```
my.booth.smooth<-function(x.str,xindex,yindex)
{
    par(mfrow=c(2,2))
    plot(x.str[[xindex]],sqrt(x.str[[yindex]]),main="Raw data and smooths")
    lines(smooth.spline(x.str[[xindex]],sqrt(x.str[[xindex]])))
    lines(smooth.spline(x.str[[xindex]],sqrt(x.str[[yindex]]),df=2),col=2)
    smsp.strcv<-smooth.spline(x.str[[xindex]],sqrt(x.str[[yindex]]))
    smspcv.resid<-sqrt(x.str[[yindex]])-approx(smsp.strcv$x,smsp.strcv$y,x.str[[xindex]])$y
    smsp.str2<-smooth.spline(x.str[[xindex]],sqrt(x.str[[yindex]]),df=2)
    smsp2.resid<-sqrt(x.str[[yindex]])-approx(smsp.str2$x,smsp.str2$y,x.str[[xindex]])$y
    SSF<-sum(smspcv.resid^2)
    SSN<-sum(smsp2.resid^2)
    Fstat<-((SSN-SSF)/(smsp.strcv$df-2))/(SSF/(length(x.str[[xindex]])-2))
    pfstat<-1-pf(Fstat,floor((smsp.strcv$df-2)),length(x.str[[xindex]])-2)
    qqnorm(smspcv.resid,main="residuals smooth")
    qqnorm(smsp2.resid,main="residuals linear")
    list(Fstatistic=Fstat, p=pfstat, dfnum=smsp.strcv$df-2, dfden=length(x.str[[1]]-2))
}

my.smooth.forKS<-function(x.str,xindex,yindex,ind.sqrt=T) {
#par(mfrow=c(2,2))
#plot(x.str[[xindex]],sqrt(x.str[[yindex]]),main="Raw data and smooths")
#lines(smooth.spline(x.str[[xindex]],sqrt[[yindex]]))
if(ind.sqrt) { #if there is a square root
        smsp.strcv<-smooth.spline(x.str[[xindex]],sqrt(x.str[[yindex]]))
        smspcv.resid<-sqrt(x.str[[yindex]])-approx(smsp.strcv$x,smsp.strcv$y,x.str[[xindex]])$y
}
else { # no sqrt is detected
        smsp.strcv<-smooth.spline(x.str[[xindex]],(x.str[[yindex]]))
        smspcv.resid<-(x.str[[yindex]])-approx(smsp.strcv$x,smsp.strcv$y,x.str[[xindex]])$y
}
# Calculate the residuals of SD
sd.resid<-sqrt(sum(smspcv.resid^2)/(length(x.str[[1]])-smsp.strcv$df))
# Calculate the studentized residuals
```

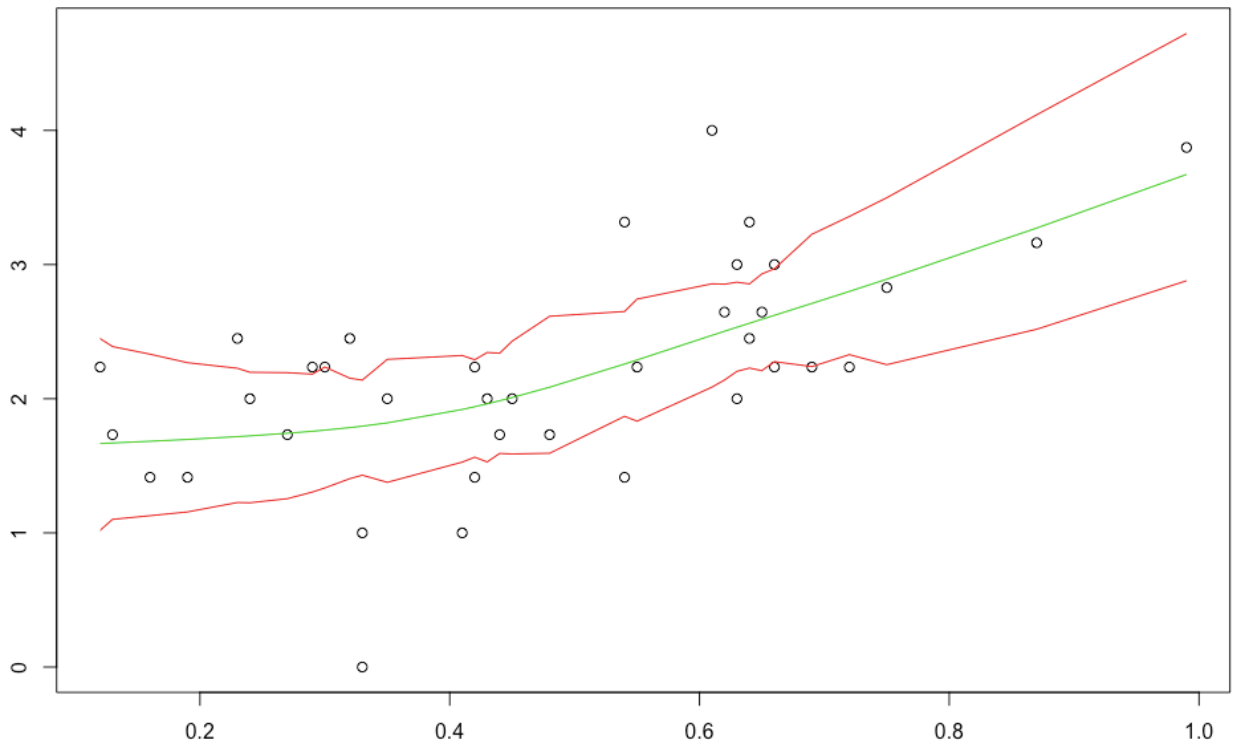
```
stud.resid<-smspcv.resid/sd.resid
# Perform Kolmogorov-Smirnov Tests
#D<-ks.test(stud.resid,pnorm)$statistic
ks.str<-ks.test(stud.resid,pnorm)
D<-ks.str$statistic
Pval<-ks.str$p.value
my.smooth<-approx(smsp.strcv$x,smsp.strcv$y,x.str[[xindex]])$y
# Display the statistics
list(D=D,raw.resid=smspcv.resid,sd.resid=sd.resid, smooth=my.smooth,P=Pval)
}

my.KS.boot.normal<-function(mydata,x.index,y.index,nboot,confidence) {
    par(mfrow=c(1,1))
    str0<-my.smooth.forKS(mydata,x.index,y.index)
       # Initialize the distance to null
    smooth.dist<-NULL
       # Initialize the base
    base.smooth<-str0$smooth
    base.sd<-str0$sd.resid
    base.resid<-str0$raw.resid
    my.bootdata<-mydata
    n1<-length(base.smooth)
    for(i in 1:nboot){
              # Generate length copies of random numbers WITH replacements
        bres<-sample(base.resid,length(base.resid),replace=T)
        boot.dat<-((base.smooth+bres))
        #print(boot.dat)
        my.bootdata[[y.index]]<-boot.dat
        bstr0<-my.smooth.forKS(my.bootdata,x.index,y.index,F)
        boot.smooth<-bstr0$smooth
              # Combines smooth.dist and boot.smooth-base.smooth
        smooth.dist<-rbind(smooth.dist,boot.smooth-base.smooth)

    }
    n1<-length(smooth.dist[1,])
       # Initialize alpha
    alpha<-1-confidence
    LB<-NULL
```

```
    UB<-NULL
    for(i in 1:n1){
        s1<-sort(smooth.dist[,i])
        n2<-length(s1)
        v1<-c(1:n2)/n2
        bvec<-approx(v1,s1,c(alpha/2,1-alpha/2))$y
              # generate lower bound
        LB<-c(LB,base.smooth[i]-bvec[2])
              # generate upper bound
        UB<-c(UB,base.smooth[i]-bvec[1])
    }
    plot(rep(mydata[[x.index]],4),c(LB,base.smooth,UB,sqrt(mydata[[y.index]])),xlab="X",ylab="Y",type="n")
    points(mydata[[x.index]],sqrt(mydata[[y.index]]))
       # sort the x.index in an ascending order
    o1<-order(mydata[[x.index]])
       # draw the lower bound line (red)
    lines(mydata[[x.index]][o1],LB[o1],col=2)
       # draw the upper bound line (red)
    lines(mydata[[x.index]][o1],UB[o1],col=2)
       # draw the smooth line between upper and lower bounds (green)
    lines(mydata[[x.index]][o1],base.smooth[o1],col=3)
}

> my.KS.boot.normal(NOAA,3,2,1000,0.95)
```


0
1
2
3
4
0.2
0.4
0.6
0.8
1.0