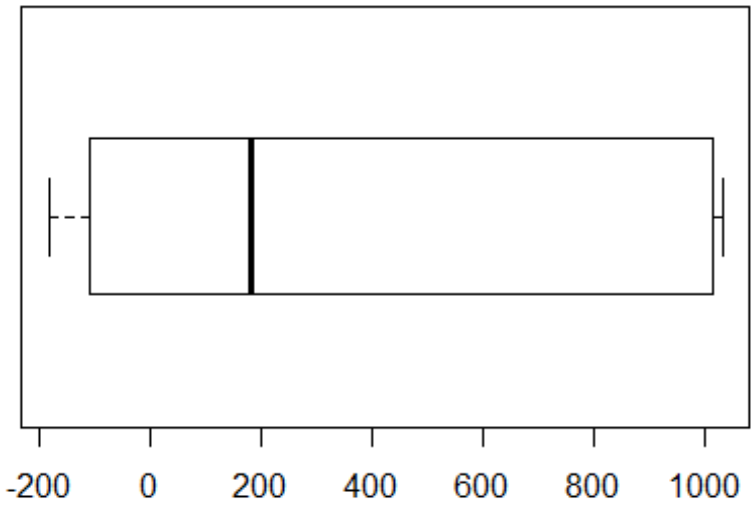
Mid_termQ1.R

```
source("lvalprogs.r")
library("aplpack")

## Loading required package: tcltk

time <- 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,-111,1024,1031,10
28)

data = matrix(data=c(time,wt2),byrow=FALSE,ncol=2,nrow=15)
colnames(data) <- c("time","wt2")

k <- summary(wt2)
print (k)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -184.0  -110.5   182.0   446.3  1014.0  1031.0

l <- lval(wt2)
#d <- lval.sub(wt2)
print (l)

##   Depth  Lower  Upper   Mid Spread pseudo-s
## M   8.0  182.0  182.0 182.0      0   0.0000
## F   4.5 -110.5 1013.5 451.5   1124 833.2224
## E   2.5 -131.0 1026.0 447.5   1157 502.8907
## D   1.5 -167.5 1029.5 431.0   1197 390.1258
## C   1.0 -184.0 1031.0 423.5   1215 326.1339

stem(wt2)

##
##   The decimal point is 3 digit(s) to the right of the |
##
##   -0 | 221111
##   0 | 12
##   0 |
##   1 | 0000000

stem.leaf(wt2)

## 1 | 2: represents 1200
##  leaf unit: 100
##             n: 15
##    6    -0* | 111110
##   (2)    0* | 11
##           t |
```

```
##           f |
##           s |
##    7     0. | 9
##    6     1* | 000000
```

```
boxplot(wt2, horizontal=TRUE) #width=summary(wt2),
```




```
#2

0.4 + 0.007*5000
```

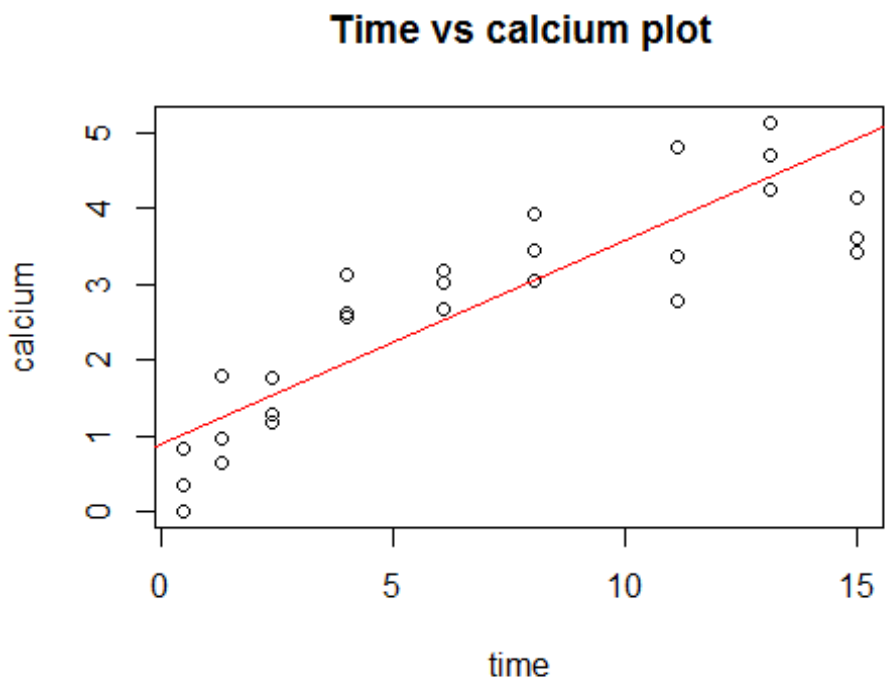
```
## [1] 35.4
```

```
#35.4


#3A)

#Here data is very small and its positively skewed, hence I suggest no transf
ormation

#3B)

#All the five batch have different spreads, so I suggest some transformation
which can result in comparable batches.
#May be a log transformation on y as log(f), where f is f-spread, x as median
.
```

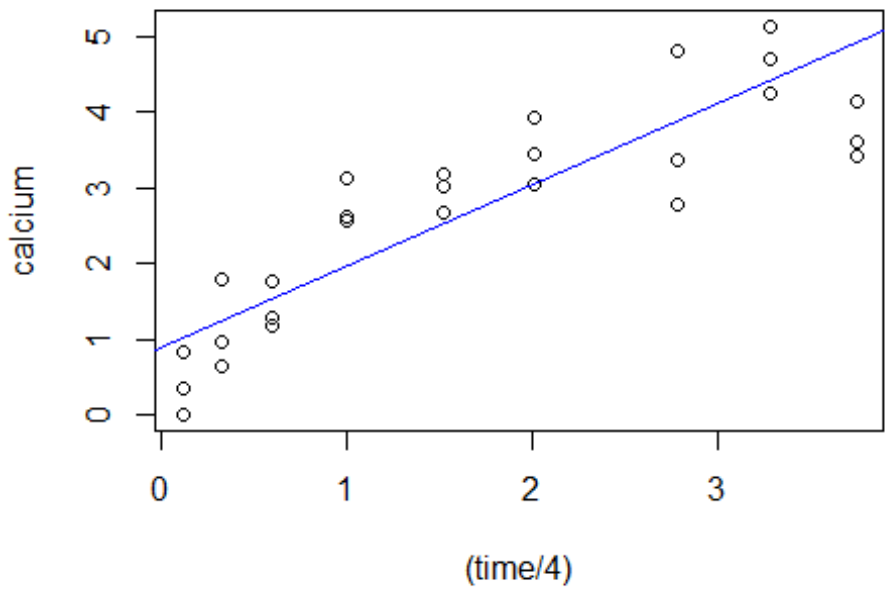
```r
#3C)
#Data is having heavy tails . Transformation would affect the tails, Hence I suggest no transformation

#3D)

#Data is positive skewed, We can transform the data.

#May be we can transform

#x = (upperforth^2+Lowerforth^2)/4*Median
#y = ( upperforth - lowerforth )/2

#4
#The kernel estimate is a weighted average of the observations within the smoothing window.
#two type of linear smothers are
#1) linear regression line and 2) Sample mean.
#linear smothers tend to be affected by outliers and are smooth over sharp features.
#nonlinear smoothers use median hence are flexible and are resistant.
#Disadvantage is the need to use iterative optimization to compute parameter estimate.
```

```r
source("E:/Study stuff/Subjects and courses/S 670 Exploratory Data Analysis/Home work/Mid term/midtermexam/rrline1.R")
#source("E:/Study stuff/Subjects and courses/S 670 Exploratory Data Analysis/Home work/Mid term/midtermexam/run_rrline.r")
source('E:/Study stuff/Subjects and courses/S 670 Exploratory Data Analysis/Home work/Assignment 4/run_rrline.R')
#5)
#5a)
time <- c(0.45,0.45,0.45,1.3,1.3,1.3,2.4,2.4,2.4,4,4,4,6.1,6.1,6.1,8.05,8.05,8.05,
          11.15,11.15,11.15,13.15,13.15,13.15,15,15,15)
calcium <- c(0.342,0,0.825,1.78,0.954,0.641,1.751,1.275,1.173,3.123,2.61,2.574,3.179,
             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)

data <- matrix(data=c(time,calcium),byrow=TRUE,nrow=2,ncol=27)
row.names(data) <- c("time","calcium")

plot(data[2,]~data[1,],main="Time vs calcium plot",xlab="time",ylab="calcium")
```

```
rr <- rrline1(data[1,],data[2,])

#fit <- lm(calcium~time)
abline(rr$a,rr$b,col='red')
```




```
#slope of resistant regression after 1st iteration is 0.26970
#Intercept of resistant regression after 1st iteration is 0.8888776

#5b)
#I would like to apply transformation on above data.

# 5C and D)I see that y - axis varaibles remained same where as x axis variab
les were shrinked specifically 15 was shrinked to a number near 4.I believe t
hat x coordinated were divided by 4.
plot((time/4),calcium)
rr2 <- rrline1((data[1,]/4),data[2,])
abline(rr2$a,rr2$b,col='blue')
```




```
#plot((time^1/4),calcium)

#slope of resistant regression after 1st iteration is 1.078819
#Intercept of resistant regression after 1st iteration is 0.8888776

#5E)The residual values are clustered in panel c where are residuals in panel
D are more scattered. Understanding panel D is more easy.

#5F)
rr2 <- run.rrline(time^(1/2),calcium)
```

```
##          a        b    |res|
## 1 -0.40474  1.28554 13.13373
## 2  0.35643 -0.12409 13.25724
## 3 -0.05537  0.03574 13.17437
## 4  0.01595 -0.01029 13.19824
## 5 -0.00459  0.00296 13.19137
##   -0.09232  1.18986 13.19137
```

```
print (rr2)
```

```
## $a
## [1] -0.09232189
##
## $b
## [1] 1.189859
##
```

```
## $res
##  [1] -0.36385990 -0.70585990  0.11914010  0.51567371 -0.31032629
##  [6] -0.62332629  0.00000000 -0.47600000 -0.57800000  0.83560356
## [11]  0.32260356  0.28660356  0.33258654  0.16158654 -0.17541346
## [16] -0.22360863  0.65939137  0.15339137  0.92618989 -0.52481011
## [21] -1.09781011  0.91554408  0.48054408  0.03454408 -0.91198284
## [26] -0.36598284 -1.09098284
##
## $coef
##              a            b    |res|
## 1 -0.404739546  1.285538623 13.13373
## 2  0.356432222 -0.124090740 13.25724
## 3 -0.055368655  0.035740313 13.17437
## 4  0.015947145 -0.010293838 13.19824
## 5 -0.004593058  0.002964806 13.19137
##   -0.092321892  1.189859165 13.19137
```

```
rr3 <- run.rrline(time^(1/2),calcium)
```

```
##          a        b    |res|
## 1 -0.40474  1.28554 13.13373
## 2  0.35643 -0.12409 13.25724
## 3 -0.05537  0.03574 13.17437
## 4  0.01595 -0.01029 13.19824
## 5 -0.00459  0.00296 13.19137
##   -0.09232  1.18986 13.19137
```

```
ynew <- calcium-(-0.124)*time^(1/2) - 0.256
rr4 <- rrline1(time^(1/2),ynew)
rr5 <- rrline1(time^(1/2),rr4$res)
print(rr5)
```

```
## $a
## [1] 0.4065408
##
## $b
## [1] -0.1399387
##
## $sumres
## [1] 13.26545
##
## $res
##  [1] -0.3919284 -0.7339284  0.0910716  0.5026035 -0.3233965 -0.6363965
##  [7]  0.0000000 -0.4760000 -0.5780000  0.8500091  0.3370091  0.3010091
## [13]  0.3620052  0.1910052 -0.1459948 -0.1824486  0.7005514  0.1945514
## [19]  0.9833885 -0.4676115 -1.0406115  0.9819180  0.5469180  0.1009180
## [25] -0.8377259 -0.2917259 -1.0167259
```

```
#slope -0.1399387
#intercept 0.4065408
```

```r
library("miscTools")
source('E:/Study stuff/Subjects and courses/S 670 Exploratory Data Analysis/Home work/Mid term/midtermexam/forgetitplot.R')
T1 <- c(5,6,3,11,10)
T2 <- c(14,10,6,12,21)
T3 <- c(16,24,15,26,32)

Area <- matrix(data=c(T1,T2,T3),nrow=3,ncol=5,byrow=TRUE)
Area1 <- Area
row.names(Area) <- c('T1','T2','T3')
colnames(Area) <- c('A1','A2','A3','A4','A5')
#6A)
run.MP <- medpolish(Area)
```

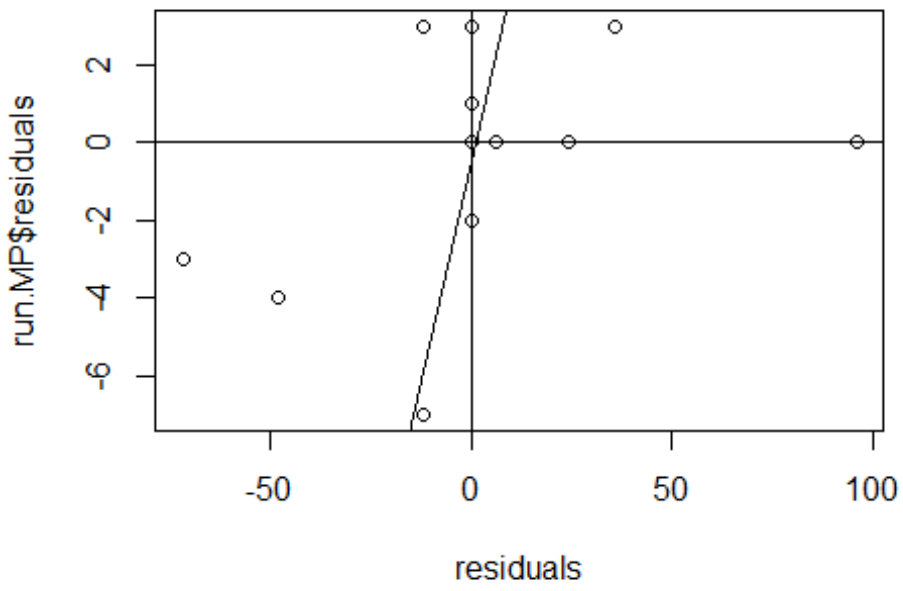
```
## 1: 28
## Final: 28
```

```r
Area <- cbind(Area,run.MP$row)
Area <- rbind(Area,c(run.MP$col,0))
Area[4,6] <- run.MP$overall
print (Area)
```

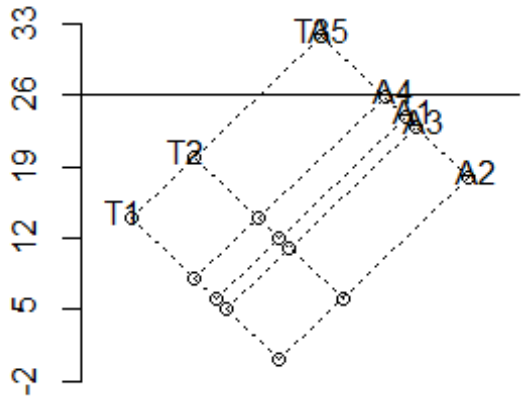
```
##    A1 A2 A3 A4 A5
## T1  5  6  3 11 10 -6
## T2 14 10  6 12 21  0
## T3 16 24 15 26 32 12
##    -1  0 -6  2  8 12
```

```r
#After applying Median polish each term in matrix Area can be obtained
#by adding Overall + roweffect + coloumn effect + residual.

#6B)
AnalogR2 <- 1 - sum(abs(run.MP$residuals))/sum(abs(Area1 - run.MP$overall))
#Analog r2 is pretty good indicating its a good fit.

#6C)
#Diagnostic plot can used to understand how well the model fits the data.

residuals <- matrix(run.MP$row,ncol=1) %*% matrix(run.MP$col,nrow=1)
plot(residuals,run.MP$residuals)
fit=lm(run.MP$residuals~residuals)
abline(fit,v=0,h=0)
```

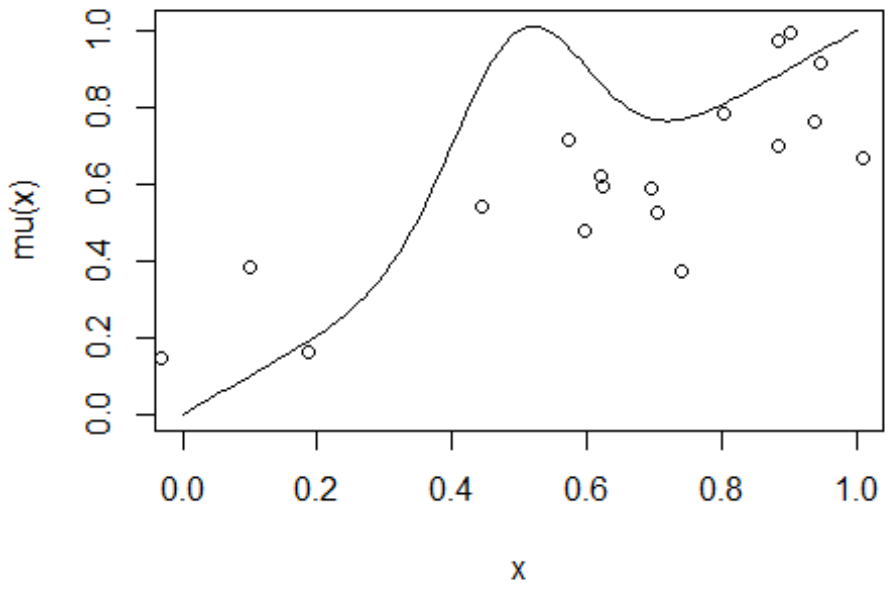
```
## Warning in abline(fit, v = 0, h = 0): only using the first two of 30
## regression coefficients
```



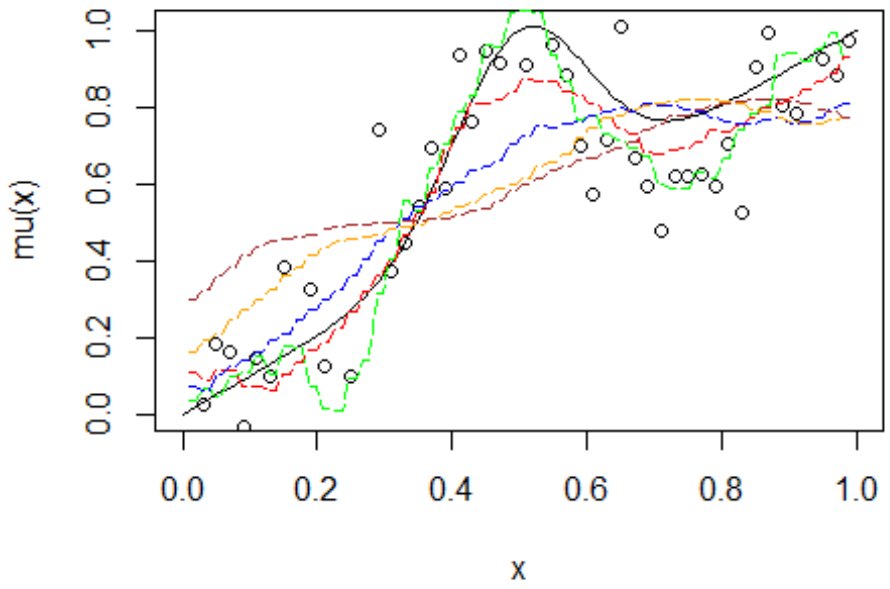

```r
#Data does not reveal any treand.

#6D)
forgetitplot(run.MP)
abline(h=10)
```



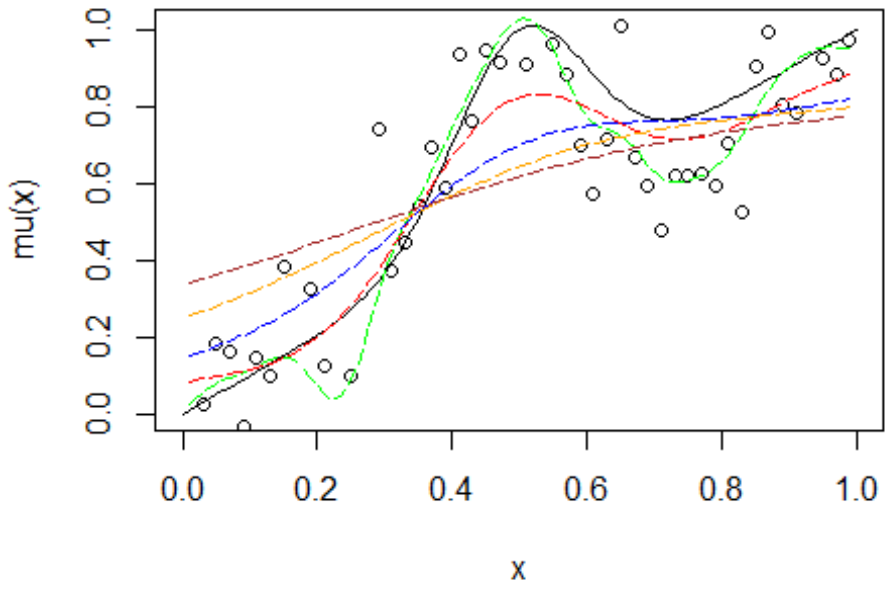

```
#

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, .93
84, .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)
mat = matrix(y, ncol = 2, byrow = T)

mu = function(t){
  t + 0.5 *exp(-50*(t-0.5)^2)
  }
curve(mu(x),0,1)
points(mat[,1],mat[,2])
```




```r
#7B)

plot_ksmooth = function(x, y, mu, method) {
  lam = c(0.1, 0.3, 0.5, 0.7, 0.9)
  col = c("green", "red", "blue", "orange", "brown")
  curve(mu(x), 0, 1)
  points(x, y)
  for(i in 1:length(lam)) {
    fit = ksmooth(x, y, kernel = method, bandwidth = lam[i])
    lines(fit$x, fit$y, lty = 5, col = col[i])
  }
}

plot_ksmooth(x,y,mu,"box")
```

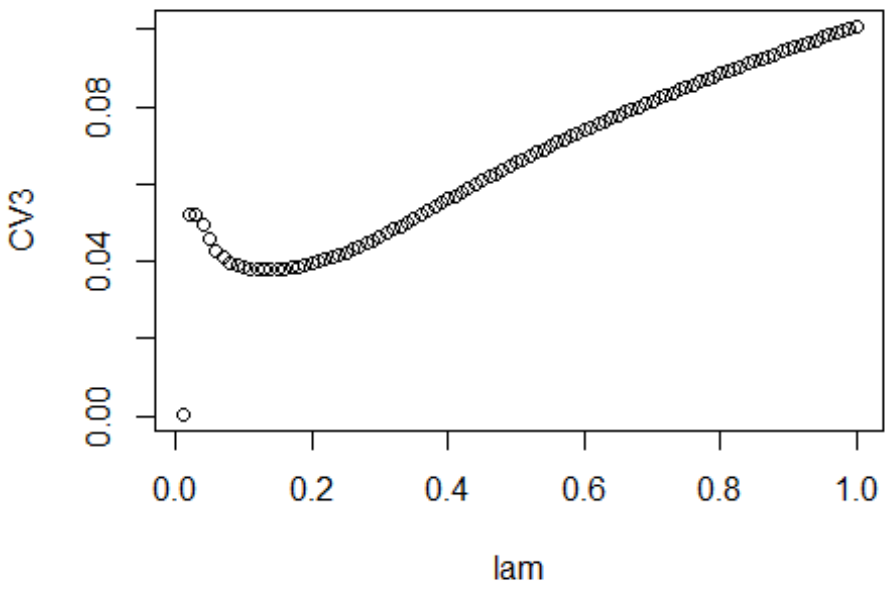



```
plot_ksmooth(x,y,mu,"normal")
```

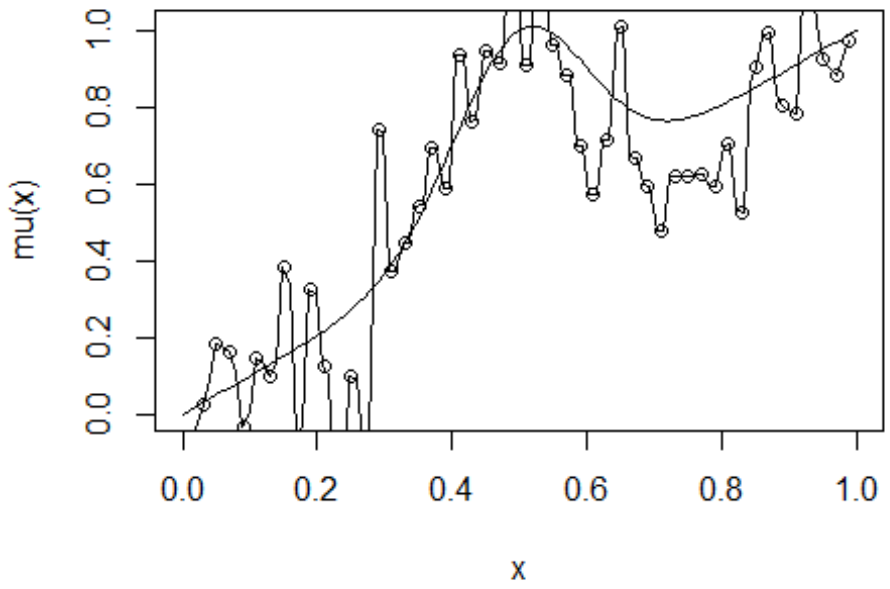



```
#7C)
CV <- function(x, y, lam, spline = TRUE) {
```

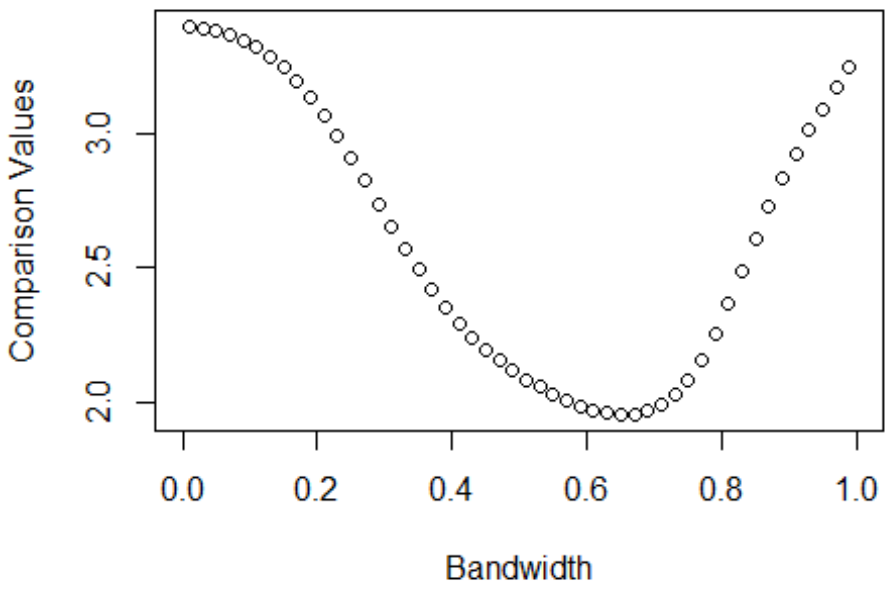
```
n <- length(x)
CV <-numeric(n)
for(i in 1:n) {
  if(spline == FALSE) {
    fit = ksmooth(x[-i], y[-i], kernel = "normal", bandwidth = lam, n.points
= n)
  } else {
    fit = smooth.spline(x[-i], y[-i], spar = lam)
  }
  CV[i] = (y[i] - fit$y[i])^(2)
}
CV = sum(CV, na.rm = T)
return(CV/n)
}

CV_plot = function(x, y, spline = TRUE) {
  lam = seq(0.01, 1, by = 0.01)
  CV3 = c()
  for(i in 1:length(lam)) {
    CV2 <- CV(x, y, lam[i], spline = spline)
    CV3 = c(CV3, CV2)
  }
  plot(lam,CV3)
  lam_min = lam[which.min(CV3)]
  return(lam_min)
  }

minLamda = CV_plot(x, y, spline = FALSE)
```



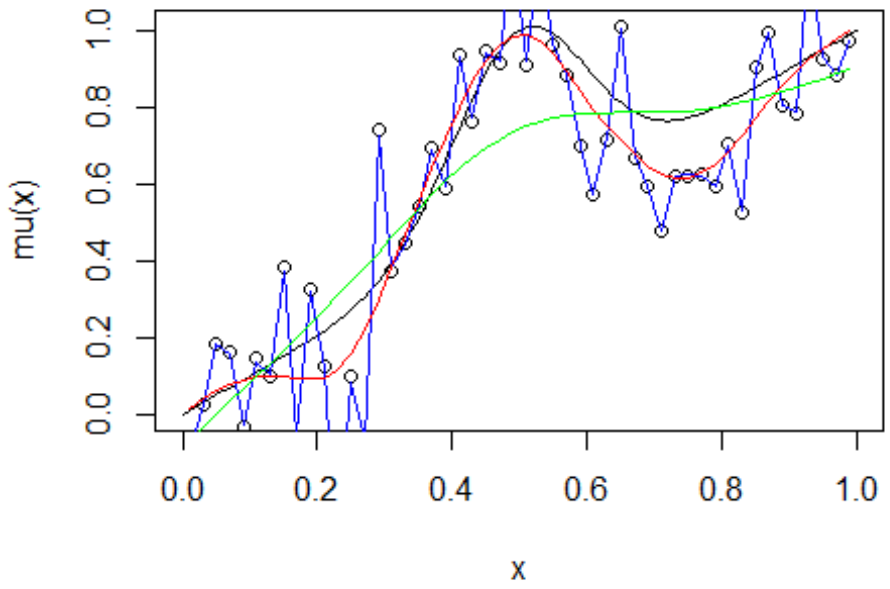

```r
curve(mu(x), 0, 1)
points(x, y)
fit = ksmooth(x, y, kernel = "normal", bandwidth = minLamda)
lines(fit$x, fit$y)
```



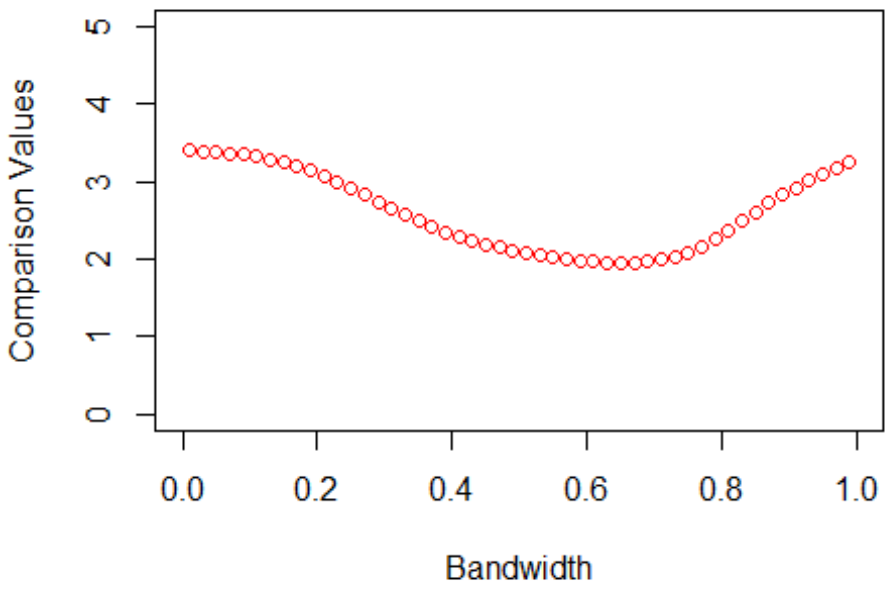

```
CV1 = function(bw,x,y){
  CV= numeric(length(x))
  for (i in 1:length(x)){
    fit = smooth.spline(x[-i],y[-i],spar=bw,cv=TRUE)
    hat_y=fit$y
    CV[i]=(y[i]-hat_y[i])^2
    }
  sum(CV,na.rm=T)
  }
Score_CV=c()
for (j in 1:length(x)){
  Score_CV <- c(Score_CV,CV1(x[j],x,y))
  }
plot(x, Score_CV, xlab="Bandwidth",ylab="Comparison Values",xlim=c(0,1))
```



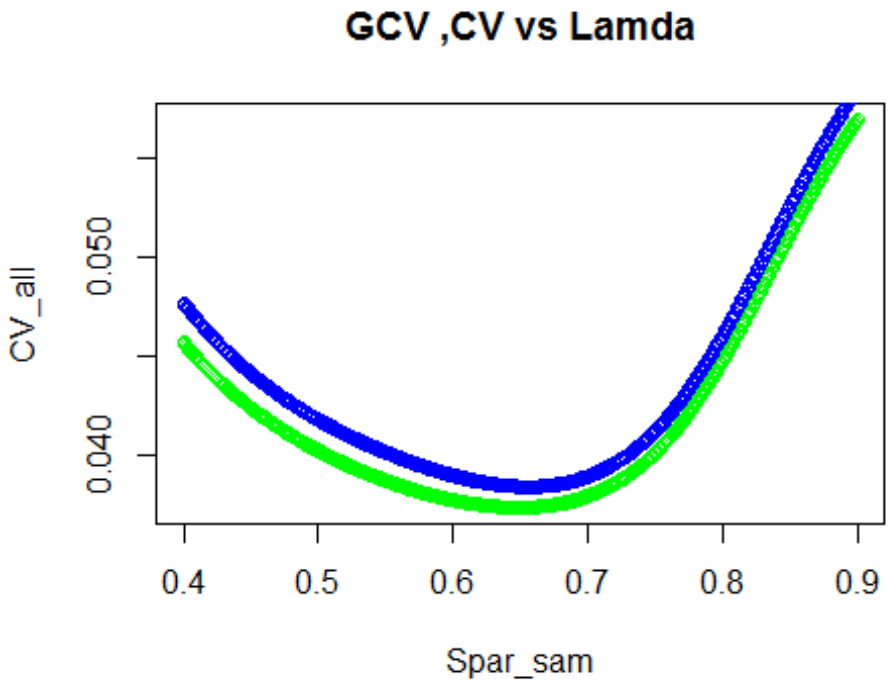

```r
#7E)

curve(mu(x),0,1)
points(x,y)
low = smooth.spline(x, y, spar=0.05)
lines(low,col="blue")
def = smooth.spline(x, y)
lines(def,col="red")
high = smooth.spline(x, y,spar=0.9)
lines(high,col="green")
```




```
#7.f
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)
  hat_y=fit$y
  CV[i]=(y[i]-hat_y[i])^2
  }
sum(CV,na.rm=T)
}
l<-seq(0.01,1,by=0.02)
Score_CV<-c()
for (j in 1:length(l)){
  Score_CV<-c(Score_CV,CVFUNCs(l[j],x,y))
  }
plot(l, Score_CV, xlab="Bandwidth",ylab="Comparison Values",xlim=c(0,1),ylim=
c(0,5),col="red")
```




```
q7x<-x
q7y<-y
OptimumLamda <- 0.0003
OptimumSpar <- 0.648
Spar_sam <- seq(0.4, 0.9, by=0.001)
getCVandGCV <- function(x, y, spar) {
  Fit_CV <- smooth.spline(x,y,cv=TRUE,spar=s)
  Fit_GCV <- smooth.spline(x,y,cv=FALSE, spar=s)
  list(cv = Fit_CV$cv.crit, gcv = Fit_GCV$cv.crit, CV_lambda = Fit_CV$lambda,
GCV_lambda = Fit_GCV$lambda)
  }
CV_all <- c()
GCV_all <- c()
CVLambda_all <- c()
Lambda_allGCV <- c()
for(s in Spar_sam) {
  fit <- getCVandGCV(q7x, q7y, s)
  CV_all <- c(CV_all, fit$cv)
  GCV_all <- c(GCV_all, fit$gcv)
  CVLambda_all <- c(CVLambda_all, fit$CV_lambda)
  Lambda_allGCV = c(Lambda_allGCV, fit$GCV_lambda)
  }
plot(Spar_sam, CV_all, col="green", main="GCV ,CV vs Lamda")
points(Spar_sam, GCV_all, col="blue")
```


GCV ,CV vs Lamda
CV_all
0.040
0.050
0.4
0.5
0.6
0.7
0.8
0.9
Spar_sam