Assignment 2

Ghatta Trivedi

2025-02-07

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

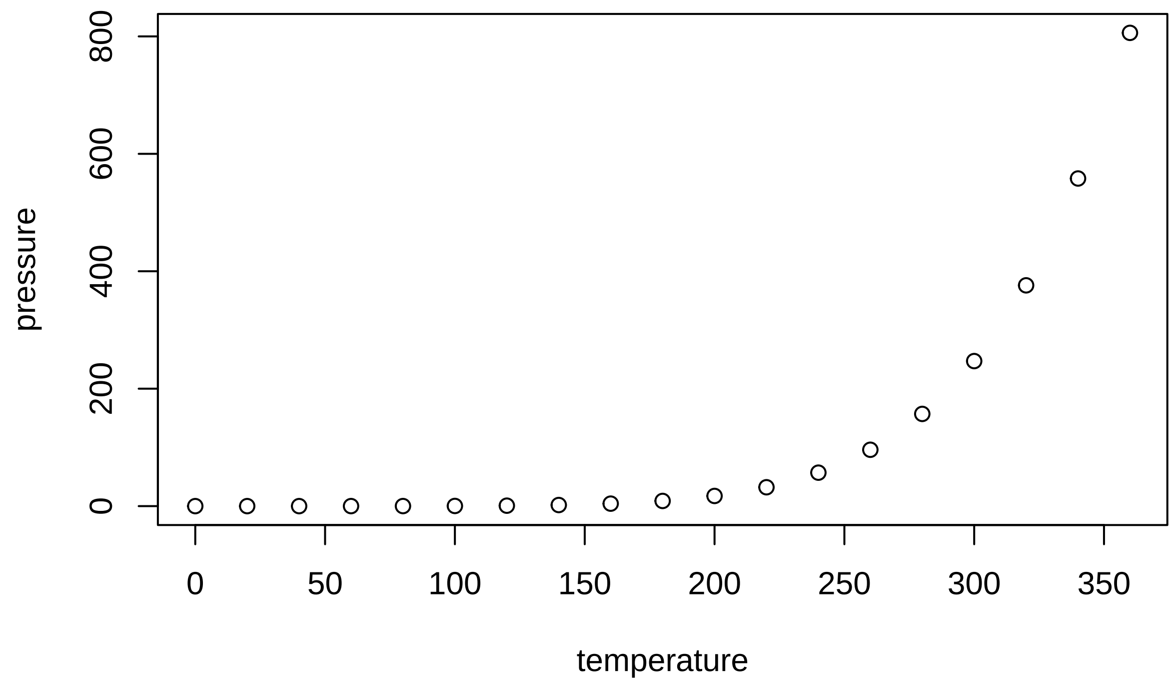
When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```r
summary(cars)
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00  
```

Including Plots

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

```
source("C:\\Users\\trive\\Documents\\Applied Stat Learning R\\codeday6.pck")
codeday6.pck
```

```
## [1] "codeday6.pck"    "my.bootstrap1"   "gui.bootstrap1"  "my.bootstrap2"
## [5] "gui.bootstrap2"  "my.bootstrap3"   "gui.bootstrapxy" "my.dat.plot5"
## [9] "my.dat.plot5a"
```

```
NOAA <- read.csv("C:\\Users\\trive\\Documents\\Applied Stat Learning R\\NOAA+GISS 2024.csv")
NOAA
```

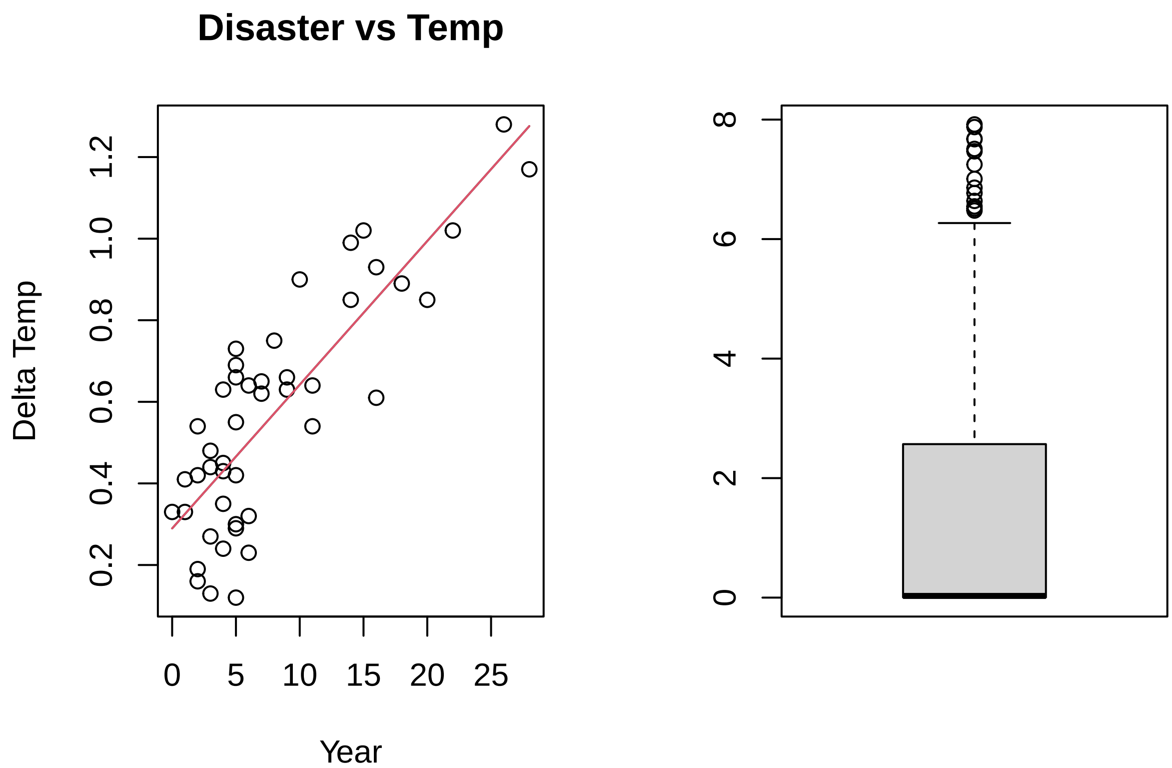
```
##    year X.disaster delta.temp
## 1  1980          3       0.27
## 2  1981          1       0.33
## 3  1982          3       0.13
## 4  1983          5       0.30
## 5  1984          2       0.16
## 6  1985          5       0.12
## 7  1986          2       0.19
## 8  1987          0       0.33
## 9  1988          1       0.41
## 10 1989          5       0.29
## 11 1990          3       0.44
```

```
## 12 1991          4       0.43
## 13 1992          6       0.23
## 14 1993          4       0.24
## 15 1994          6       0.32
## 16 1995          4       0.45
## 17 1996          4       0.35
## 18 1997          3       0.48
## 19 1998          9       0.63
## 20 1999          5       0.42
## 21 2000          2       0.42
## 22 2001          2       0.54
## 23 2002          4       0.63
## 24 2003          7       0.62
## 25 2004          5       0.55
## 26 2005          5       0.69
## 27 2006          6       0.64
## 28 2007          5       0.66
## 29 2008         11       0.54
## 30 2009          7       0.65
## 31 2010          5       0.73
## 32 2011         16       0.61
## 33 2012         11       0.64
## 34 2013          9       0.66
## 35 2014          8       0.75
## 36 2015         10       0.90
## 37 2016         15       1.02
## 38 2017         16       0.93
## 39 2018         14       0.85
## 40 2019         14       0.99
## 41 2020         22       1.02
## 42 2021         20       0.85
## 43 2022         18       0.89
## 44 2023         28       1.17
## 45 2024         26       1.28
```

```r
#commented out code
# to fullfill the first section of the assignment, we have to make a function that takes in a data fram
first <- function(mat=NOAA,i=3,j=2,zxlab,zylab,zmain,zcol,do.sqrt=F,nboot=10000){
#function(mat=NOAA,i=3,j=2,zxlab,zylab,zmain,zcol,do.sqrt=F,do.plot=T,in.boot=T)
#stat.out<-list(smstrmod=smstr,smstrlin=smstr.lin,resid.mod=resid1,resid.lin=resid2,dfmod=dfmod,dflin=d

  #first we need to set up the plot area to show plots side by side
  par(mfrow=c(1,2))

  #this line calls for the function my.dat.plot5, which will: plot the data with a smoothing spline adn
  # the argument do.plot = T means it will show the plot and in.boot = F means it's not in the bootstra
  stat.out0<-my.dat.plot5(mat,i,j,zxlab,zylab,zmain,zcol,do.sqrt,do.plot=T,in.boot=F)

  #now, we need to store results
  # this one stores the fstat
  F0<-stat.out0$F

  #gets the residual from the linear fit
  resid0<-stat.out0$resid.lin
```

```
  #initialize a vecotre to store f stats from the bootstrapped fits
  bootvec<-NULL

  #get predicted values from the linear model using the first column of the data
  y0<-predict(stat.out0$smstrlin,mat[,i])$y

  #next we need to create a copy of the input matrix for bootstrapping
  matb<-mat

  #now we need to loop to perform bootstrap iterations
  for(i1 in 1:nboot){

    #lets us know progress by printing 500,1000,etc
      if(floor(i1/500)==(i1/500)){print(i1)}

    #sample residuals with replacement to create a bootstrapped residual vector
      residb<-sample(resid0,replace=T)

      #add the bootstrapped residuals to the predicted values to generate new bootstrapped rensponse va
      Yb<-y0+residb

      # now we can replace the dependent variable with the bootstrapped response vals
      matb[,j]<-Yb

      #again call the my.dat.plot5 again but this time with in.boot = T so it doesn't plot anything onl
      stat.outb<-my.dat.plot5(matb,i,j,zxlab,zylab,zmain,zcol,do.sqrt,do.plot=F,in.boot=T)
      #store the f stat from the bootstrapped model fit
      bootvec<-c(bootvec,stat.outb$F)
  }

  # now, outside the loop we will calculate the pval by determining the proportion of bootstrapped f-st
  pvalboot<-sum(bootvec>F0)/nboot
  #create the boxplot of the bootstrapped f-stat
  boxplot(bootvec)
  #add the bootstrap p val to the lis tof output from the original fit
  stat.out0$pvalboot<-pvalboot
  #returns the full results, including hte original fit stat and the bootstrap model's p-val
  stat.out0
}
#basically this function is performing a model comparison between a smoothing spline and a linear fit. 

first(NOAA, "X.disaster", "delta.temp", "Year", "Delta Temp", "Disaster vs Temp", 1, do.sqrt = FALSE, n
```

```
## [1] 500
## [1] 1000
```



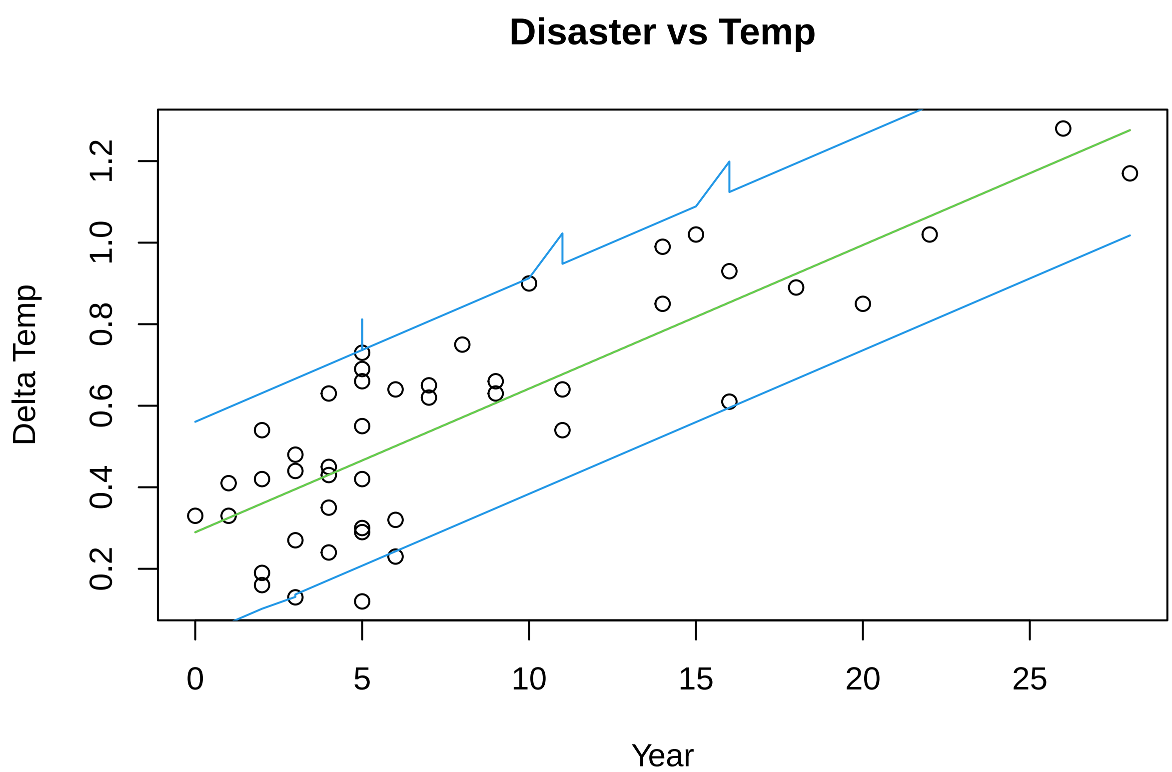

```
## $smstrmod
## Call:
## smooth.spline(x = mat[, i], y = mat[, j])
##
## Smoothing Parameter  spar= 1.493422  lambda= 7188.814 (26 iterations)
## Equivalent Degrees of Freedom (Df): 2.000011
## Penalized Criterion (RSS): 0.3294866
## GCV: 0.02673542
##
## $smstrlin
## Call:
## smooth.spline(x = mat[, i], y = mat[, j], df = 2)
##
## Smoothing Parameter  spar= 1.49992  lambda= 8007.977 (32 iterations)
## Equivalent Degrees of Freedom (Df): 2.00001
## Penalized Criterion (RSS): 0.3294866
## GCV: 0.02673542
##
## $resid.mod
##  [1] -0.1253104918  0.0051411772 -0.2653104918 -0.1657621628 -0.2000846568
##  [6] -0.3457621628 -0.1700846568  0.0403670109  0.0851411772 -0.1757621628
## [11]  0.0446895082 -0.0005363276 -0.2709879954 -0.1905363276 -0.1809879954
## [16]  0.0194636724 -0.0805363276  0.0846895082  0.0233345467 -0.0457621628
## [21]  0.0599153432  0.1799153432  0.1994636724  0.0837861766  0.0842378372
## [26]  0.2242378372  0.1390120046  0.1942378372 -0.1371170364  0.1137861766
## [31]  0.2642378372 -0.2432457759 -0.0371170364  0.0533345467  0.1785603564
```

```
## [36]  0.2581087489  0.2019799475  0.0767542241  0.0672056823  0.2072056823
## [41] -0.0445999761 -0.1441485930 -0.0336971964 -0.1059540797  0.0744972827
##
## $resid.lin
##  [1] -0.1253105089  0.0051411456 -0.2653105089 -0.1657621652 -0.2000846812
##  [6] -0.3457621652 -0.1700846812  0.0403669721  0.0851411456 -0.1757621652
## [11]  0.0446894911 -0.0005363373 -0.2709879908 -0.1905363373 -0.1809879908
## [16]  0.0194636627 -0.0805363373  0.0846894911  0.0233345685 -0.0457621652
## [21]  0.0599153188  0.1799153188  0.1994636627  0.0837861878  0.0842378348
## [26]  0.2242378348  0.1390120092  0.1942378348 -0.1371170089  0.1137861878
## [31]  0.2642378348 -0.2432457564 -0.0371170089  0.0533345685  0.1785603735
## [36]  0.2581087741  0.2019799710  0.0767542436  0.0672057088  0.2072057088
## [41] -0.0445999956 -0.1441485979 -0.0336971879 -0.1059541481  0.0744972311
##
## $dfmod
## [1] 2.000011
##
## $dflin
## [1] 2
##
## $F
## [1] 0.1733031
##
## $P
## [1] 7.32035e-05
##
## $n
## [1] 45
##
## $pvalboot
## [1] 0.42
```

```
my.bootstrap2 <-function(mat=NOAA,i=3,j=2,zxlab,zylab,zmain,zcol,do.sqrt=F,nboot=10000,pred.bound=T,con
  #function(mat=NOAA,i=3,j=2,zxlab,zylab,zmain,zcol,do.sqrt=F,do.plot=T,in.boot=T)
  #stat.out<-list(smstrmod=smstr,smstrlin=smstr.lin,resid.mod=resid1,resid.lin=resid2,dfmod=dfmod,dflin

  # again, we need to set up the plot so:
  par(mfrow=c(1,1))

  #Calling the my.dat.plot5 to plot the data with a smoothing spline adn linear fit and to get the resi
  stat.out0<-my.dat.plot5(mat,i,j,zxlab,zylab,zmain,zcol,do.sqrt,do.plot=T,in.boot=F)

  #now, we Save residuals
  resid0<-stat.out0$resid.mod

  #Create an empty and initialize it with 0s matrix to save results of bootstrapped data
  bootmat<-NULL

  # get the predicted values from the non-linear model
  y0<-predict(stat.out0$smstrmod,mat[,i])$y

  #create a copy of the original dataset for bootstrapping
  matb<-mat
```

```r
# in this for loop we will perform the bootstrapping by resampling residuals with replacement
#so we have make this loop fo 10000 times
for(i1 in 1:nboot){
  #print progress every 500 iterations
    if(floor(i1/500)==(i1/500)){print(i1)}

  #sample residuals with replacemnt
    residb<-sample(resid0,replace=T)

    #generate bootstrapped results by adding the bootstrapped residuals
    Yb<-y0+residb

    #Replace the dependent variable in matb with the bootstrapped values
    matb[,j]<-Yb
  #print(matb)

    #call the my.dat.plot5 to fit the model on the boostrapped data without plotting
    stat.outb<-my.dat.plot5(matb,i,j,zxlab,zylab,zmain,zcol,do.sqrt,do.plot=F,in.boot=T)

    #extract residuals and predicted values from the bootstrapped model fit
    Ybp<-predict(stat.outb$smstrmod,matb[,i])$y

    # construct confidencce intervals based on the choice of 'pivotal' and 'pred.bound'
    if(pred.bound){
        if(pivotal){
            bootmat<-rbind(bootmat,stat.outb$resid.mod+Ybp-y0)
        }else{
            bootmat<-rbind(bootmat,stat.outb$resid.mod+Ybp)
        }
    }else{
        if(pivotal){
            bootmat<-rbind(bootmat,Ybp-y0)
        }else{
            bootmat<-rbind(bootmat,Ybp)
        }
    }

}
#calculate confidence intervals for the bootstrapped data
alpha<-(1-conf.lev)/2
my.quant<-function(x,a=alpha){quantile(x,c(a,1-a))}
bounds<-apply(bootmat,2,my.quant)

#adjust bounds for pivotal bootstrap
if(pivotal){
    bounds[1,]<-y0-bounds[1,]
    bounds[2,]<-y0-bounds[2,]
}

#sort the data based on the x calues
x<-mat[,i]
if(do.sqrt){
    x<-sqrt(x)
```

```
  }
    o1<-order(x)

  #add confidence interval lines to the plot
  lines(x[o1],bounds[1,o1],col=zcol+2)
  lines(x[o1],bounds[2,o1],col=zcol+2)
}

my.bootstrap2(NOAA, "X.disaster", "delta.temp", "Year", "Delta Temp", "Disaster vs Temp", 2)
```

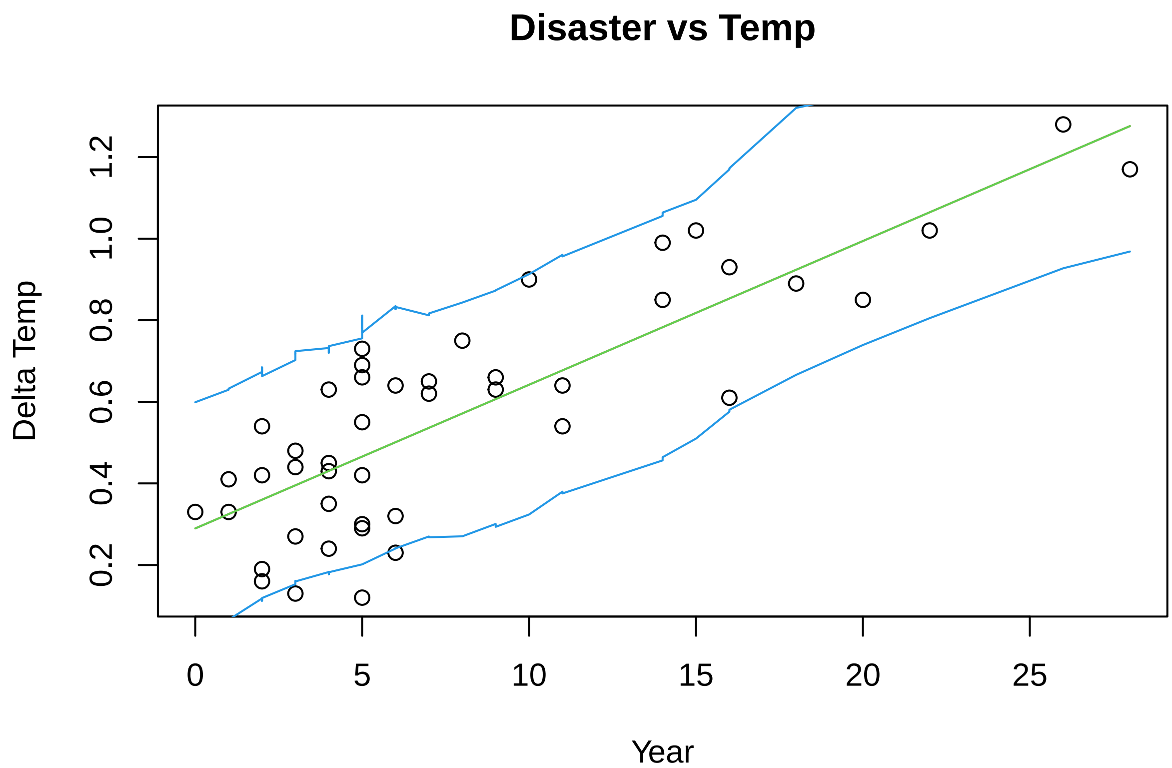



```
## [1] 500
## [1] 1000
## [1] 1500
## [1] 2000
## [1] 2500
## [1] 3000
## [1] 3500
## [1] 4000
## [1] 4500
## [1] 5000
## [1] 5500
## [1] 6000
## [1] 6500
## [1] 7000
## [1] 7500
```

```
## [1] 8000
## [1] 8500
## [1] 9000
## [1] 9500
## [1] 10000
```

```
my.bootstrap3 <-
function(mat=NOAA,i=3,j=2,zxlab,zylab,zmain,zcol,do.sqrt=F,nboot=10000,pred.bound=T,conf.lev=.95,pivota

  #set up the plot again
  par(mfrow=c(1,1))

  #call the plot function to plot data
  stat.out0<-my.dat.plot5(mat,i,j,zxlab,zylab,zmain,zcol,do.sqrt,do.plot=T,in.boot=F)

  #intialize matrix to store bootstrapped data again
  bootmat<-NULL

  #get the predicted values from the plot again
  y0<-predict(stat.out0$smstrmod,mat[,i])$y

  #create a copy of the original dataset for bootstrapping
  matb<-mat
  #get the length of the dataset
  nm<-length(matb[,1])

  #perform xy bootstrapping by sampling entire rows of the data
  for(i1 in 1:nboot){
    #checking progress again every 500 times
      if(floor(i1/500)==(i1/500)){print(i1)}

    # sample rows with replacement ie entire rows are resampled, not just residuals
      zed<-sample(nm,replace=T)
      matb<-mat[zed,]

    #print(matb)
      #fit the model with the bootstrapped data
      stat.outb<-my.dat.plot5a(matb,mat,i,j,zxlab,zylab,zmain,zcol,do.sqrt,do.plot=F,in.boot=T)

      #get predicted values from the bootstrapped model fit
      Ybp<-predict(stat.outb$smstrmod,mat[,i])$y

      #constuct confidence intervals based on the choice
      if(pred.bound){
          if(pivotal){
            #pivotal bootstrap adjusts for the baseline prediction qwhich is to subtract y0 to create c
              bootmat<-rbind(bootmat,stat.outb$resid.mod+Ybp-y0)
          }else{
            #non pivotal directly adds the residuals to the predictions
              bootmat<-rbind(bootmat,stat.outb$resid.mod+Ybp)
          }
      }else{
          if(pivotal){
            #pivotal bootstrap with no boundary would ajust residuals + predictions from bootstrap
```

```r
            bootmat<-rbind(bootmat,Ybp-y0)
        }else{
          #no boundary for non pivotal bootstrap
            bootmat<-rbind(bootmat,Ybp)
        }
    }

  }

  #calculate quantiles for the bootstrapped data
  #set the alpha level based on ci
  alpha<-(1-conf.lev)/2
  my.quant<-function(x,a=alpha){quantile(x,c(a,1-a))}
  #apply quant function to each colum of the bootstrapped data
  bounds<-apply(bootmat,2,my.quant)

  #if pivotal is used, adjust bounds to account for the baseline model prediction
  if(pivotal){
    #lower bound is adjusted by subtracting y0
      bounds[1,]<-y0-bounds[1,]
      #upper bound is also adjusted by subtracting y0
      bounds[2,]<-y0-bounds[2,]
  }
  # extract x values for plotting
  x<-mat[,i]

  if(do.sqrt){
    #apply square root transformation if required
      x<-sqrt(x)
  }

  #order the data based on the x values
  o1<-order(x)

  #add ci bounds to the plot
  lines(x[o1],bounds[1,o1],col=zcol+2)
  lines(x[o1],bounds[2,o1],col=zcol+2)

}


my.bootstrap3(NOAA, "X.disaster", "delta.temp", "Year", "Delta Temp", "Disaster vs Temp",2)
```




```
## [1] 500
## [1] 1000
## [1] 1500
## [1] 2000
## [1] 2500
## [1] 3000
## [1] 3500
## [1] 4000
## [1] 4500
## [1] 5000
## [1] 5500
## [1] 6000
## [1] 6500
## [1] 7000
## [1] 7500
## [1] 8000
## [1] 8500
## [1] 9000
## [1] 9500
## [1] 10000
```

```
# commenting on my.datplot5

my.dat.plot5 <-function(mat=NOAA,i=3,j=2,zxlab,zylab,zmain,zcol,do.sqrt=F,do.plot=T,in.boot=T){
  # if in boot is true then we need to set do sqrt to false. we do this to make sure that sqrt tranform
```

```
if(in.boot){
do.sqrt<-F
}

#if dosqrt is dalse no sqroot tranformation is applied to the dependent variable
if(!do.sqrt){

  #fit a smoothing spline model to the original dat
  smstr<-smooth.spline(mat[,i],mat[,j])
  #fit a linear smoothing spline because of df=2
  smstr.lin<-smooth.spline(mat[,i],mat[,j],df=2)

  #if do.plot is true, plot the data
if(do.plot){
  #plot the raw data
  plot(mat[,i],mat[,j],xlab=zxlab,ylab=zylab,main=zmain)
  #add line of non-linear smoothing spline
  lines(smstr,col=zcol)
  #plot the linear smoothing spline
  lines(smstr.lin,col=(zcol+1))

}
  # compute residuals for the transformed data first one is for the non-linear and the second is for
resid1<-mat[,j]-predict(smstr,mat[,i])$y
resid2<-mat[,j]-predict(smstr.lin,mat[,i])$y

}else{
#  fit a smoothing spline model to the original dat
smstr<-smooth.spline(mat[,i],sqrt(mat[,j]))
 #fit a linear smoothing spline because of df=2
smstr.lin<-smooth.spline(mat[,i],sqrt(mat[,j]),df=2)
  #if do.plot is true, plot the data
if(do.plot){
   #plot the raw data
plot(mat[,i],sqrt(mat[,j]),xlab=zxlab,ylab=zylab,main=zmain)
  #add line of non-linear smoothing spline
lines(smstr,col=zcol)
#plot the linear smoothing spline
lines(smstr.lin,col=(zcol+1))
}
 # compute residuals for the transformed data first one is for the non-linear and the second is for t
resid1<-sqrt(mat[,j])-predict(smstr,mat[,i])$y
resid2<-sqrt(mat[,j])-predict(smstr.lin,mat[,i])$y
}

#get df fpr tje non linear smoothing spline model
dfmod<-smstr$df
#df for the lin model is fixed at 2
dflin<-2

#calculate sum of squares for the  eacj of the model
ssmod<-sum(resid1^2) # which is sum of squared residuls
sslin<-sum(resid2^2)
```

```
  #compute the difference in sum of squares between the two models
  numss<-sslin-ssmod

  #get num of data points
  n1<-length(mat[,j])

  #perform an f test to compare the models
  Fstat<-(numss/(dfmod-dflin))/(ssmod/(n1-dfmod))

  #calculate the pvalue associated with the fstat
  pvalue<-1-pf(Fstat,dfmod-dflin,n1-dfmod)

  #return a list contain non linear spline model, linear spline model, residuals for each model, dfs fo
  stat.out<-list(smstrmod=smstr,smstrlin=smstr.lin,resid.mod=resid1,resid.lin=resid2,dfmod=dfmod,dflin=d
}
```