,
                   control=nls.control(warnOnly=T))
  results <- nlsBoot(night.fit, niter=100 )
  a <- t(results$estiboot)[1, 1:2]
  names(a) <- c('a.est', 'b.est')
  b <- t(results$estiboot)[2, 1:2]
  names(b) <- c('a.se', 'b.se')
  c <- t(data.frame(c(a,b)))
  boot.NEE[c(boot.NEE$MONTH == j), 2:5] <- c[1, 1:4]
  rm(day.fit, a, b, c, results, y1)
}
```

```
## [1] 1
```

```
## Warning in nls(NEE ~ a * exp(b - TA), data = y1, start = list(a = iv$a, :
## singular gradient
## Error in nlsBoot(night.fit, niter = 100): Procedure aborted: the fit only converge
d in 1 % during bootstrapping
```

```
trc <- merge( parms.Month, boot.NEE)

####new trc with the zeros and the p values and such
trc
```

MO... <dbl>	a <dbl>	b <dbl>	a.pvalue <dbl>	b.pvalue <dbl>	a.est <dbl>	b.est <dbl>	a.se <dbl>	b.se <dbl>
1	1.282263	0.02790180	1.525352e-220	3.018140e-13	0	0	0	0
2	1.235765	0.03136320	3.371053e-312	1.048487e-17	0	0	0	0
3	1.100977	0.03822106	0.000000e+00	3.384413e-36	0	0	0	0
4	1.270271	0.04778224	9.364373e-119	2.004514e-28	0	0	0	0
5	1.755597	0.05711204	5.909270e-83	1.960327e-57	0	0	0	0
6	2.400125	0.03898796	5.739331e-21	1.561759e-10	0	0	0	0
7	2.005208	0.04542368	2.136097e-11	4.612000e-09	0	0	0	0
8	4.798788	-0.01422836	4.284395e-10	1.053243e-01	0	0	0	0
9	1.821047	0.03695793	3.277657e-23	2.662523e-09	0	0	0	0
10	1.679584	0.03924262	1.563406e-84	7.956977e-20	0	0	0	0

1-10 of 12 rows Previous **1** 2 Next

```r
####subset the data by month
jan <- subset(night, night$MONTH==01)
feb <- subset(night, night$MONTH==02)
mar <- subset(night, night$MONTH==03)
apr <- subset(night, night$MONTH==04)
may <- subset(night, night$MONTH==05)
jun <- subset(night, night$MONTH==06)
jul <- subset(night, night$MONTH==07)
aug <- subset(night, night$MONTH==08)
sep <- subset(night, night$MONTH==09)
oct <- subset(night, night$MONTH==10)
nov <- subset(night, night$MONTH==11)
dec <- subset(night, night$MONTH==12)

####determine what TA values work for model
####know from running model before that apr-oct don't work in the model
####model is based off TA values, so something must be throwing the model off
####Determine min and max of TA for each month

month=c("jan","feb","mar","apr","may","jun","jul","aug","sep","oct","nov","dec")
min=c(min(jan$TA),min(feb$TA),min(mar$TA),min(apr$TA),min(may$TA),min(jun$TA),min(jul
$TA),min(aug$TA),min(sep$TA),min(oct$TA),min(nov$TA),min(dec$TA))
max=c(max(jan$TA),max(feb$TA),max(mar$TA),max(apr$TA),max(may$TA),max(jun$TA),max(jul
$TA),max(aug$TA),max(sep$TA),max(oct$TA),max(nov$TA),max(dec$TA))

####subsetting data to fit a curve
y1 <-night[which(night$MONTH == jan),]
y2 <-night[which(night$MONTH == feb),]
y3 <-night[which(night$MONTH == mar),]
y11 <-night[which(night$MONTH == nov),]
y12 <-night[which(night$MONTH == dec),]

#### Determine starting values for each month
iv1 <- getInitial(NEE ~ SS.trc('TA', "a", "b"), data = y1)
iv2 <- getInitial(NEE ~ SS.trc('TA', "a", "b"), data = y2)
iv3 <- getInitial(NEE ~ SS.trc('TA', "a", "b"), data = y3)
iv11 <- getInitial(NEE ~ SS.trc('TA', "a", "b"), data = y11)
iv12 <- getInitial(NEE ~ SS.trc('TA', "a", "b"), data = y12)

####fit curve
night.fit1 <- nls( NEE ~ a * exp(b-TA), data=y1,
                start=list(a= iv1$a, b= iv1$b),
                na.action=na.exclude, trace=F, control=nls.control(warnOnly=T))
```

```
## Error in nlsModel(formula, mf, start, wts): singular gradient matrix at initial pa
rameter estimates
```

```
night.fit2 <- nls( NEE ~ a * exp(b-TA), data=y2,
                   start=list(a= iv2$a, b= iv2$b),
                   na.action=na.exclude, trace=F, control=nls.control(warnOnly=T))
```

```
## Error in nlsModel(formula, mf, start, wts): singular gradient matrix at initial pa
rameter estimates
```

```
night.fit3 <- nls( NEE ~ a * exp(b-TA), data=y3,
                   start=list(a= iv3$a, b= iv3$b),
                   na.action=na.exclude, trace=F, control=nls.control(warnOnly=T))
```

```
## Error in nlsModel(formula, mf, start, wts): singular gradient matrix at initial pa
rameter estimates
```

```
night.fit11 <- nls( NEE ~ a * exp(b-TA), data=y11,
                   start=list(a= iv11$a, b= iv11$b),
                   na.action=na.exclude, trace=F, control=nls.control(warnOnly=T))
```

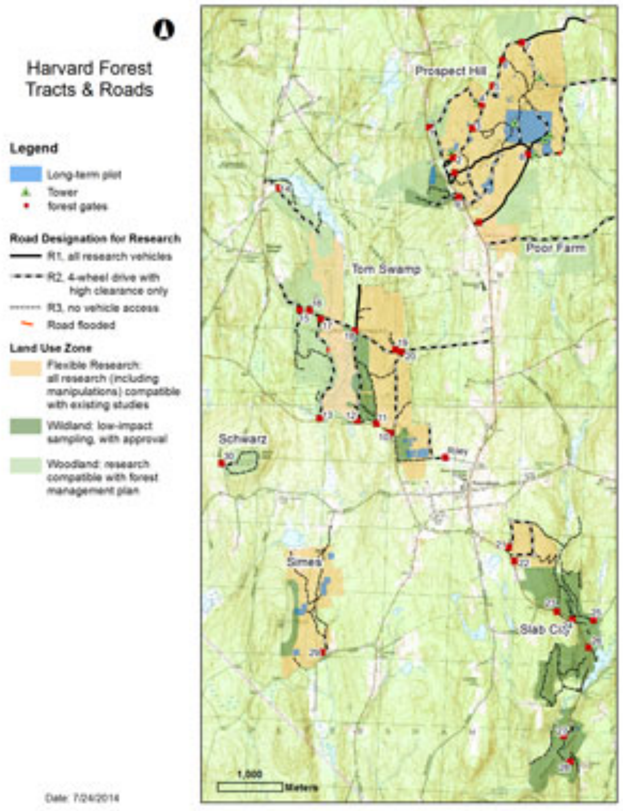
```
## Error in nlsModel(formula, mf, start, wts): singular gradient matrix at initial pa
rameter estimates
```

```
night.fit12 <- nls( NEE ~ a * exp(b-TA), data=y12,
                   start=list(a= iv12$a, b= iv12$b),
                   na.action=na.exclude, trace=F, control=nls.control(warnOnly=T))
```

```
## Warning in nls(NEE ~ a * exp(b - TA), data = y12, start = list(a = iv12$a, :
## singular gradient
```

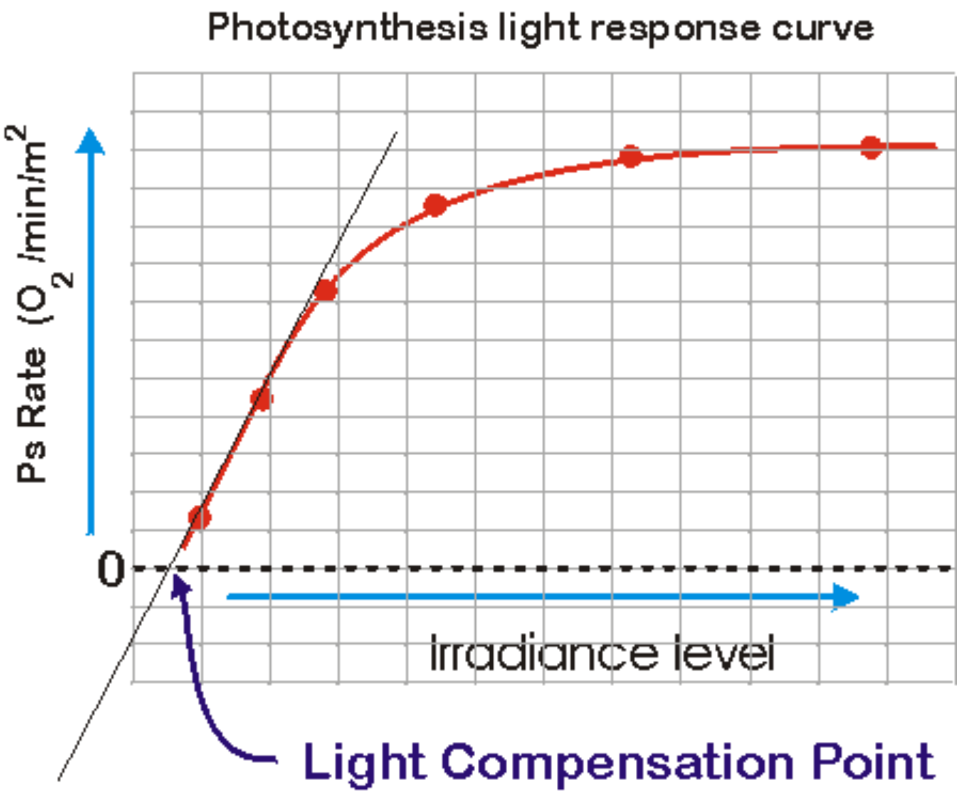
Site Information (include a map of the harvard forest site)

The data for this analysis was taken from the Environmental Measurement Station Eddy Flux Tower (EMS). This forest is located in Massachusetts and is a cool, moist temperate forest that experiences average temperatures between 20 degrees C to -7 degrees C throughout the year. Precipitation is distributed evenly throughout the year and averages about 110 cm per year. The dominant plant species in this area are Red oak (*Quercus rubra*), Red maple (*Acer rubrum*), Black birch (*Betula lenta*), White pine (*Pinus strobus*), and Eastern hemlock (*Tsuga canadensis*).


Harvard Forest Tracts & Roads
Legend
Long-term plot
Tower
forest gates
Road Designation for Research
R1, all research vehicles
R2, 4-wheel drive with high clearance only
R3, no vehicle access
Road flooded
Land Use Zone
Flexible Research: all research (including manipulations) compatible with existing studies
Wildland: low-impact sampling, with approval
Woodland: research compatible with forest management plan
Prospect Hill
Poor Farm
Tom Swamp
Schwarz
Riley
Simes
Slab City
1,000
Meters
Date: 7/24/2014

Photosynthetic Potential

Photosynthetic potential is the highest rate of photosynthesis that a specific plant or part of a plant can reach. This potential depends on a number of things including the species of the plant and the types of leaves that the plant grows as well as climate variables like temperature and amount of sunlight. This potential is calculated by using a light response curve, where you compare the photosynthetically active radiation to net photosynthesis.



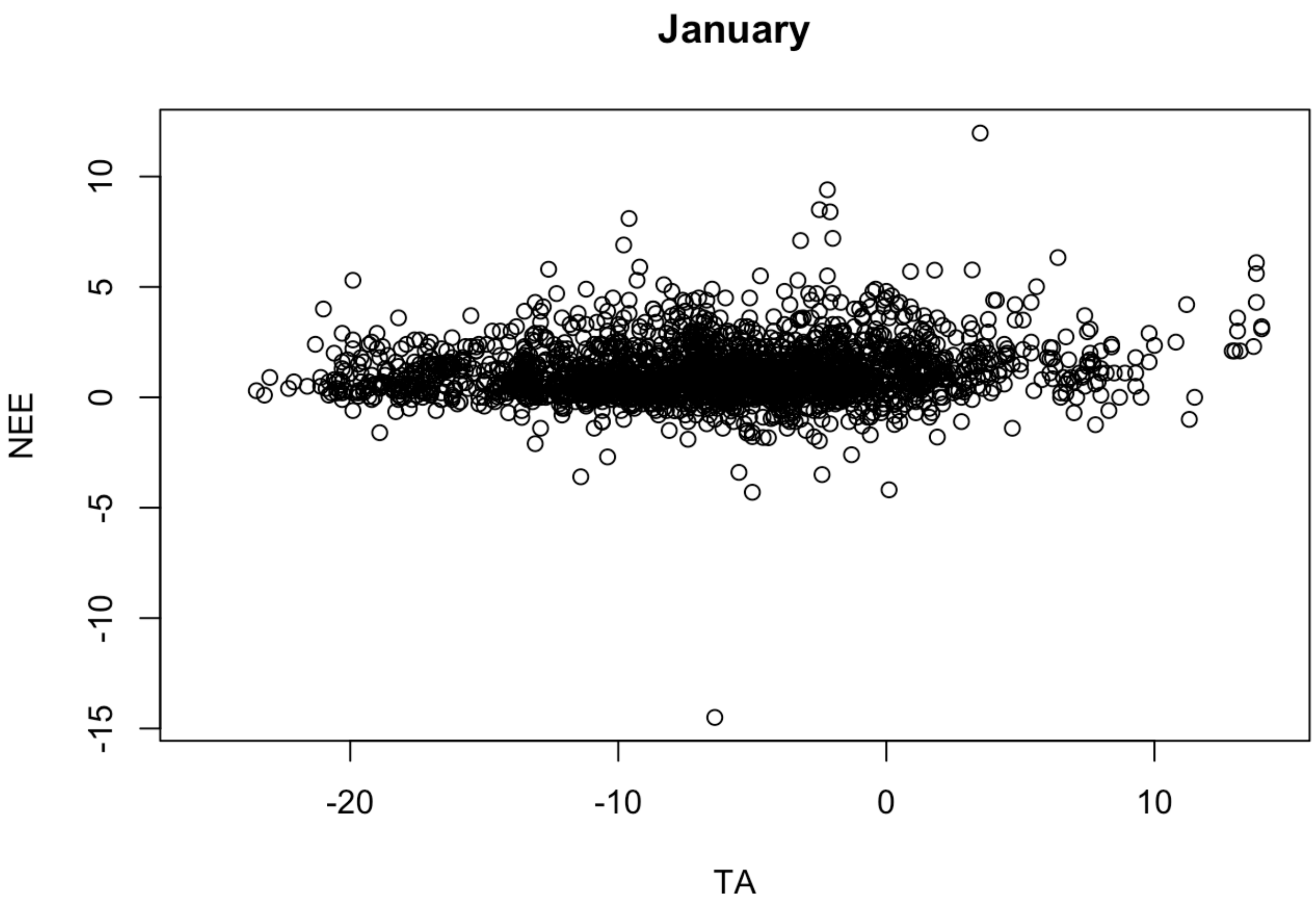
Ecosystem Respiration

Ecosystem respiration is the sum of all respiration occurring in one ecosystem. This means that any organism that converts organic carbon into carbon dioxide contributes to ecosystem respiration. Ecosystem respiration contributes to the calculation of Net Ecosystem Exchange (NEE), which is calculated by the difference between photosynthetic and respiratory CO2 exchanges.

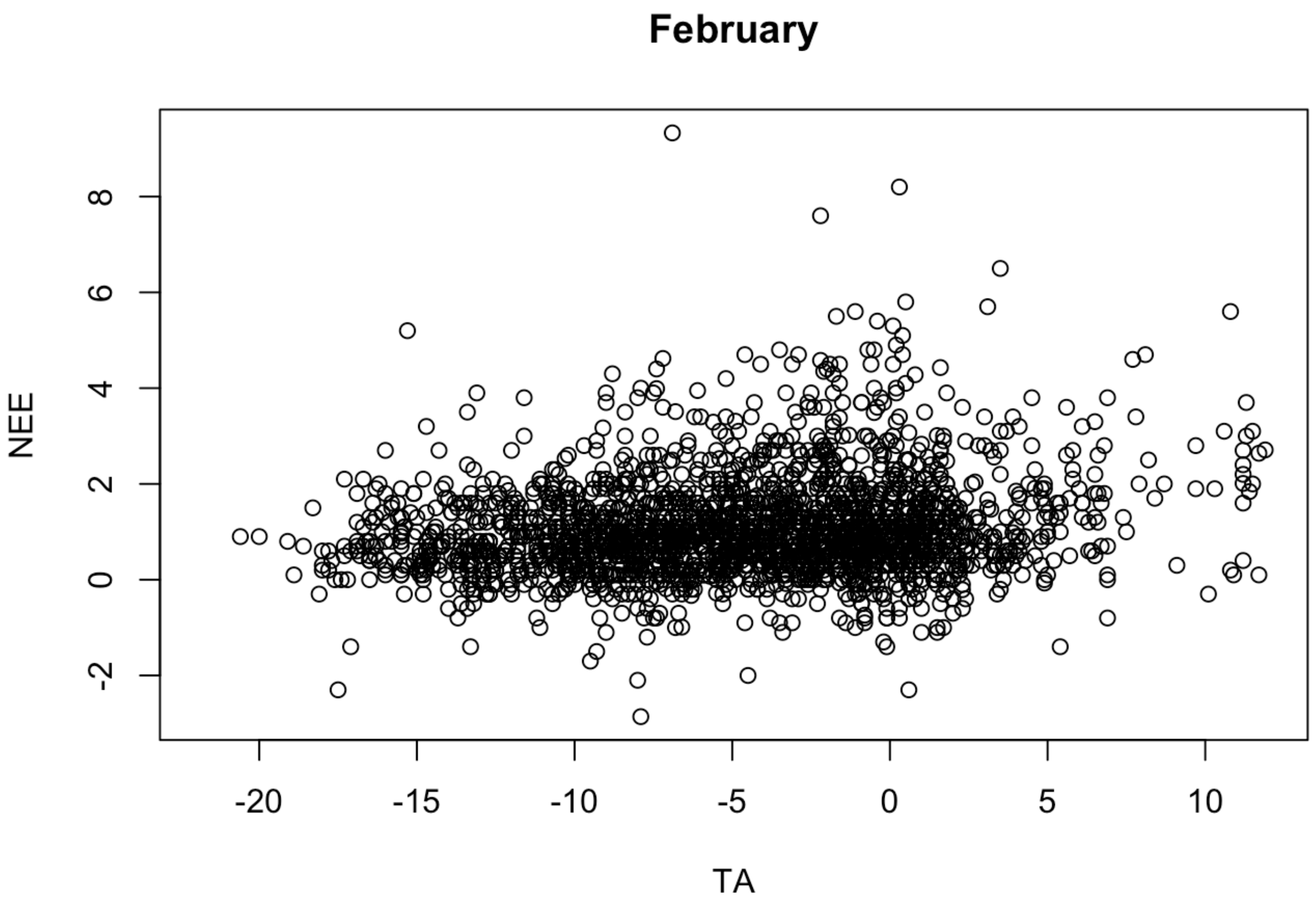
Results (at least 1 plot and one table)

After attempting to complete this challenge, I was unable to fit a curve to any of the months due to the singular gradient matrix error continually occurring no matter how much I changed the coding. I do know that the Arrhenius Equation should have worked for January, February, March, November, and December but not for the summer months (April, June, July, August, September, October). When looking at the plots of these months, we can see that they all follow a similar trend of having relatively stable NEE until reaching higher temperatures, which is when we see a gentle increase in NEE. In the summer months, we do not find this kind of curve which explains why the Arrhenius Equation did not work for April through September.

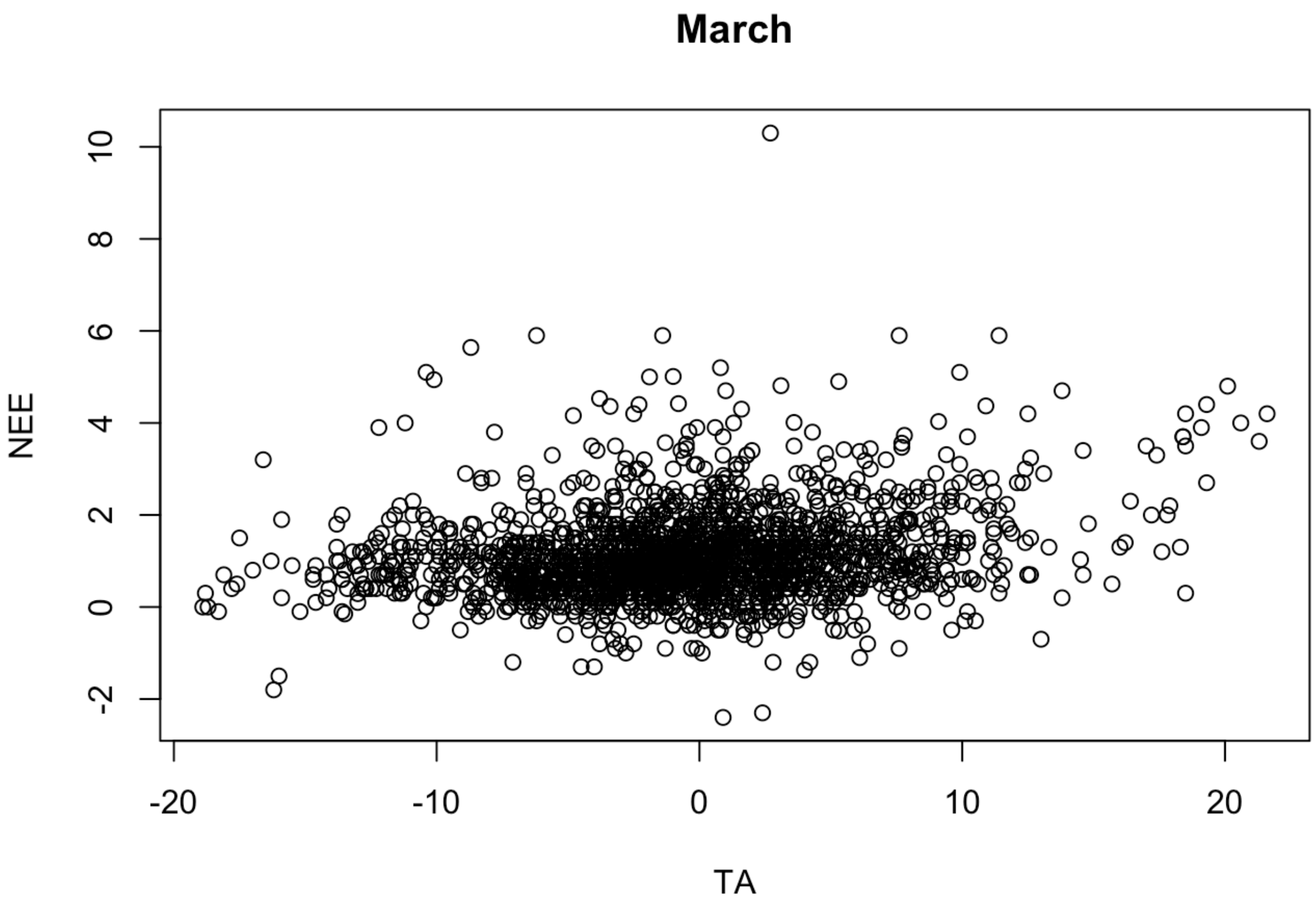
```
plot(NEE ~ TA, data = jan, main = "January")
```



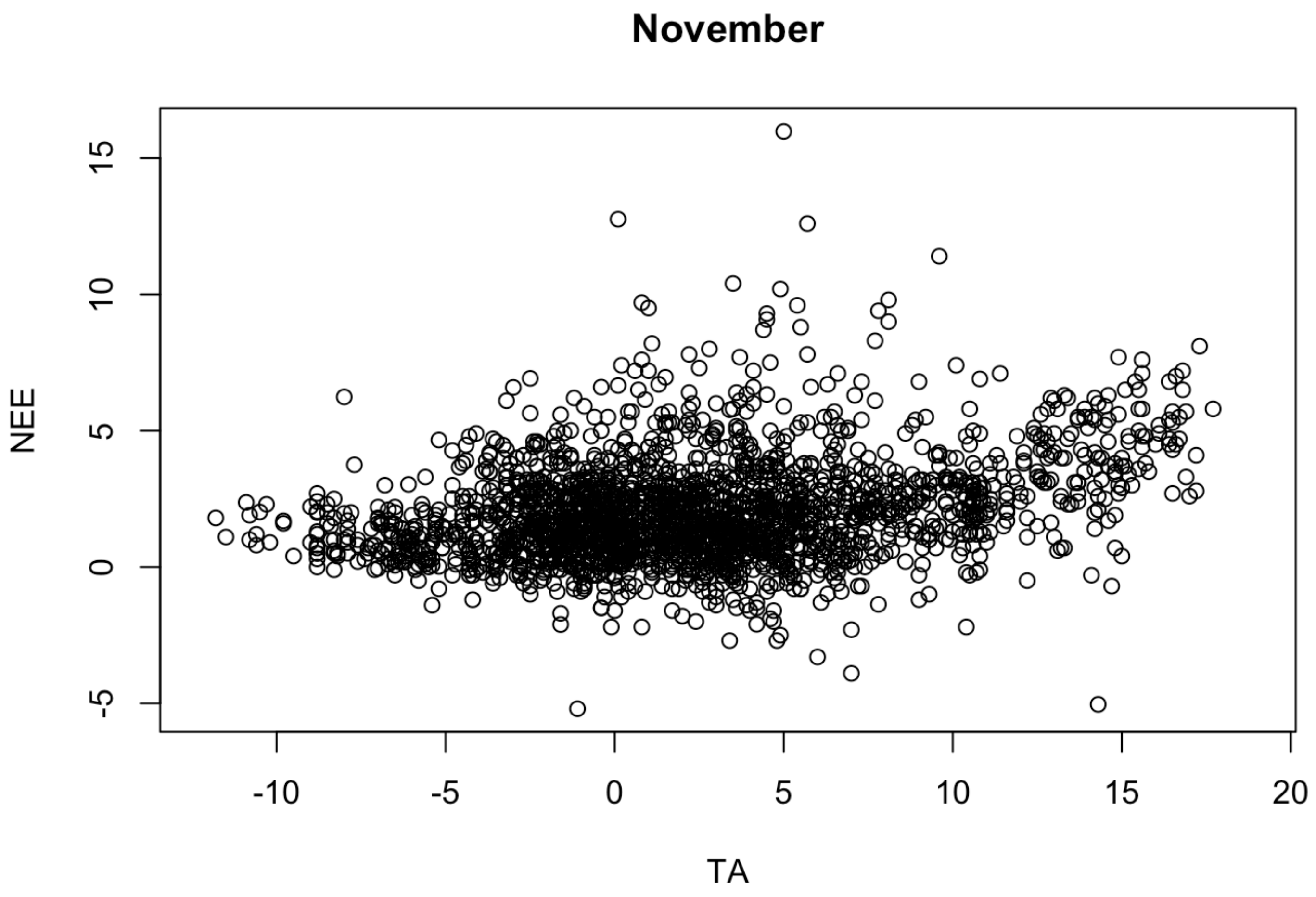
```
plot(NEE ~ TA, data = feb, main = "February")
```



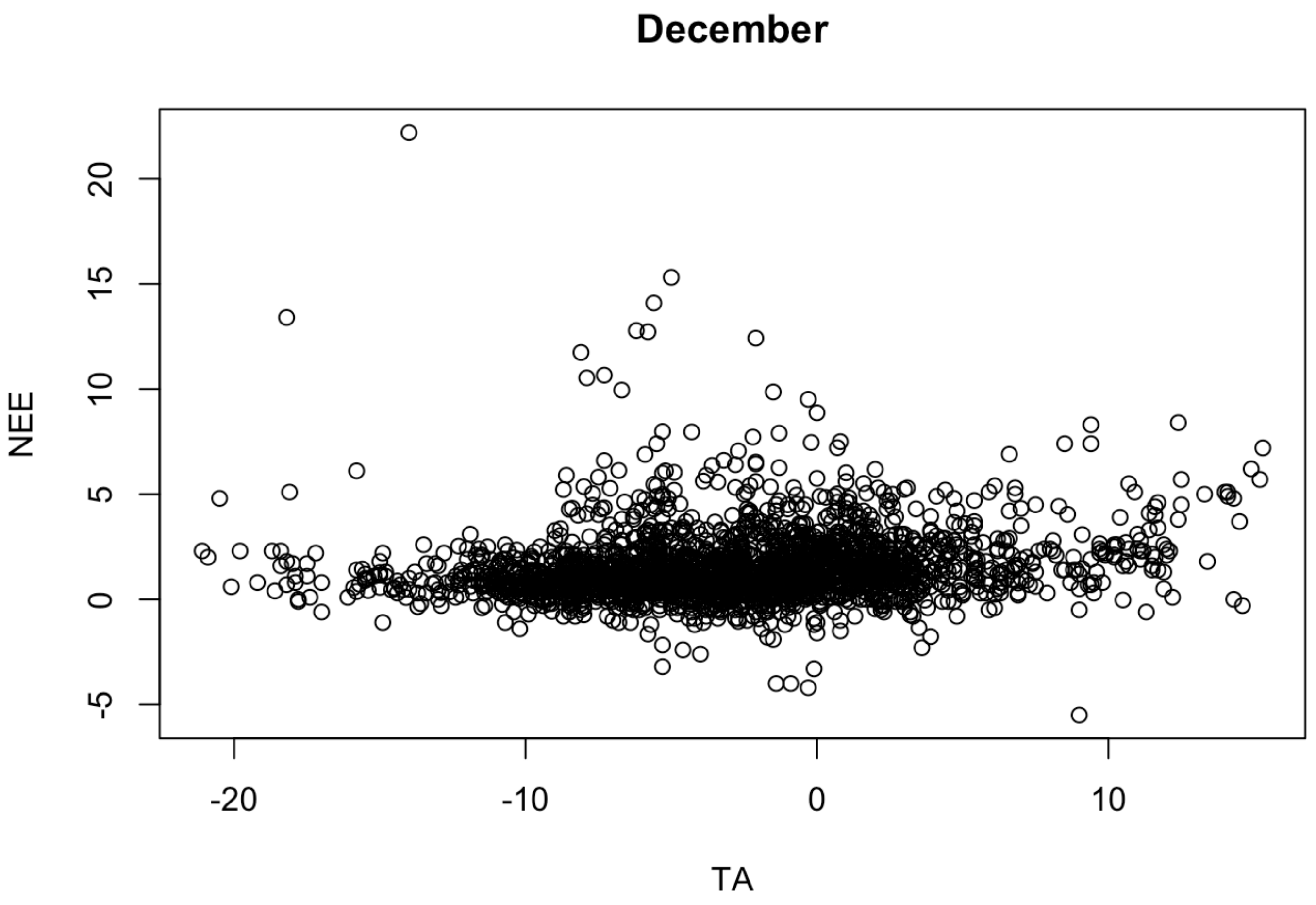
```
plot(NEE ~ TA, data = mar, main = "March")
```



```
plot(NEE ~ TA, data = nov, main = "November")
```



```
plot(NEE ~ TA, data = dec, main = "December")
```



Minimum and Maximum Temperatures for All Months

Month	Min	Max
jan	-25.5	14.2
feb	-21.8	11.9
mar	-18.9	21.6
apr	-6.7	25.3
may	0.0	25.2
jun	4.4	26.6
jul	8.7	28.1
aug	8.6	29.0
sep	0.9	26.2
oct	-2.9	22.2

nov	-12.2	18.9
dec	-21.1	15.3

Discussion (1 paragraph)

Although I was not able to successfully fit the curve for the nonsummer months on R, I understand what the program was trying to accomplish. The summer months months had higher variation in the warmer temperatures for NEE, but the 5 months where the curve was supposed to fit had a distinct, gentle rise at warmer temperatures. I believe that the summer months did not fit the Arrhenius Curve model because there is such a wide range of respiration rates for the forest depending on the amount of precipitation or sunshine that the Harvard Forest receives every year. There was too much variation in NEE for all temperatures in the summer months, which did not allow the program to fit the Arrhenius Equation to it. Also, temperatures during the summer months are normally higher and could lead to decreases in NEE if it gets too hot for the plant to survive and function normally.