1

Arun Ram Sankaranarayanan

November 4, 2015

Q1

```r
day= c( 1,7,8,10,14,16,21,24,28,30,42,46,60,63,65)
wt2=c(143,-184,182,-110,1017,986,1010,1001,-111,-60,-151,-111,1024,1031,1028)



#a

fivenum(wt2)
```

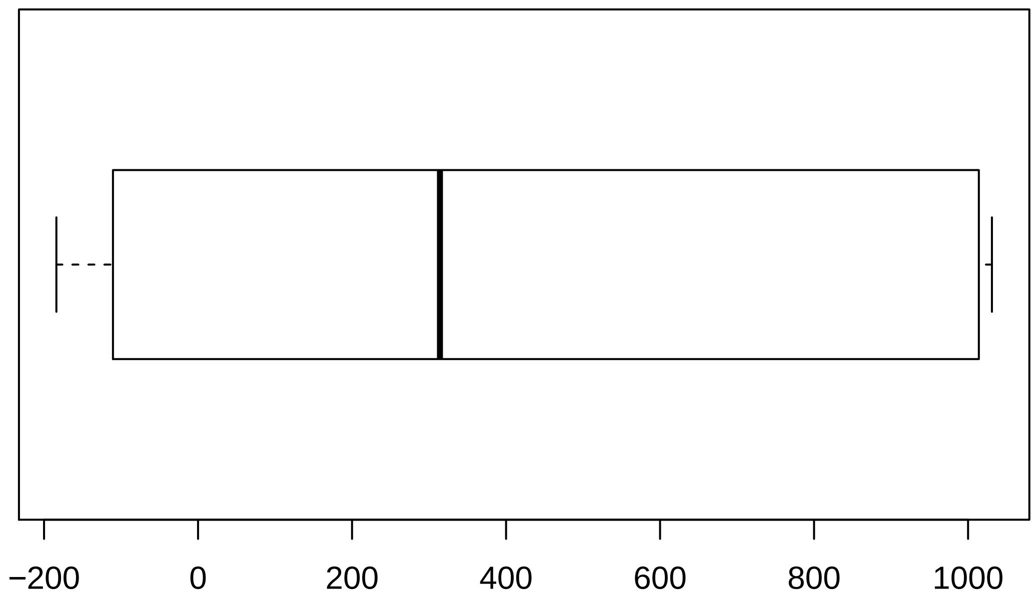
```
## [1] -184.0 -110.5  182.0 1013.5 1031.0
```

```r
k=summary(wt2)

#b
stem(k)
```

```
##
##   The decimal point is 3 digit(s) to the right of the |
##
##   -0 | 21
##   0 | 24
##   0 |
##   1 | 00
```

```r
boxplot(k,horizontal=TRUE)
```

(b)



Q2

```
n=5000


gaus= 0.4+0.0007*n

gaus
```

```
## [1] 3.9
```

```
cat("outside values=" , gaus)
```

```
## outside values= 3.9
```

Q3 Indicate which of the following would or would not need a re-expression. For those situations that do indicate re-expression, the tool you would use to identify it. [If the tool is a plot, state the quantities on the x- (horizontal) and y- (vertical) axes.]
(a)

The plot is left skewed so we have to ascend tukey's power curve so square or cubed would be a good transformation.ress large values.So we can take squares or cubes or higher powers

(b)

Transformation for spread must be done so the power is usually log or square root.

(C)

The transformation has heavy tails and centered data and hence no transformation is required.

(D) Since the data is side skewed we have to compress the data hence log or square root should be used.

Q4 A smoother is a statistic technique used to identify the patterns in data. When the smoothed values can be written as a linear transformation of the observed values, the smoothing operation is known as a linear smoother. Some of the linear smoothers are 1. Lowess Smoother 2. Running mean smoothers One major disadvantage of linear smoothers are that it overfits the data.

Non linear smoothers are quite flexible when compared to linear smoothers as they dont have any linear constraints.

Disadvantage of non linear smoothers over linear Non linear smoothers require only dense points for correct prediction

Q5

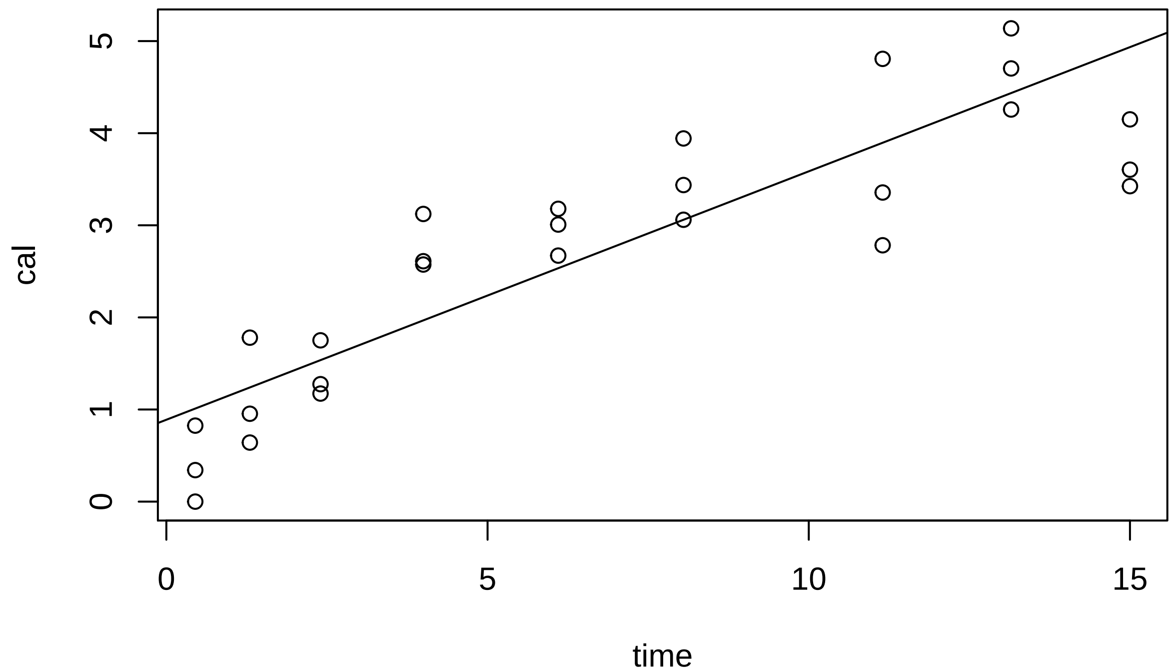
```
#a
source("rrline.r")

time = c(0.45, 0.45, 0.45, 1.30, 1.30, 1.30, 2.40,
         2.40, 2.40, 4.0, 4.0, 4.0, 6.1, 6.1, 6.1,
         8.05, 8.05, 8.05, 11.15, 11.15, 11.15, 13.15,
         13.15, 13.15, 15.0, 15.0, 15.0)

cal = c(0.342, 0.0, 0.825, 1.78, 0.954,
        0.641, 1.751, 1.275, 1.173, 3.123,
        2.61, 2.574, 3.179, 3.008, 2.671, 3.06,
        3.943, 3.437, 4.807, 3.356, 2.783, 5.138,
        4.703, 4.257, 3.604, 4.15, 3.425)

a=rrline1(time,cal)

slope=a$b

Intercept= a$a

cat("Slope=", slope)
```

```
## Slope= 0.2697046
```

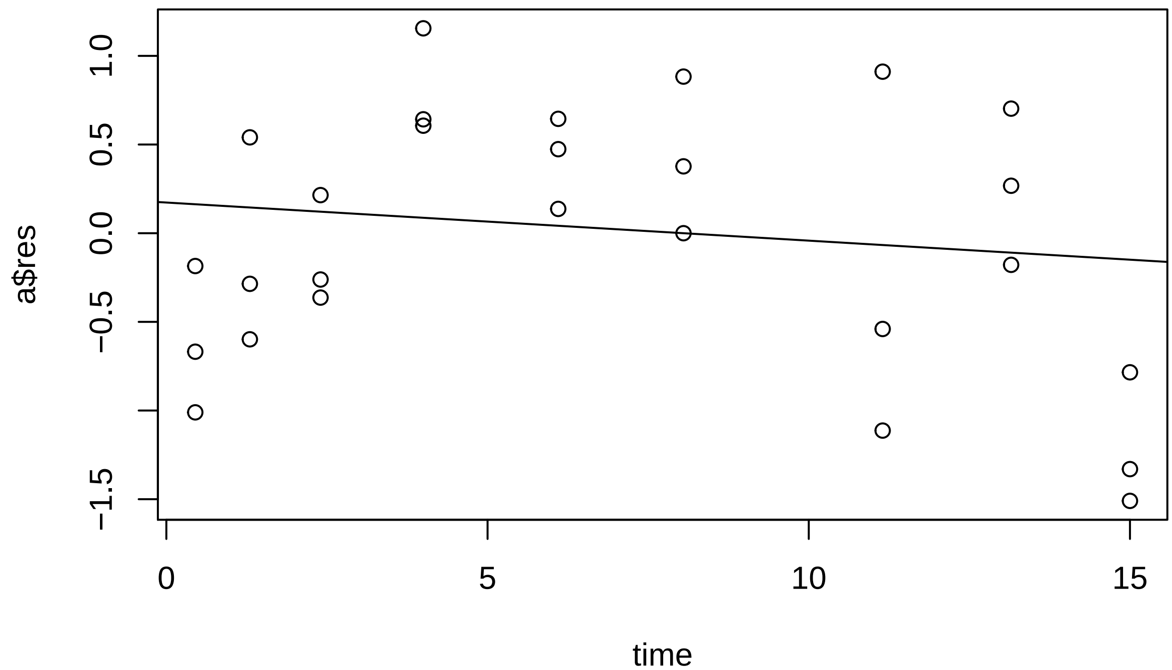
```
cat("Intercept=",Intercept )
```

```
## Intercept= 0.8888776
```

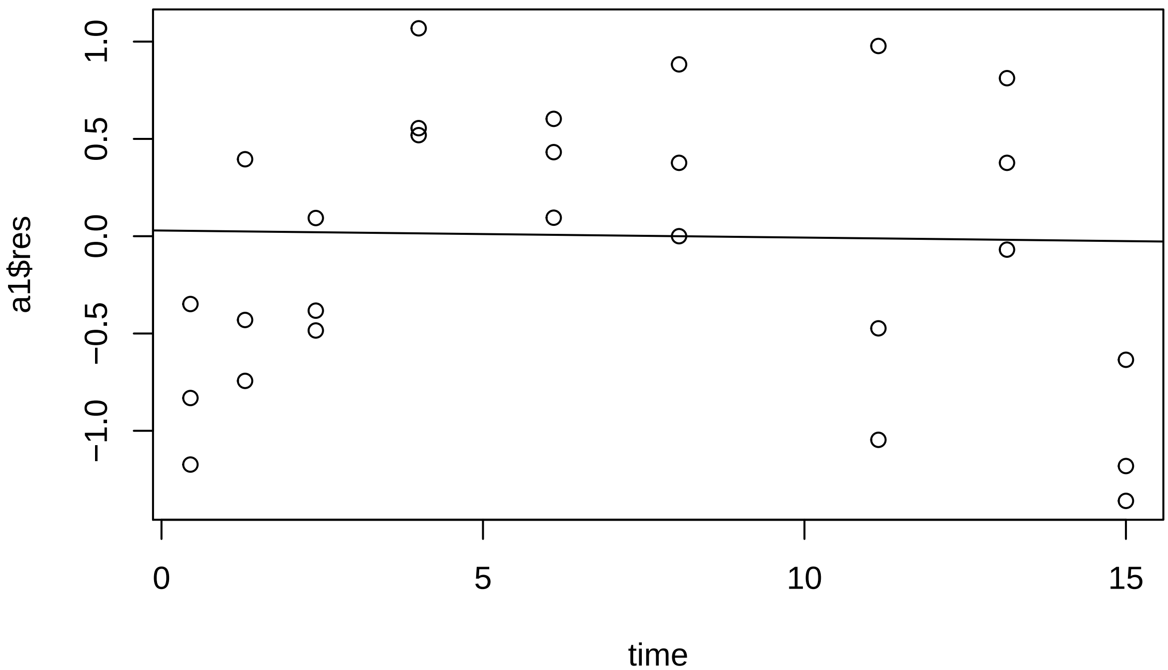
```
plot.new()

#b
plot(time,cal)
abline(Intercept,slope)
```



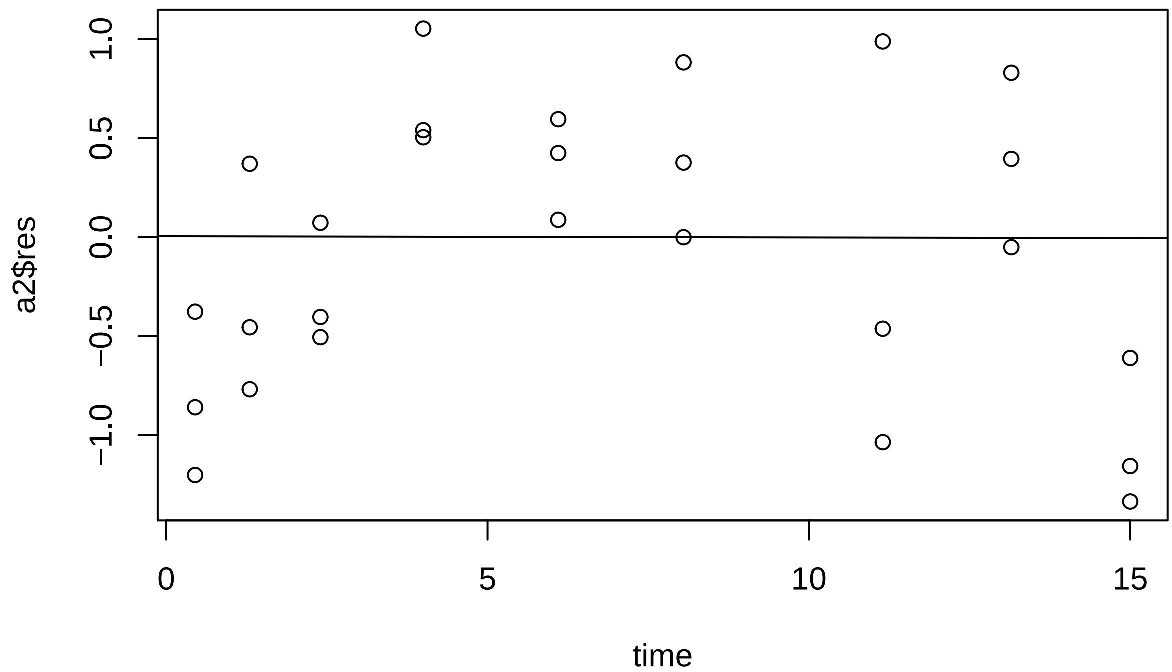

```
a1=rrline1(time,a$res)

plot.new()
plot(time,a$res)

abline(a1$a,a1$b)
```



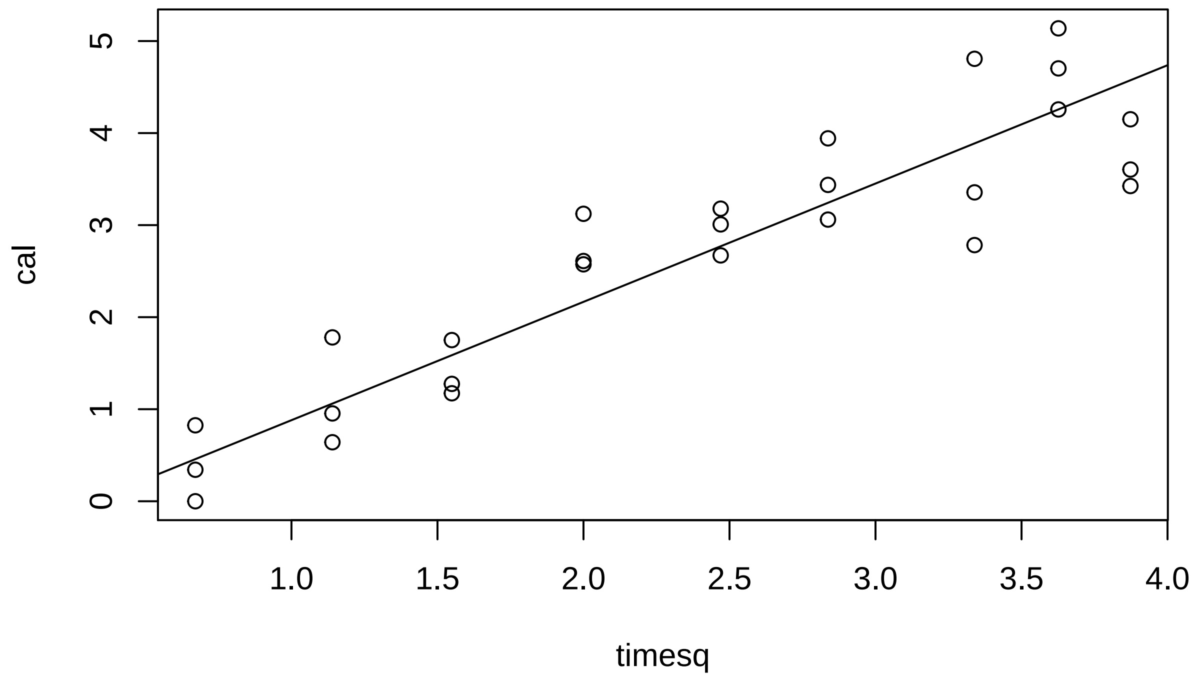

```
a2=rrline1(time,a1$res)

plot.new()
plot(time,a1$res)

abline(a2$a,a2$b)
```



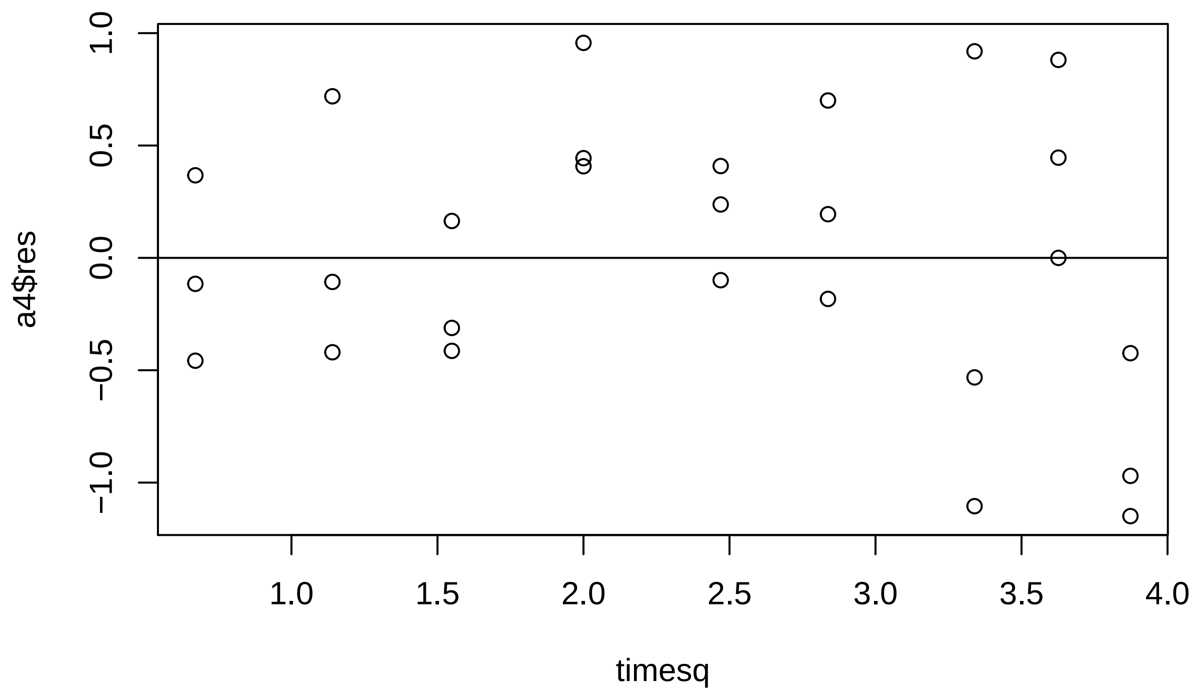

```
a3=rrline1(time,a2$res)

plot.new()
plot(time,a2$res)

abline(a3$a,a3$b)
```




```
#C
#From b we know that the slope is 0.24 , so we go for the near square root transformaton


#d
timesq=sqrt(time)

plot.new()
plot(timesq,cal)

a4=rrline1(timesq,cal)
abline(a4$a,a4$b)
```




```r
plot(timesq,a4$res)
abline(0,0)
```




```
#E


#From c and d, we can see
# more values above zero in the panel C graph and
#we can say the residuals remained the same and the X axis transformed.


#f
#given
intercept = 0.256
slope = -0.124
re = intercept + slope*timesq
out = run.rrline(timesq, re, iter = 2)
```

```
##        a      b |res|
## 1 0.256 -0.124     0
## 2 0.000  0.000     0
##   0.256 -0.124     0
```

```
cat("Final slope" ,out$b )
```

```
## Final slope -0.124
```

```r
cat("Final intercept" ,out$a )
```

```
## Final intercept 0.256
```

Q6

```r
a1=c(5,6,3,11,10)
a2=c(14,10,6,12,21)
a3=c(16,24,15,26,32)

k= matrix(c(a1,a2,a3),nrow=3,byrow=TRUE)

med= medpolish(k)
```

```
## 1: 28
## Final: 28
```

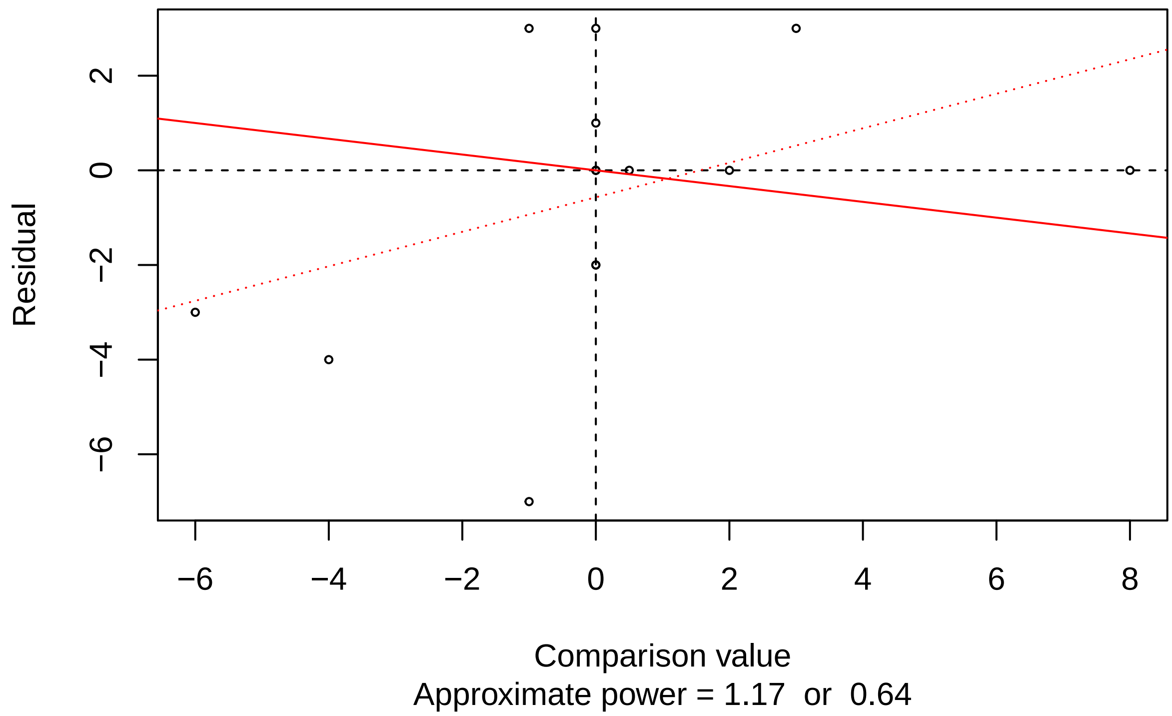
```r
med$overall
```

```
## [1] 12
```

```r
AnalogSquare<- 1-((sum(abs(med$residuals))) /(sum(abs(k-med$overall))))

getwd()
```

```
## [1] "C:/Users/Arun Ram/Documents/R/mid"
```

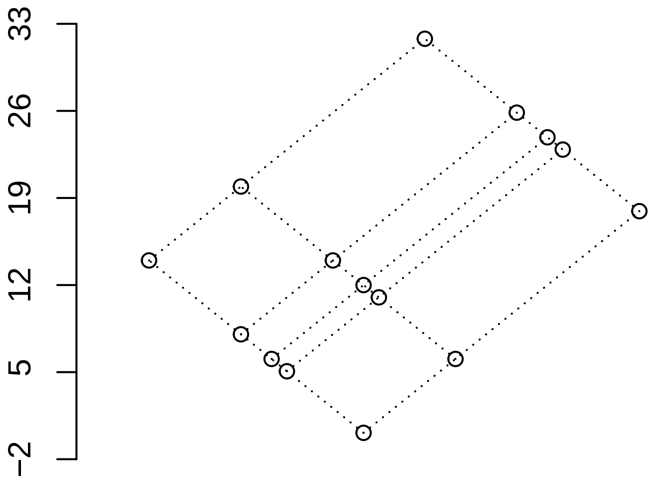
```r
source("rrline.r")

diag.MP <- function(fit){
  fit.comp <- matrix(fit$row,ncol=1) %*% matrix(fit$col,nrow=1)/fit$overall
  plot(fit.comp, fit$res,xlab="Comparison value",ylab="Residual",cex=0.5)
  abline(v=0,h=0,lty=2)
  ls <- lm(c(fit$res)~c(fit.comp))
  abline(ls,col="red",lty=3)
  rr <- run.rrline(fit.comp,fit$res,iter=10)
  abline(rr$a, rr$b, col="red")
  pwr1 <- 1 - rr$b
  pwr2 <- 1 - ls$coef[2]
  title("",paste("Approximate power =",format(round(pwr1,2))," or ", format(round(pwr2,2))))
}


diag.MP(med)
```




```
##    a        b    |res|
##  1 0  1.00000 31.50000
##  2 0 -1.16667 31.91667
##  3 0  1.66667 43.25000
##  4 0 -1.50000 28.00000
##  5 0  1.00000 31.50000
##  6 0 -1.16667 31.91667
##  7 0  1.66667 43.25000
##  8 0 -1.50000 28.00000
##  9 0  1.00000 31.50000
## 10 0 -1.16667 31.91667
##    0 -0.16667 31.91667
```

```
forgetitplot <- function(outmpol,outlim=0,...) {
  # outmpol is output of medpolish in library(eda) or library(stats)
  # be sure to assign dimnames to matrix being polished
  oldpar <- par()
  par(fig=c(0,.7,0,1))
  nc <- length(outmpol$col)
  nr <- length(outmpol$row)
  a <- rep(outmpol$row,nc)
  b <- rep(outmpol$col,rep(nr,nc))
  sqrt2 <- sqrt(2)
  ab <- cbind((a-b)/sqrt2,(a+b)/sqrt2)
  xrange <- range(ab[,1]) + c(-.1,.1)*(max(ab[,1])-min(ab[,1]))
```

```
  yrange <- range(ab[,2]) + c(-.1,.1)*(max(ab[,2])-min(ab[,2]))
  dx <- (xrange[2]-xrange[1])/50
  dy <- (yrange[2]-yrange[1])/50
  plot(ab[,1],ab[,2],axes=F,xlim=xrange,ylim=yrange,xlab="",ylab="",...)
  segments((min(a)-outmpol$col)/sqrt2, (min(a)+outmpol$col)/sqrt2,
           (max(a)-outmpol$col)/sqrt2, (max(a)+outmpol$col)/sqrt2,lty=3)
  segments((outmpol$row-min(b))/sqrt2, (outmpol$row+min(b))/sqrt2,
           (outmpol$row-max(b))/sqrt2, (outmpol$row+max(b))/sqrt2,lty=3)
  # segments((outmpol$row)/sqrt2-min(b), (outmpol$row)/sqrt2+min(b),
  #          (outmpol$row)/sqrt2-max(b), (outmpol$row)/sqrt2+max(b),lty=3)
  yrowloc <-  rep(max(b),nr)
  xrowloc <-  outmpol$row
  # text((xrowloc-yrowloc)/sqrt2-dx,dy+(xrowloc+yrowloc)/sqrt2,format(1:nr))
  text((xrowloc-yrowloc)/sqrt2-dx,dy+(xrowloc+yrowloc)/sqrt2,
       names(sort(outmpol$row)))
  xcolloc <- rep(max(a),nc)
  ycolloc <- outmpol$col
  # text(dx+(xcolloc-ycolloc)/sqrt2,dy+(xcolloc+ycolloc)/sqrt2,format(1:nc))
  text(dx+(xcolloc-ycolloc)/sqrt2,dy+(xcolloc+ycolloc)/sqrt2,
       names(sort(outmpol$col)))
  ynames <- format(round(outmpol$overall + sqrt2*pretty(ab[,2])))
  axis(2,at=pretty(ab[,2]),labels=ynames)
  # add vertical lines when there is an outlier
  if(abs(outlim) > 1e-4) {
    out.index <- which(abs(outmpol$res) > outlim, arr.ind=T)
    # find (r,c) for outlier indices
    zz.x <- outmpol$row[out.index[,1]]
    zz.y <- outmpol$col[out.index[,2]]
    # outlier points at (zz.x-zz.y)/sqrt2, (zz.x+zz.y)/sqrt2
    # draw segment from here to end of residual
    segments((zz.x-zz.y)/sqrt2, (zz.x+zz.y)/sqrt2,
             (zz.x-zz.y)/sqrt2, (zz.x+zz.y)/sqrt2 + outmpol$res[out.index])
  }
  par <- oldpar
  invisible()
}


forgetitplot(med)
```



Q7

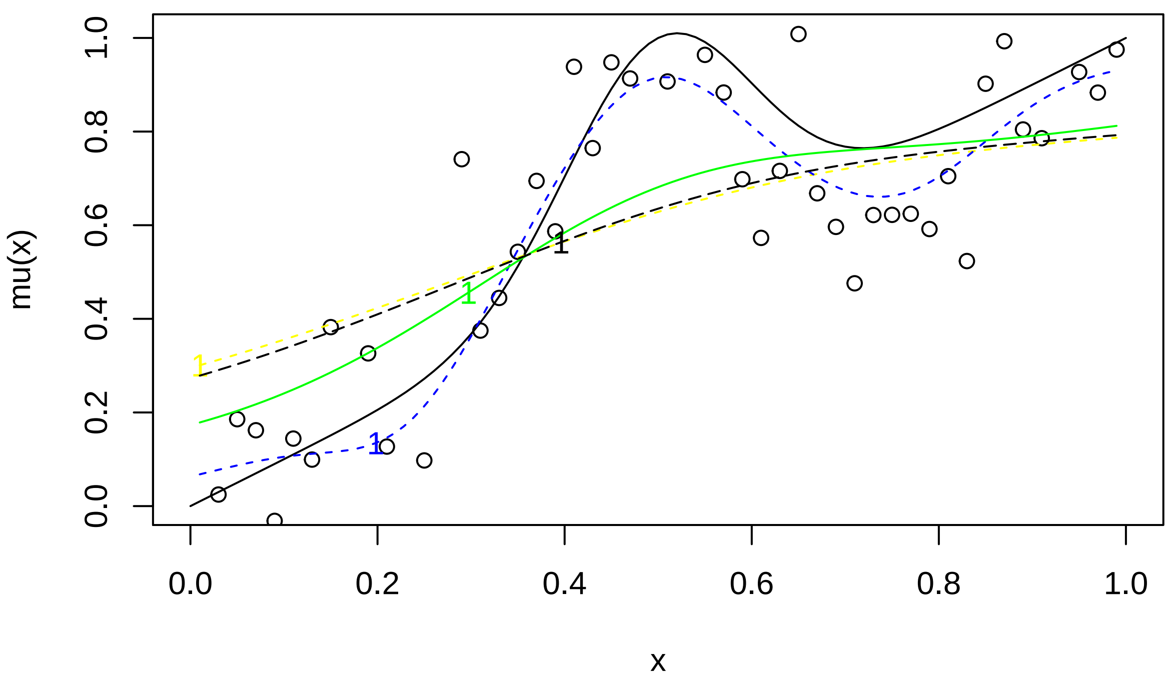
$R(\hat{\mu}) = \frac{1}{n}\sum_{i=1}^{n} E[\hat{\mu}(t_i) - \mu(t_i)]^2 = \frac{1}{n}\sum_{i=1}^{n}[E\hat{\mu}(t_i) - \mu(t_i)]^2 + \frac{1}{n}\sum_{i=1}^{n} var(\hat{\mu}(t_i))$

Let $\hat{\mu}(t_i) = \hat{\theta}$
$\mu(t_i)$=θ
so, $E(\hat{\theta} - \theta)^2 = E(\hat{\theta} - E[\hat{\theta}] + E[\hat{\theta}] - \theta)$
$= E(\hat{\theta} - E[\theta])^2 + E(E(\hat{\theta}) - \theta)^2 + 2E(\hat{\theta} - E[\hat{\theta}])E(\hat{\theta}] - \theta)$
Now,
$2E(\hat{\theta} - E[\hat{\theta}])E(\hat{\theta}] - \theta) == 0$
$E[\hat{\theta} - E[\theta]]^2 = Var(\hat{\theta}) = var(\hat{\mu}(t_i))$
$E(\hat{\theta}) - \theta)^2 = [E\hat{\mu}(t_i) - \mu(t_i)]^2$

```
x = seq(0.01, 0.99, by = 0.02)
y = c(-.0937, .0247, .1856, .1620, -.0316,
      .1442, .0993, .3823, -.0624, .3262, .1271,
      -.4158, .0975, -.0836, .7410, .3749, .4446,
      .5432, .6946, .5869, .9384, .7647, .9478, .9134,
      1.2437, .9070, 1.2289, .9638, .8834, .6982, .5729,
      .7160, 1.0083, .6681, .5964, .4759, .6217, .6221, .6244,
      .5918, .7047, .5234, .9022, .9930, .8045, .7858, 1.1939, .9272,
      .8832, .9751)


mu= function(t){t+0.5*exp(-50*(t-0.5)^2)}

curve(mu(x),0,1)
```

```r
points(x,y)
```

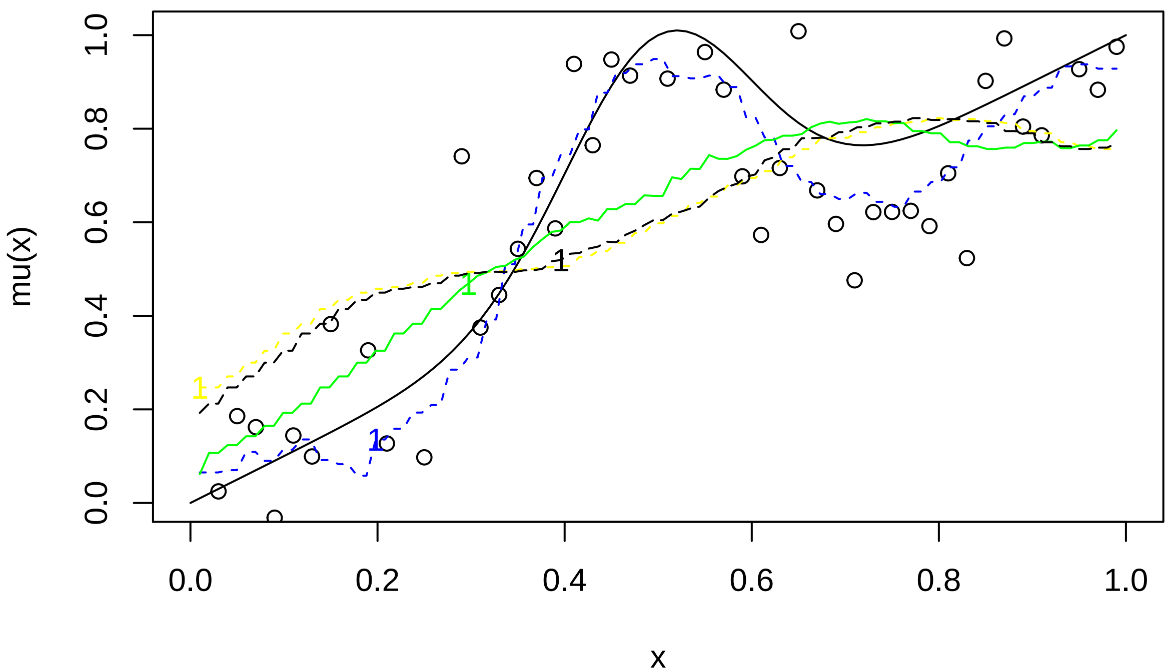
```r
a=ksmooth(x,y,kernel="normal",bandwidth=0.8)
lines(a$x,a$y,col="Yellow",lty=2)
text(a$x[1],a$y[1],col="yellow")
a=ksmooth(x,y,kernel="normal",bandwidth=0.2)
lines(a$x,a$y,col="blue",lty=2)
text(a$x[20],a$y[20],col="blue")
a=ksmooth(x,y,kernel="normal",bandwidth=0.55)
lines(a$x,a$y,col="green")
text(a$x[30],a$y[30],col="green")
a=ksmooth(x,y,kernel="normal",bandwidth=0.75)
lines(a$x,a$y,lty=5)
text(a$x[40],a$y[40])
```




```r
mu= function(t){t+0.5*exp(-50*(t-0.5)^2)}

curve(mu(x),0,1)
points(x,y)
```

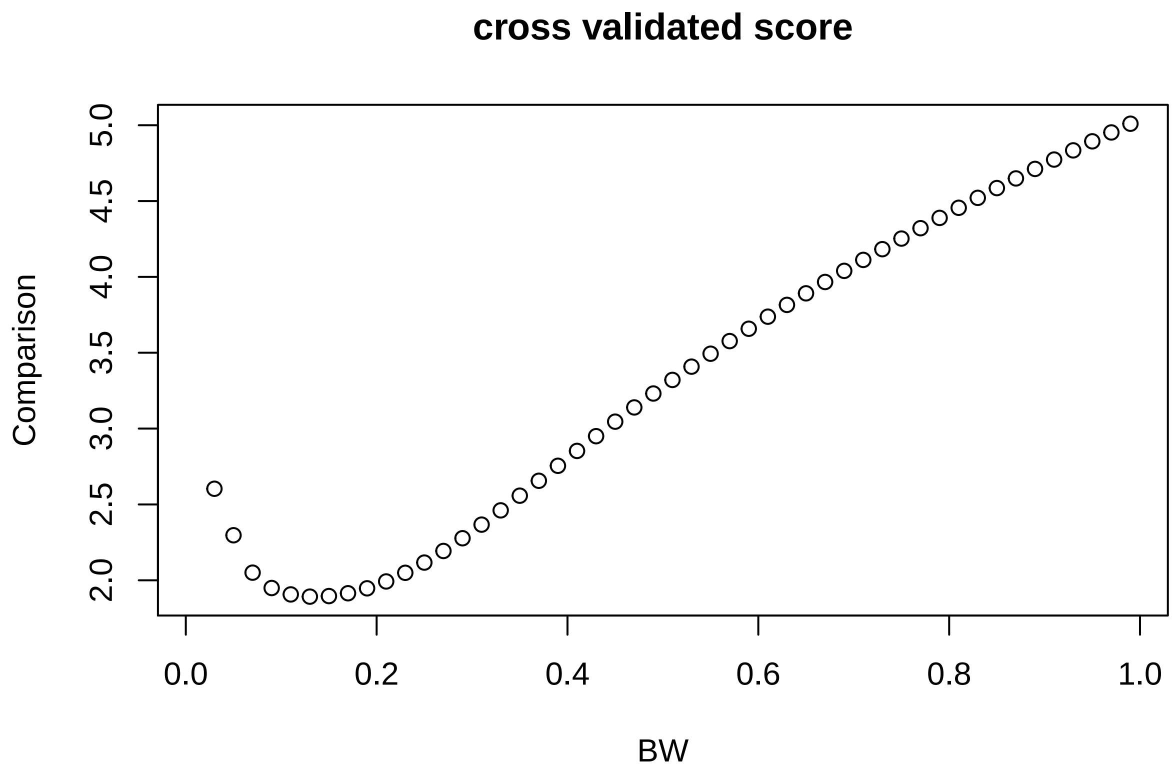
```r
#Box kernel
a=ksmooth(x,y,kernel="box",bandwidth=0.8)
```

```
lines(a$x,a$y,col="Yellow",lty=2)
text(a$x[1],a$y[1],col="yellow")
a=ksmooth(x,y,kernel="box",bandwidth=0.2)
lines(a$x,a$y,col="blue",lty=2)
text(a$x[20],a$y[20],col="blue")
a=ksmooth(x,y,kernel="box",bandwidth=0.55)
lines(a$x,a$y,col="green")
text(a$x[30],a$y[30],col="green")
a=ksmooth(x,y,kernel="box",bandwidth=0.75)
lines(a$x,a$y,lty=5)
text(a$x[40],a$y[40])
```



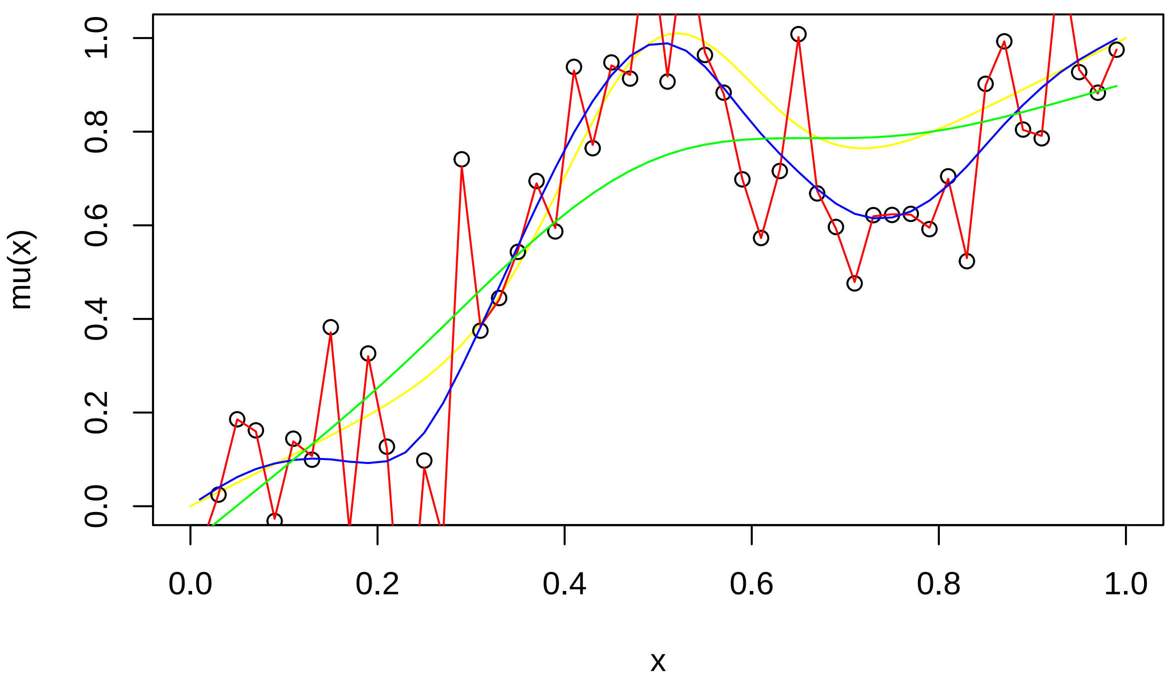

```
#c


CVFUNC = function(bw,x,y){

  n = length(x)

  cv = numeric(n)

  for (i in 1:n){

    fit = ksmooth(x[-i],y[-i],kernel="normal",bandwidth=bw,n.points=n)
```

```r
    yhat=fit$y

    cv[i]=(y[i]-yhat[i])^2

  }

  return(sum(cv))

}



#d



k<-seq(0.01,1,by=0.02)
cv<-c()
for (j in 1:length(k))
{cv<-c(cv, CVFUNC(k[j],x,y))}
plot(k, cv, xlab= "BW" ,ylab="Comparison", main="cross validated score")
```




```r
#CV(lambda) gets minimum at $\lambda 0.1$

#e
```

```
curve(mu(x),0,1,col="yellow")

points(x,y)
smoothingSplinelow = smooth.spline(x, y, spar=0.05)
lines(smoothingSplinelow,col="red")

smoothingSplinedefault = smooth.spline(x, y)
lines(smoothingSplinedefault,col="blue")

smoothingSplinehigh = smooth.spline(x, y,spar=0.9)

lines(smoothingSplinehigh,col="green")
```




```
CVFUNCs = function(bw,x,y){
  n = length(x)
  cv = numeric(n)
  for (i in 1:n){
    fit = smooth.spline(x[-i],y[-i],spar=bw,cv=TRUE)
    yhat=fit$y
    cv[i]=(y[i]-yhat[i])^2
  }
  sum(cv,na.rm=T)
}
```

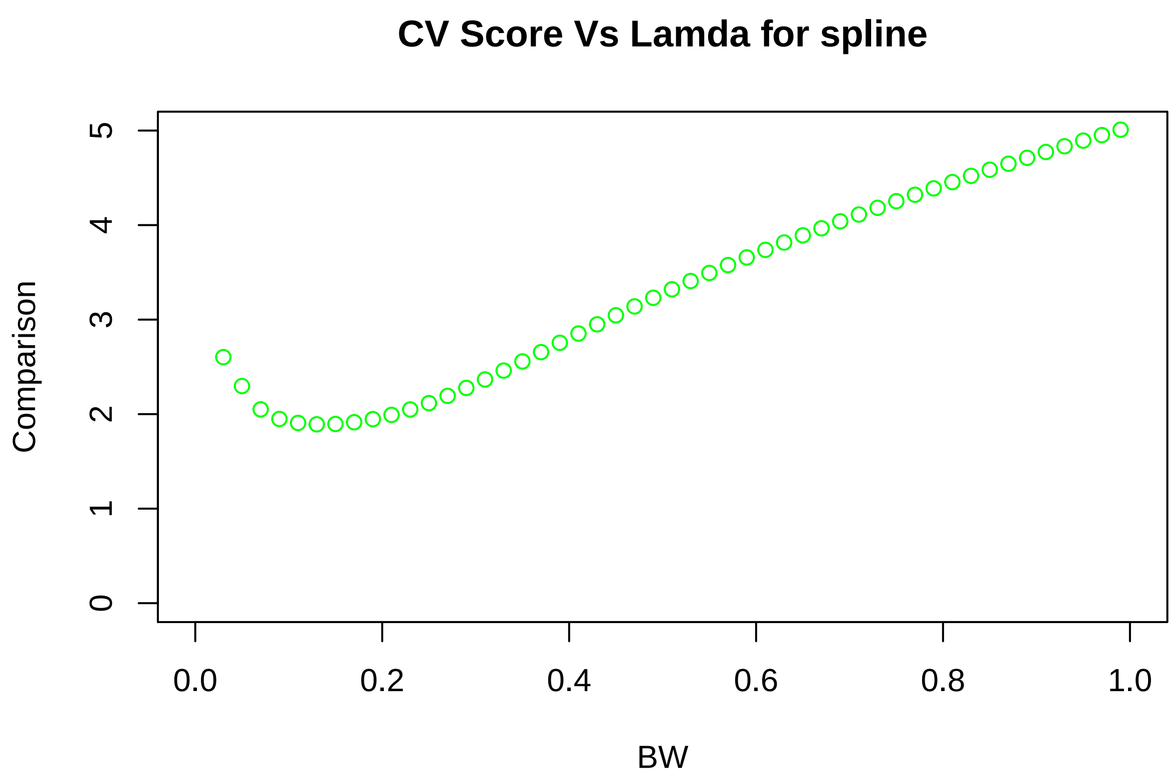
```r
cv1<-c(cv,CVFUNCs(k[j],x,y))
length(k)
```

```
## [1] 50
```

```r
length (cv1)
```

```
## [1] 51
```

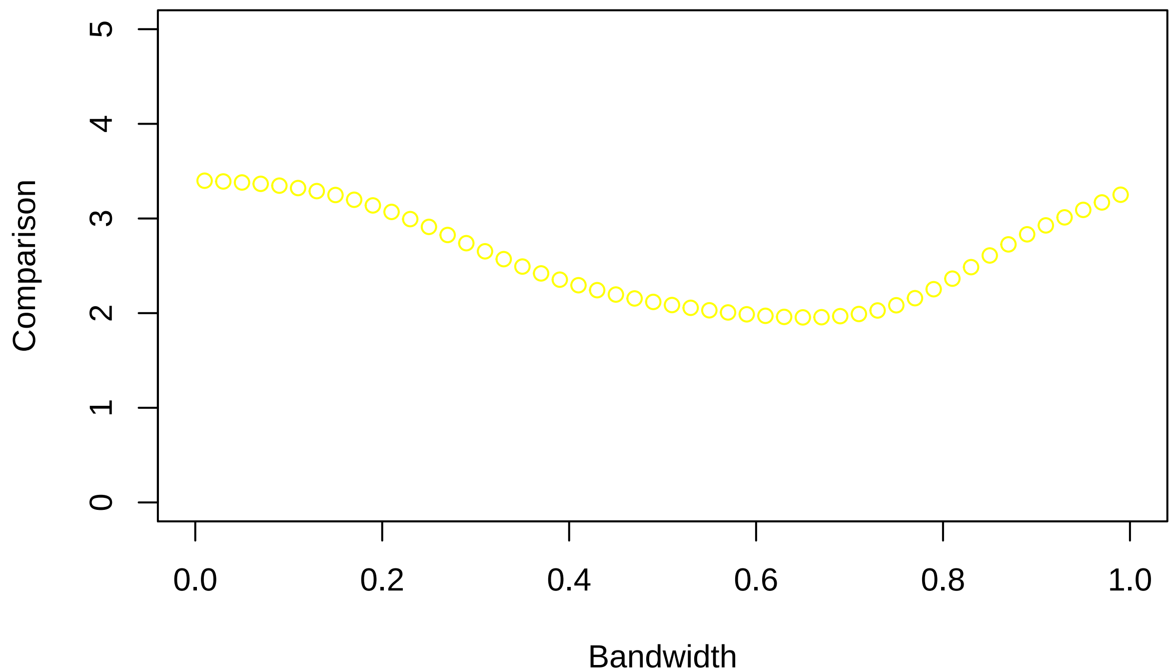
```r
plot(k, cv,  xlab="BW",ylab="Comparison"
     ,xlim=c(0,1),ylim=c(0,5),main= "CV Score Vs Lamda for spline",col="green")
```




```r
GCVFUNCs = function(bw,x,y){
  n = length(x)
  cv = numeric(n)
  for (i in 1:n){
    fit = smooth.spline(x[-i],y[-i],spar=bw,cv=FALSE) #CV = False give GCV Values
    yhat=fit$y
    cv[i]=(y[i]-yhat[i])^2
  }}
sum(cv,na.rm=TRUE)
```

```
## [1] 162.6519
```

```
l<-seq(0.01,1,by=0.02)
gcvscore<-c()
for (j in 1:length(l)){
  gcvscore<-c(gcvscore,CVFUNCs(l[j],x,y))
}


plot(l, gcvscore,  xlab="Bandwidth",ylab=" Comparison "
     ,xlim=c(0,1),ylim=c(0,5),col="yellow")
```




```
OptimumLamda = 0.0003
OptimumSpar = 0.648


Sm = seq(0.4, 0.9, by=0.001)


getCVandGCV = function(x, y, spar) {
  cvFit = smooth.spline(x,y,cv=TRUE,spar=s)
  gcvFit = smooth.spline(x,y,cv=FALSE, spar=s)

  list(cv = cvFit$cv.crit, gcv = gcvFit$cv.crit, lambdaCV = cvFit$lambda, lambdaGCV = gcvFit$lambda)
}
allCV = c()
allGCV = c()
```

```
allCVLambda = c()


q7x<-x
q7y<-y
allGCVLambda = c()

for(s in Sm) {
  fit = getCVandGCV(q7x, q7y, s)
  allCV = c(allCV, fit$cv)
  allGCV = c(allGCV, fit$gcv)
  allCVLambda = c(allCVLambda, fit$lambdaCV)
  allGCVLambda = c(allGCVLambda, fit$lambdaGCV)
}

plot(Sm, allCV, col="red")
points(Sm, allGCV, col="yellow")
```