Emery_Assignment_Nonlinear_Temp_Night

Challange:

Fit monthly temperature response curves using a similar approach with the night data from harv (night).

NEE ~ a exp(b□TA)

a is the base respiration rate when air temperature is 0 rC and b is an empirical coefficient.

Workflow: 1. Create a dataframe to store month parameter values (parms.Month). 2. Write a function to the fit model and extract parameters (nee.night). 3. Write a loop to fit monthly curves and add parameters to a dataframe (parms.Month). 4. Bootstrapping for error estimation.

```
# Set your working directory to the Malone NLM Workshop
rm(list=ls())

load("C:/Users/Mere/Desktop/FIU Courses/Spring 2020/Quantative Ecology_WS/Nonlinear/NLM_Workshop.R
Data")

library(nlstools)
```

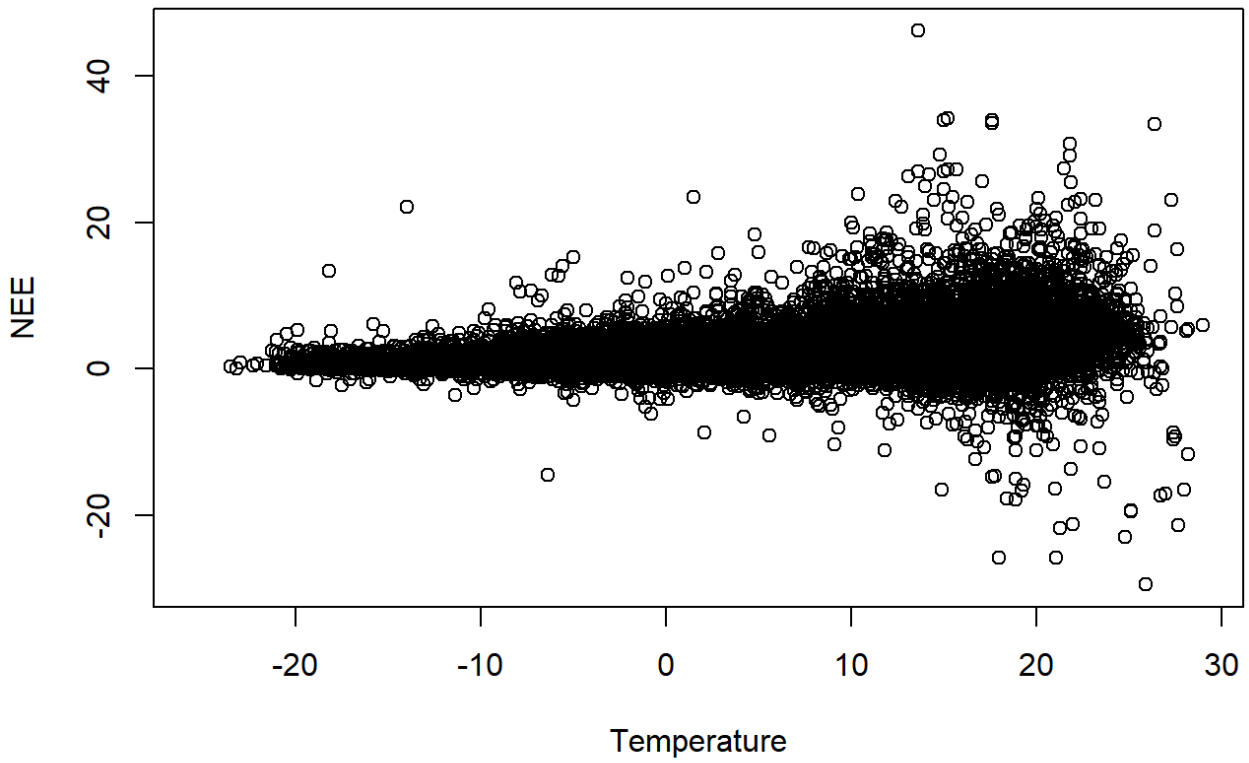
```
##
## 'nlstools' has been loaded.
```

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

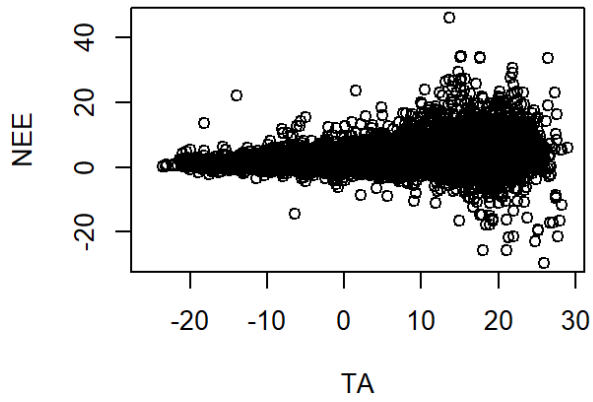
```
## related to predictive microbiolgy have been moved to the package 'nlsMicrobio'
```

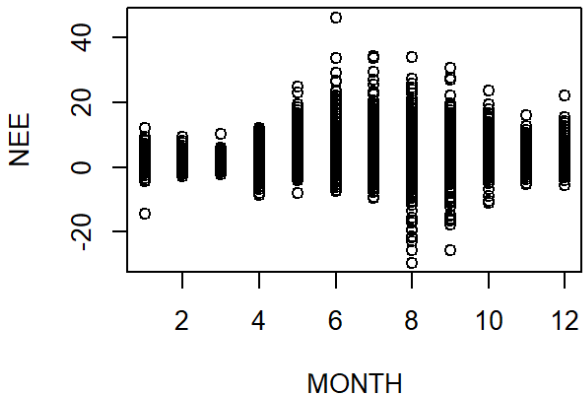
Temperature Response Curve

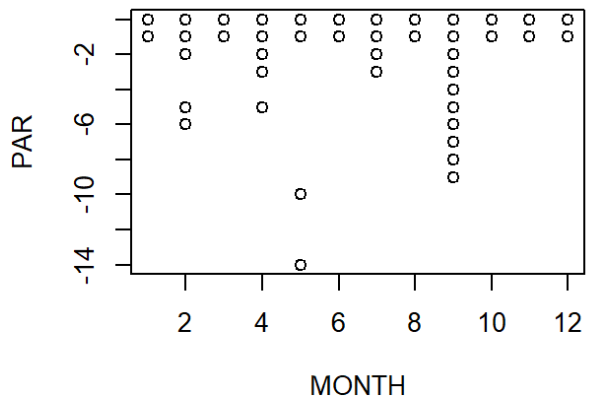
```
plot( NEE ~ TA, data= night, xlab="Temperature")
```

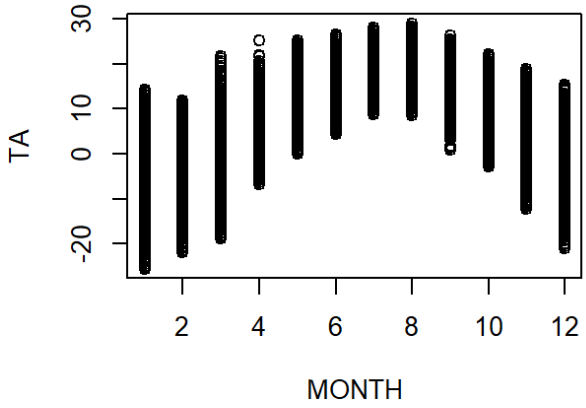



```
par(mai=c(1,1,0.1,0.1))
par(mfrow=c(2,2))
plot( NEE ~ TA, data= night)
plot( NEE ~ MONTH, data= night)
plot( PAR ~ MONTH, data= night)
plot( TA ~ MONTH, data= night)
```



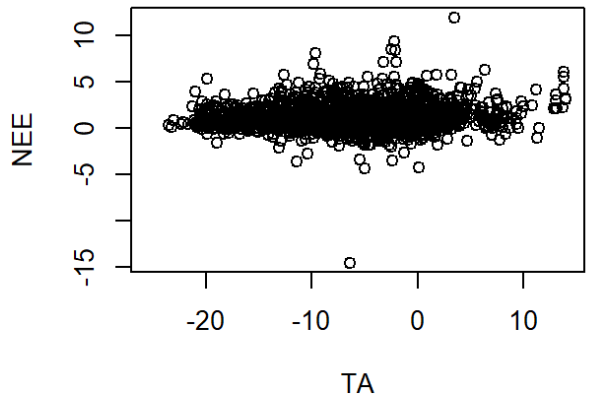


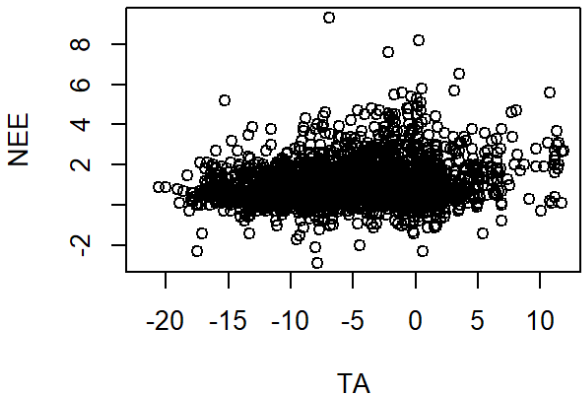


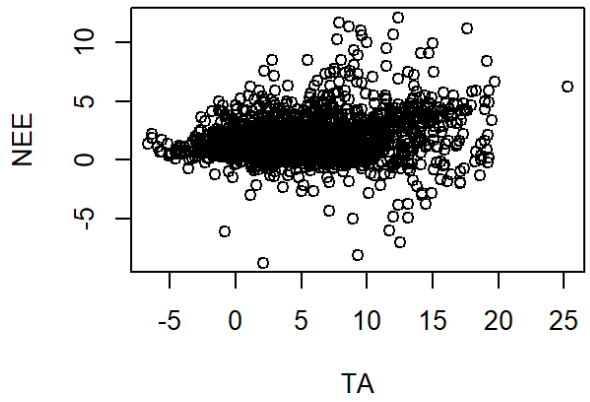


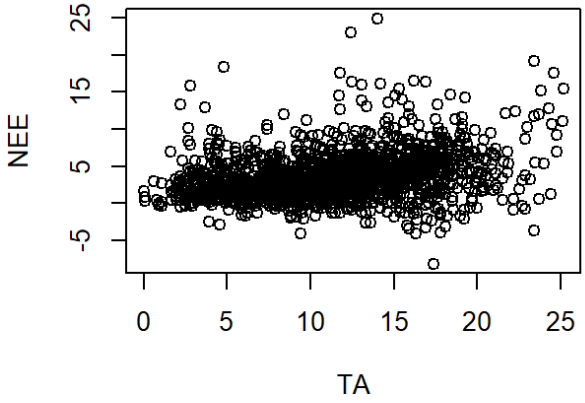

```
jan <- subset(night, MONTH == 1)
feb <- subset(night, MONTH == 2)
march <- subset(night, MONTH == 3)
april <- subset(night, MONTH == 4)
may <- subset(night, MONTH == 5)
june <- subset(night, MONTH == 6)
july <- subset(night, MONTH == 7)
aug <- subset(night, MONTH == 8)
sept <- subset(night, MONTH == 9)
oct <- subset(night, MONTH == 10)
nov <- subset(night, MONTH == 11)
dec <- subset(night, MONTH == 12)

plot( NEE ~ TA, data= jan)
plot( NEE ~ TA, data= feb)
plot( NEE ~ TA, data= april)
plot( NEE ~ TA, data= may)
```

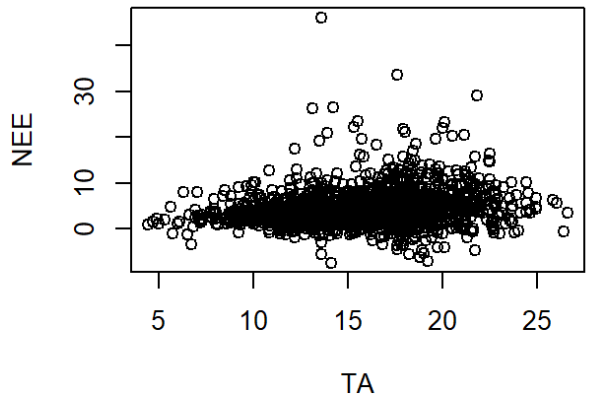


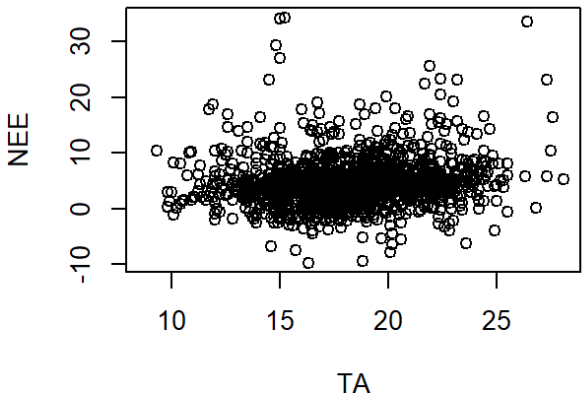


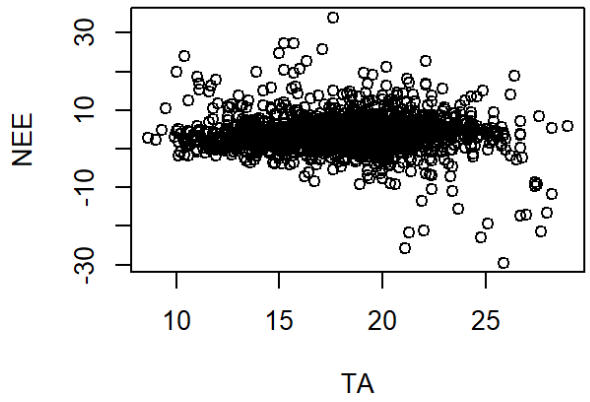


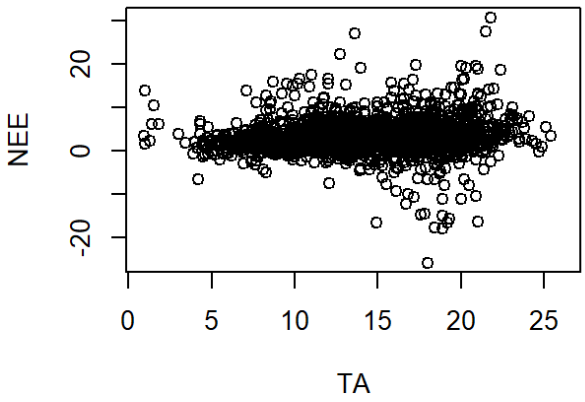



```
plot( NEE ~ TA, data= june)
plot( NEE ~ TA, data= july)
plot( NEE ~ TA, data= aug)
plot( NEE ~ TA, data= sept)
```

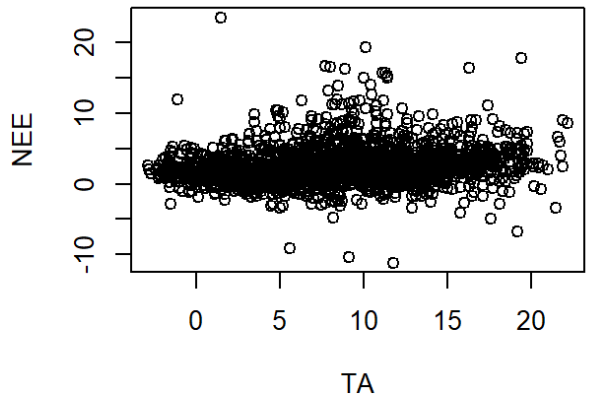


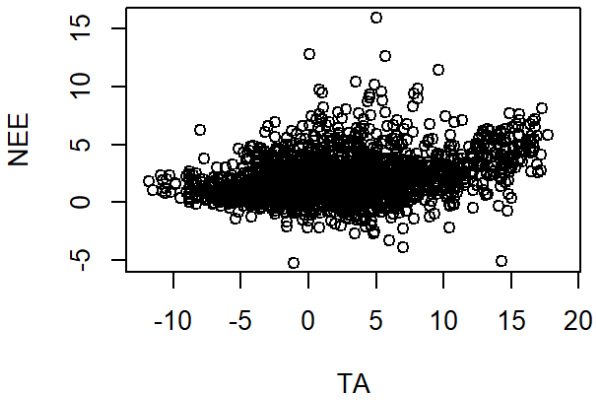


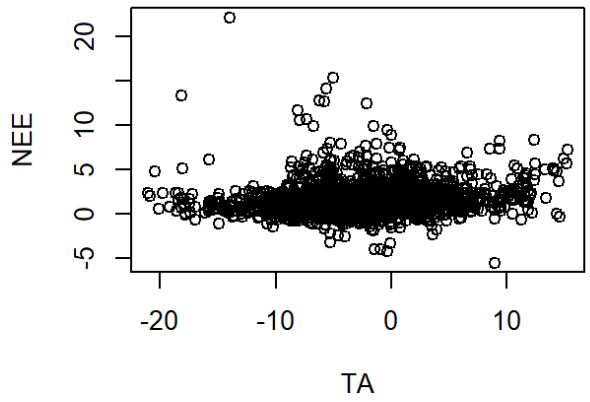





```
plot( NEE ~ TA, data= oct)
plot( NEE ~ TA, data= nov)
plot( NEE ~ TA, data= dec)
```








```
##STARTING VALUES FOR NONLINEAR FUNCTIONS

# 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)
```

```
#Dataframe to store parms and se
# Create Dataframe to store the data:
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: Use Intial Values in the model
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):
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
```

```
parms.Month
```

	MONTH <dbl>	a <dbl>	b <dbl>	a.pvalue <dbl>	b.pvalue <dbl>
1	1	1.282263	0.02790180	1.525355e-220	3.018140e-13
2	2	1.235765	0.03136320	3.371051e-312	1.048487e-17
3	3	1.100977	0.03822106	0.000000e+00	3.384407e-36
4	4	1.270271	0.04778224	9.364368e-119	2.004516e-28
5	5	1.755597	0.05711204	5.909274e-83	1.960328e-57
6	6	2.400125	0.03898796	5.739335e-21	1.561758e-10
7	7	2.005208	0.04542368	2.136098e-11	4.612002e-09
8	8	4.798788	-0.01422836	4.284396e-10	1.053243e-01
9	9	1.821047	0.03695793	3.277656e-23	2.662523e-09
10	10	1.679584	0.03924262	1.563411e-84	7.956971e-20

1-10 of 12 rows Previous 1 2 Next

```
# Create file to store parms and se
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(night.fit, a, b, c, results, y1)
}
```

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

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

trc
```

M… <dbl>	a <dbl>	b <dbl>	a.pvalue <dbl>	b.pvalue <dbl>	a.est <dbl>	b.est <dbl>	a.se <dbl>
1	1.282263	0.02790180	1.525355e-220	3.018140e-13	1.286432	0.02902050	0.04258248
2	1.235765	0.03136320	3.371051e-312	1.048487e-17	1.230251	0.03124143	0.02871379
3	1.100977	0.03822106	0.000000e+00	3.384407e-36	1.102911	0.03827917	0.03028090
4	1.270271	0.04778224	9.364368e-119	2.004516e-28	1.268106	0.04771474	0.05735067
5	1.755597	0.05711204	5.909274e-83	1.960328e-57	1.764739	0.05691674	0.10742690
6	2.400125	0.03898796	5.739335e-21	1.561758e-10	2.443683	0.03853749	0.30662514
7	2.005208	0.04542368	2.136098e-11	4.612002e-09	1.968304	0.04677943	0.33397457
8	4.798788	-0.01422836	4.284396e-10	1.053243e-01	4.699105	-0.01258482	0.74589251
9	1.821047	0.03695793	3.277656e-23	2.662523e-09	1.840553	0.03669647	0.21417726
10	1.679584	0.03924262	1.563411e-84	7.956971e-20	1.674778	0.03965920	0.10174779