Nonlinear_Assignment2_Light_Curve

Meredith Emery

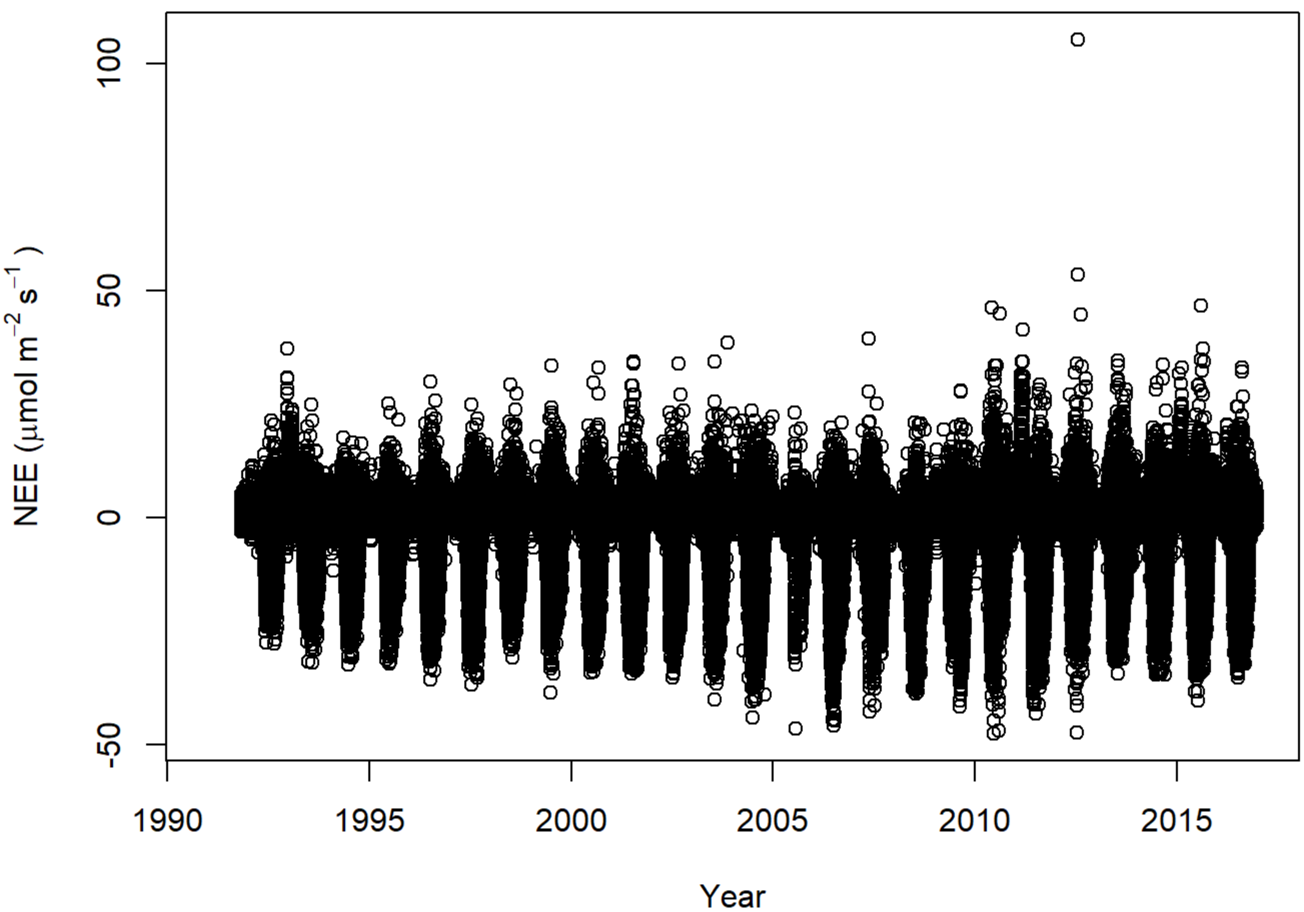
```
load("C:/Users/Mere/Desktop/FIU Courses/Spring 2020/Quantative Ecology_WS/Nonlinear/NLM_Worksho
p.RData")
library(nlstools)
```

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

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

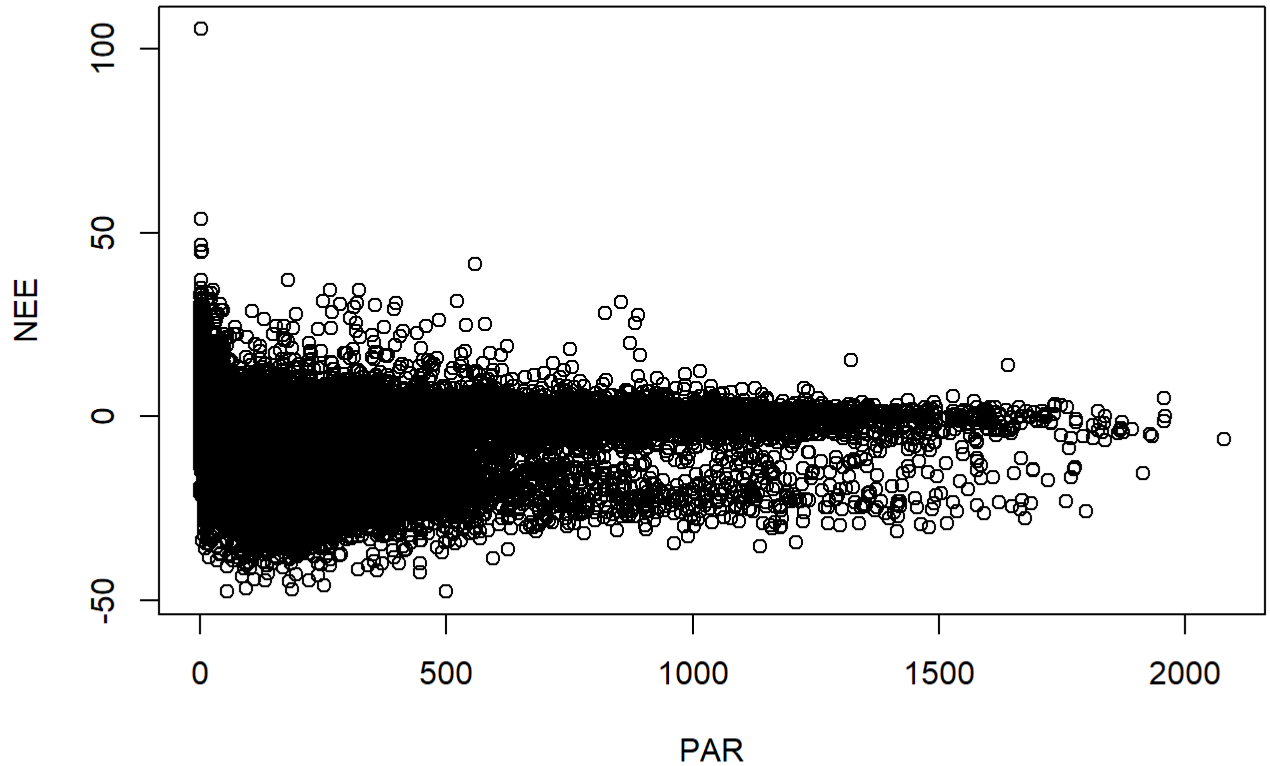
```
## related to predictive microbiolgy have been moved to the package 'nlsMicrobio'
```

```
par(mai=c(1,1,0.1,0.1))
plot(harv$TIMESTAMP, harv$NEE,
     ylab=expression(paste("NEE (",mu,"mol m"^{-2} ~ s^{-1} ~ ")" )), xlab="Year")
```



Visualizing Data:

```
plot( NEE ~ PAR, data= day)
```



Fitting Light Response Curves With nls()

```
y = nls( NEE ~ (a1 * PAR * ax)/(a1 * PAR + ax) + r, data=day[which(day$MONTH == 07),],
start=list(a1= -1 , ax= -1, r= 1),
na.action=na.exclude, trace=F, control=nls.control(warnOnly=T))
```

```
## Warning in nls(NEE ~ (a1 * PAR * ax)/(a1 * PAR + ax) + r, data =
## day[which(day$MONTH == : step factor 0.000488281 reduced below 'minFactor' of
## 0.000976562
```

```
summary(y)
```

```
##
## Formula: NEE ~ (a1 * PAR * ax)/(a1 * PAR + ax) + r
##
## Parameters:
##    Estimate Std. Error t value Pr(>|t|)
## a1  19428.2   152692.4   0.127    0.899
## ax    199.4      763.6   0.261    0.794
## r    -208.7      763.7  -0.273    0.785
##
## Residual standard error: 12.54 on 6656 degrees of freedom
##
## Number of iterations till stop: 13
## Achieved convergence tolerance: 0.8081
## Reason stopped: step factor 0.000488281 reduced below 'minFactor' of 0.000976563
##   (1760 observations deleted due to missingness)
```

Starting Values for Nonlinear Models:

```
# 1. Create a function of the model:

lrcModel <- function(PAR, a1, ax, r) {
NEE <- (a1 * PAR * ax)/(a1 * PAR + ax) + r
return(NEE)
}

# 2. Initial: create a function that calculates the intial values from the data

lrc.int <- function (mCall, LHS, data){
x <- data$PAR
y <- data$NEE
r <- max(na.omit(y), na.rm=T) # Maximum NEE
ax <- min(na.omit(y), na.rm=T) # Minimum NEE
a1 <- (r + ax)/2 # Midway between r and a1

# Create limits for the parameters:
a1[a1 > 0]<- -0.1
r[r > 50] <- ax*-1
r[r < 0] <- 1
value = list(a1, ax, r) # Must include this for the selfStart function
names(value) <- mCall[c("a1", "ax", "r")] # Must include this for the selfStart function
return(value)
}
```

Use the selfStart function to calculate initial values:

```
# Selfstart function
SS.lrc <- selfStart(model=lrcModel,initial= lrc.int)
# 3. Find initial values:
iv <- getInitial(NEE ~ SS.lrc('PAR', "a1", "ax", "r"),
data = day[which(day$MONTH == 07),])

iv #Use initial values in the model
```

```
## $a1
## [1] -0.1
##
## $ax
## [1] -47.44
##
## $r
## [1] 47.44
```

Model with intial values in the model:

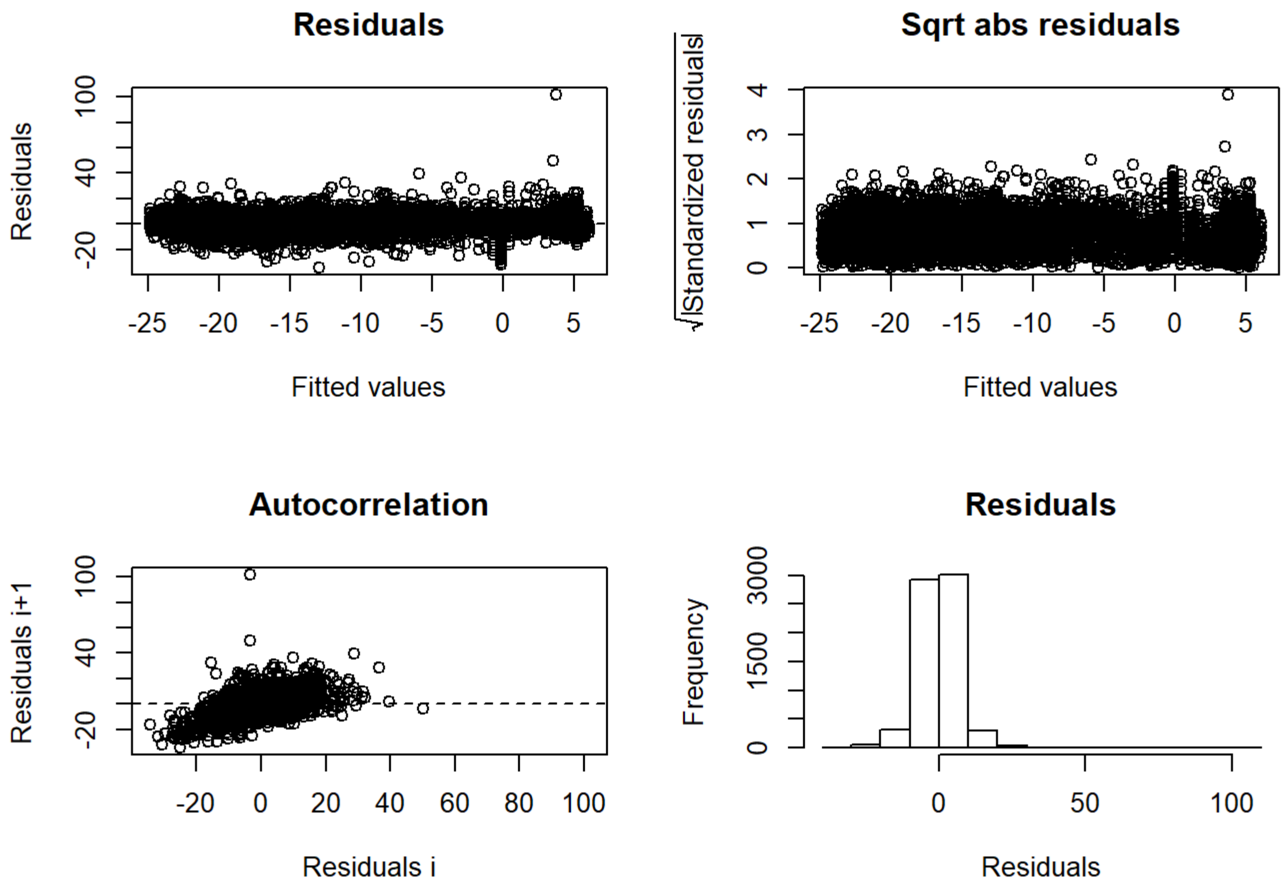
```
y = nls( NEE ~ (a1 * PAR * ax)/(a1 * PAR + ax) + r, day[which(day$MONTH == 07),],
start=list(a1= iv$a1 , ax= iv$ax, r= iv$r),
na.action=na.exclude, trace=F, control=nls.control(warnOnly=T))

summary(y)
```

```
##
## Formula: NEE ~ (a1 * PAR * ax)/(a1 * PAR + ax) + r
##
## Parameters:
##    Estimate Std. Error t value Pr(>|t|)
## a1  -0.8732     0.0289  -30.21   <2e-16 ***
## ax -31.7395     0.2828 -112.25   <2e-16 ***
## r    6.1558     0.1878   32.79   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.728 on 6656 degrees of freedom
##
## Number of iterations to convergence: 6
## Achieved convergence tolerance: 4.108e-06
##   (1760 observations deleted due to missingness)
```

Model converged. Now, let check assumptions

```
res.lrc <- nlsResiduals(y)
par(mfrow=c(2,2))
plot(res.lrc, which=1)# Residulas vs fitted values (Constant Variance)
plot(res.lrc, which=3) # Standardized residuals
plot(res.lrc, which=4) # Autocorrelation
plot(res.lrc, which=5) # Histogram (Normality)
```



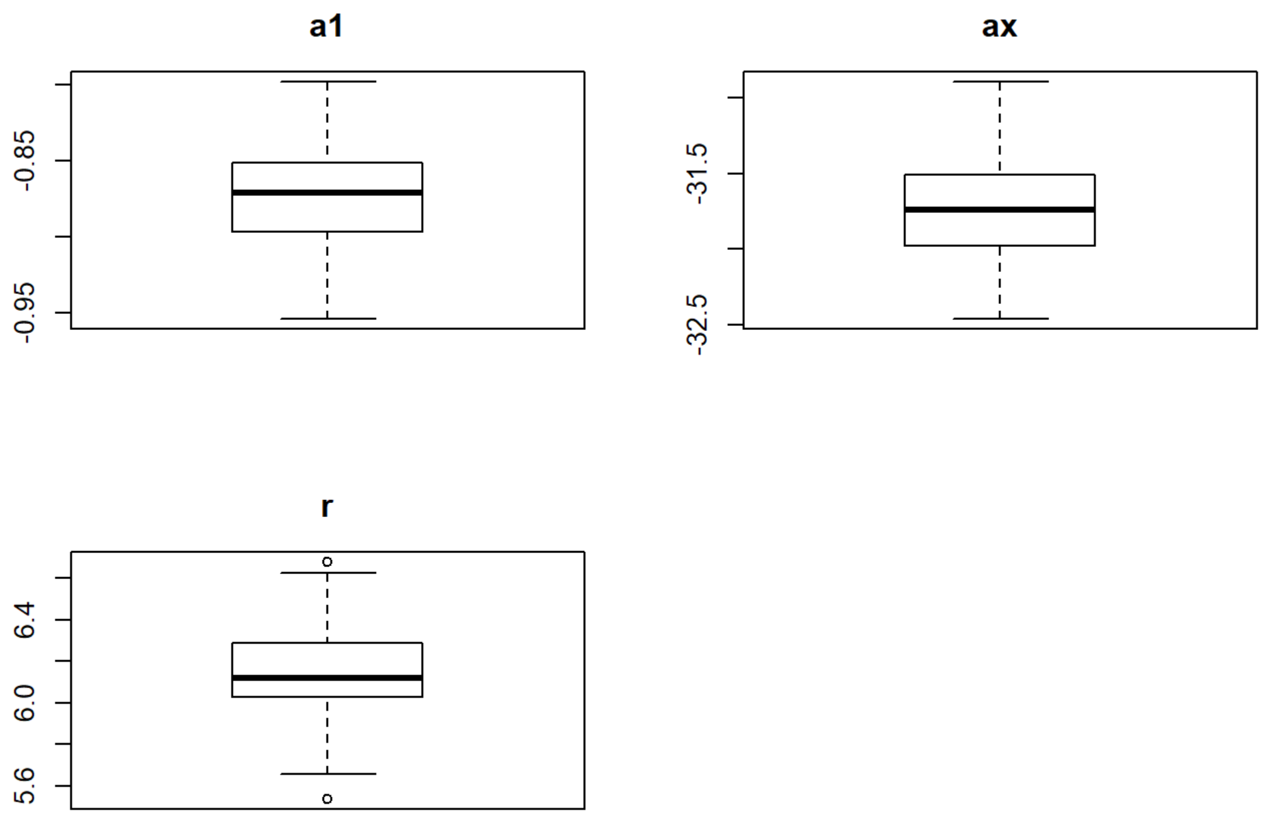
Bootstrap

```
results <- nlsBoot(y, niter=100 )

summary(results)
```

```
##
## ------
## Bootstrap statistics
##       Estimate Std. error
## a1  -0.8735624 0.03358879
## ax -31.7217969 0.31487794
## r    6.1502644 0.19791938
##
## ------
## Median of bootstrap estimates and percentile confidence intervals
##         Median        2.5%      97.5%
## a1  -0.8711073  -0.9382934  -0.810862
## ax -31.7368073 -32.3059875 -31.096280
## r    6.1194547   5.8335563   6.518513
```

```
plot(results, type = "boxplot")
```



Exercise: How variable a NEE rates over an annual cycle in Harvard Forest?

1. Create a dataframe to store a month parameter values (parms.Month)

```
# Dataframe to store parms and se

parms.Month <- data.frame(
MONTH=numeric(),
a1=numeric(),
ax=numeric(),
r=numeric(),
a1.pvalue=numeric(),
ax.pvalue=numeric(),
r.pvalue=numeric(), stringsAsFactors=FALSE, row.names=NULL)

parms.Month[1:12, 1] <- seq(1,12,1) # Adds months to the file
```

2. Write a function to fit the model and extract parameters (nee.day)

```r
nee.day <- function(dataframe){ y = nls( NEE ~ (a1 * PAR * ax)/(a1 * PAR + ax) + r, dataframe,
start=list(a1= iv$a1 , ax= iv$ax, r= iv$r),
na.action=na.exclude, trace=F,
control=nls.control(warnOnly=T))

y.df <- as.data.frame(cbind(t(coef(summary(y)) [1:3, 1]), t(coef(summary(y)) [1:3, 4])))
names(y.df) <-c("a1","ax", "r", "a1.pvalue", "ax.pvalue", "r.pvalue")
return (y.df )}
```

3. Write a loop to fit monthly curves and add parameters to a dataframe (parms.Month)

```r
try(for(j in unique(day$MONTH)){

# Determines starting values:
iv <- getInitial(NEE ~ SS.lrc('PAR', "a1", "ax", "r"), data = day[which(day$MONTH == j),])

# Fits light response curve:
y3 <- try(nee.day(day[which(day$MONTH == j),]), silent=T)

# Extracts data and saves it in the dataframe
try(parms.Month[c(parms.Month$MONTH == j ), 2:7 ] <- cbind(y3), silent=T)

rm(y3)
}, silent=T)

parms.Month
```

	M... <dbl>	a1 <dbl>	ax <dbl>	r <dbl>	a1.pvalue <dbl>	ax.pvalue <dbl>	r.pval <db
1	1	-4.081421e-05	4.464131e-03	1.470930	3.305954e-02	3.326488e-02	0.000000e+
2	2	8.348064e-03	1.012233e+00	1.253452	3.297912e-02	2.148410e-09	2.682011e-
3	3	7.591131e-04	9.939385e-02	1.296303	7.835446e-01	3.520764e-01	2.762107e-1
4	4	-3.785140e+00	1.010519e+00	-0.457579	7.239583e-14	3.005470e-16	4.428395e-
5	5	-1.720700e-01	-8.108276e+00	3.687535	1.568976e-33	0.000000e+00	2.133334e-1
6	6	-8.245779e-01	-2.751034e+01	5.855905	7.538870e-151	0.000000e+00	7.147670e-1
7	7	-8.732107e-01	-3.173946e+01	6.155766	5.182107e-188	0.000000e+00	1.061783e-2
8	8	-7.112455e-01	-3.105874e+01	5.739692	1.063182e-239	0.000000e+00	6.777948e-2
9	9	-8.034940e-01	-2.508709e+01	4.880482	2.360522e-201	0.000000e+00	4.037587e-2
10	10	-4.999720e-01	-7.923081e+00	3.146178	1.270325e-25	0.000000e+00	2.038746e-1

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

4. Bootstrapping

```r
# Create file to store parms and se
boot.NEE <- data.frame(parms.Month[, c("MONTH")]); names (boot.NEE) <- "MONTH"
boot.NEE$a1.est <- 0
boot.NEE$ax.est<- 0
boot.NEE$r.est<- 0
boot.NEE$a1.se<- 0
boot.NEE$ax.se<- 0
boot.NEE$r.se<- 0

for ( j in unique(boot.NEE$Month)){

y1 <-day[which(day$MONTH == j),] # Subsets data

# Determines the starting values:
iv <- getInitial(NEE ~ SS.lrc('PAR', "a1", "ax", "r"), data = y1)

# Fit curve:
day.fit <- nls( NEE ~ (a1 * PAR * ax)/(a1 * PAR + ax) + r, data=y1,
start=list(a1= iv$a1 , ax= iv$ax, r= iv$r),
na.action=na.exclude, trace=F, control=nls.control(warnOnly=T))

# Bootstrap and extract values:
try(results <- nlsBoot(day.fit, niter=100 ), silent=T)
try(a <- t(results$estiboot)[1, 1:3], silent=T)
try(names(a) <- c('a1.est', 'ax.est', 'r.est'), silent=T)
try( b <- t(results$estiboot)[2, 1:3], silent=T)
try(names(b) <- c('a1.se', 'ax.se', 'r.se'), silent=T)
try(c <- t(data.frame(c(a,b))), silent=T)

# Add bootstrap data to dataframe:
try(boot.NEE[c(boot.NEE$MONTH == j), 2:7] <- c[1, 1:6], silent=T)
try(rm(day.fit, a, b, c, results, y1), silent=T)
}

lrc <- merge( parms.Month, boot.NEE, by.x="MONTH", by.y="MONTH") # Merge dataframes

lrc
```

M… <dbl>	a1 <dbl>	ax <dbl>	r <dbl>	a1.pvalue <dbl>	ax.pvalue <dbl>	r.pvalue <dbl>
1	-4.081421e-05	4.464131e-03	1.470930	3.305954e-02	3.326488e-02	0.000000e+00
2	8.348064e-03	1.012233e+00	1.253452	3.297912e-02	2.148410e-09	2.682011e-90
3	7.591131e-04	9.939385e-02	1.296303	7.835446e-01	3.520764e-01	2.762107e-129
4	-3.785140e+00	1.010519e+00	-0.457579	7.239583e-14	3.005470e-16	4.428395e-04
5	-1.720700e-01	-8.108276e+00	3.687535	1.568976e-33	0.000000e+00	2.133334e-165
6	-8.245779e-01	-2.751034e+01	5.855905	7.538870e-151	0.000000e+00	7.147670e-180
7	-8.732107e-01	-3.173946e+01	6.155766	5.182107e-188	0.000000e+00	1.061783e-218

M…	a1	ax	r	a1.pvalue	ax.pvalue	r.pvalue
<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
8	-7.112455e-01	-3.105874e+01	5.739692	1.063182e-239	0.000000e+00	6.777948e-271
9	-8.034940e-01	-2.508709e+01	4.880482	2.360522e-201	0.000000e+00	4.037587e-297
10	-4.999720e-01	-7.923081e+00	3.146178	1.270325e-25	0.000000e+00	2.038746e-132

1-10 of 12 rows | 1-8 of 13 columns

Previous **1** 2 Next