Mid Term (S-670)

FNU Anirudh

November 4, 2015

Solution 1:-

```
library(aplpack)
```

```
## Loading required package: tcltk
```

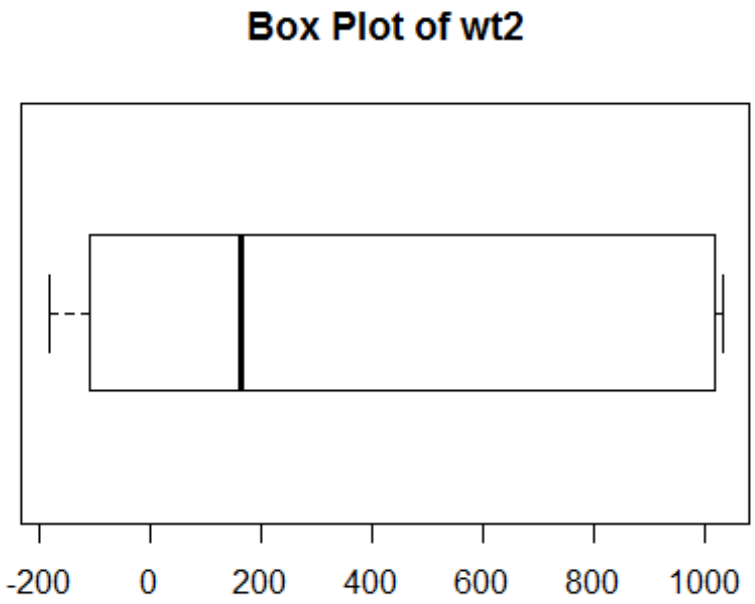
```
wt2<- c(143,-184,182,-110,1017,986,1010,-111,-60,-151,-111,1024,1031,1028)
summ=summary(wt2)
summ
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -184.0  -110.8   162.5   406.7  1015.0  1031.0
```

```
stem.leaf(wt2)
```

```
## 1 | 2: represents 1200
##  leaf unit: 100
##             n: 14
##    6    -0* | 111110
##   (2)    0* | 11
##           t |
##           f |
##           s |
##    6     0. | 9
##    5     1* | 00000
```

```
boxplot(wt2,horizontal=TRUE,main="Box Plot of wt2")
```



Solution 2:-

n=5000

Number of Outliers= 0.4+0.007*n

Outliers= 0.4+ 0.007* 5000= 35.4

Solution 3:-

a) Data is left skewed and transformation would be make data symmetric. X= ((d_u)^2 + (d_l)^2) / (4M) , Y = ((d_u) - (d_l)) / 2 Where (d_u) and (d_l) = distance from (x_l) and (x_u) to M. Using this we will now calculate the slope and then the value of power transform.

b) Spreads are large and it's necessary to transform data to do better analysis of spread versus level plot and will give good estimate of mid summaries and accordingly we can use transformation. X= log M, Y= log(d_f) is F-spread and M is the median for all batches.

c) Since there are heavy tails which won't go even after transformation hence transformation will not be a good idea.

d) There is one heavy tail and one light tail hence we can say that there is skewness in data hence we need to perform transformation since it is single batch we will plot transformation plot between X= ((d_u)^2 + (d_l)^2) / (4M) , Y = ((d_u) - (d_l)) / 2 Where (d_u) and (d_l) = distance from (x_l) and (x_u) to M. We can know the power transformation.

Solution 4

A linear smoother means that the vector y-hat of predicted values at the observed predictor values is a linear function of the data vector y.

Types of Linear Smoothers

1. Kernel Smoother
2. Lowess Smoother

Problems with Linear Smoothers

1) Strongly affected by outliers
2) Smooth over sharp features

Advantage of Non- Linear Smoother

- reduce influence of outliers and easy to do by hand

Disadvantage of Linear Smoother

- cannot be expressed as summation of wiyi

Solution 5

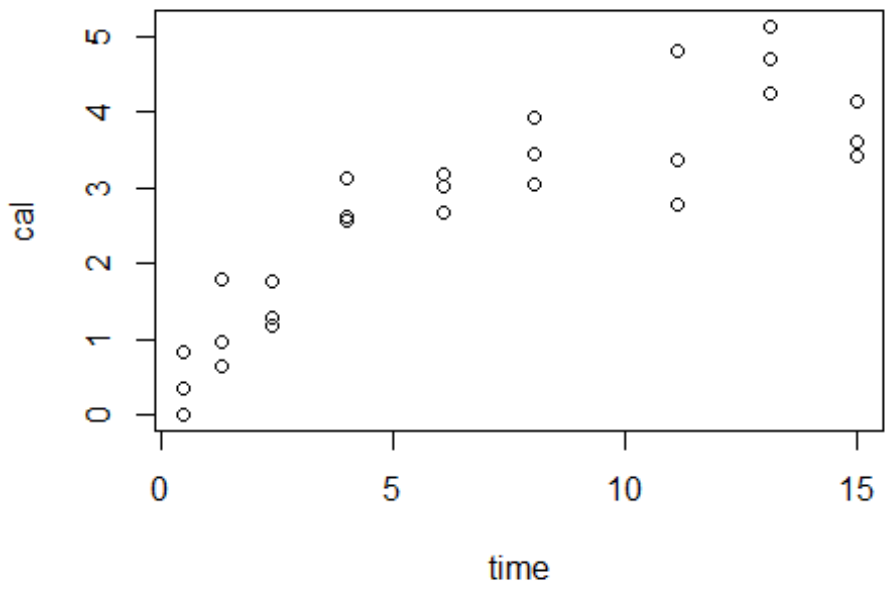
```
time<- c(0.450,0.45,0.450,1.300,1.300,1.300,2.400,2.400,2.400,4.000,4.00,
         4.000,6.100,6.100,6.100,8.05,8.050,8.050,11.150,11.150,11.150,
         13.150,13.150,13.150,15.000,15.00,15.000)
cal<- c(0.342,0.00,0.825,1.780,0.954,0.641,1.751,1.275,1.173,3.123,2.61
        ,2.574,3.179,3.008,2.671,3.06,3.943,3.437,4.807,3.356,2.783,5.138
        ,4.703,4.257,3.604,4.15,3.425)
source("rrline.r")
rr=rrline1(time,cal)
# a)
inter=rr$a
slope=rr$b
inter
```

```
## [1] 0.8888776
```

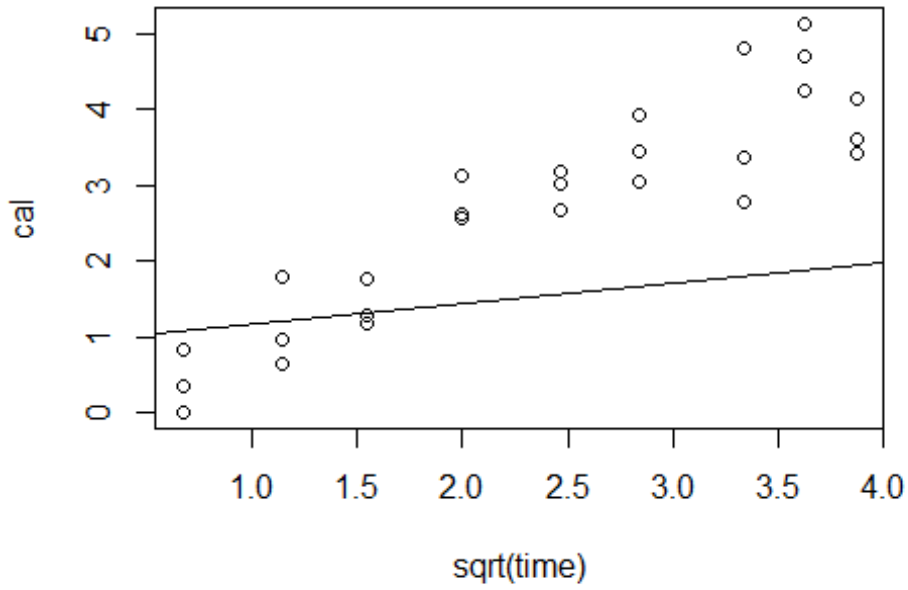
```
slope
```

```
## [1] 0.2697046
```

```
# b) To straighten the plot, we need to do re-expression or perform
# transformation.
# c)
plot(time,cal)
```




```
plot(sqrt(time),cal)
abline(inter,slope)
```



```r
# I have applied square root transformation

# d)
rr2=rrline1(sqrt(time),cal)
rr2
```

```
## $a
## [1] -0.4047395
##
## $b
## [1] 1.285539
##
## $sumres
## [1] 13.13373
##
## $res
##  [1] -0.11562598 -0.45762598  0.36737402  0.71900000 -0.10700000
##  [6] -0.42000000  0.16419167 -0.31180833 -0.41380833  0.95666230
## [11]  0.44366230  0.40766230  0.40869336  0.23769336 -0.09930664
## [16] -0.18265773  0.70034227  0.19434227  0.91911838 -0.53188162
## [21] -1.10488162  0.88100000  0.44600000  0.00000000 -0.97013013
## [26] -0.42413013 -1.14913013
```

```r
inter2=rr2$a
slope2=rr2$b
inter2
```

```
## [1] -0.4047395
```

```r
slope2
```

```
## [1] 1.285539
```

```r
# e)
# Panel c shows curvy trend and we can predict residuals.We need residual
# plot where residuals are scattered all over the place and we cannot
# find a trend then we can say that we have fit appropriate model.
# After performing transformation Panel d we can still see that most
# of the points are above the line and on bottom left there are no
# residuals which implies that transformation performed is still not
#perfect.
# f)
Slope3=0.256
Inter3=-0.124
newy= 0.256*sqrt(time)+Inter3
newy
```

```
##  [1] 0.04773002 0.04773002 0.04773002 0.16788491 0.16788491 0.16788491
##  [7] 0.27259349 0.27259349 0.27259349 0.38800000 0.38800000 0.38800000
## [13] 0.50827336 0.50827336 0.50827336 0.60233656 0.60233656 0.60233656
## [19] 0.73082536 0.73082536 0.73082536 0.80433098 0.80433098 0.80433098
## [25] 0.86748374 0.86748374 0.86748374
```

```
rr3=rrline1(sqrt(time),newy)
rr3
```

```
## $a
## [1] -0.124
##
## $b
## [1] 0.256
##
## $sumres
## [1] 1.165734e-15
##
## $res
##  [1] -5.551115e-17 -5.551115e-17 -5.551115e-17  0.000000e+00  0.000000e+00
##  [6]  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
## [11]  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
## [16]  0.000000e+00  0.000000e+00  0.000000e+00  1.110223e-16  1.110223e-16
## [21]  1.110223e-16  1.110223e-16  1.110223e-16  1.110223e-16  1.110223e-16
## [26]  1.110223e-16  1.110223e-16
```

```
slopenew=rr3$a
internew=rr3$b
slopenew
```

```
## [1] -0.124
```

```
internew
```

```
## [1] 0.256
```

Solution 6

```
# a)
T1<-c(5,6,3,11,10)
T2<-c(14,10,6,12,21)
T3<-c(16,24,15,26,32)
coln<- c("A1","A2","A3","A4","A5")
flyinsctble<-rbind(T1,T2,T3)
colnames(flyinsctble)<-coln
flyinsctble.MP<-medpolish(flyinsctble)
```

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

```
flyinsctble.MP
```

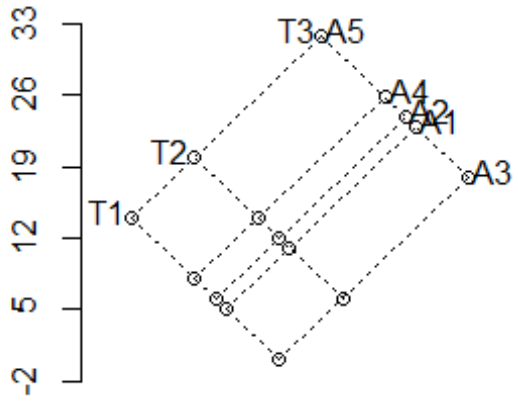
```
##
## Median Polish Results (Dataset: "flyinsctble")
##
## Overall: 12
##
## Row Effects:
## T1 T2 T3
```

```
## -6  0 12
##
## Column Effects:
## A1 A2 A3 A4 A5
## -1  0 -6  2  8
##
## Residuals:
##    A1 A2 A3 A4 A5
## T1  0  0  3  3 -4
## T2  3 -2  0 -2  1
## T3 -7  0 -3  0  0
```

```
# Row Effects shows that Flying insects visits T3 (Pink) most followed by
# white which is second most visisted and then red which is least visited.
# Column effect shows that insects visit A5 most then A4 then A2, then A1
# lastly A3. A5 > A4 >A2 > A1 > A3
# b)
sum.residual<- sum(abs(flyinsctble.MP$residuals))
sum.data <- sum(abs(flyinsctble - flyinsctble.MP$overall))
Analog.RSqr<- 1-(sum.residual/sum.data)
Analog.RSqr
```

```
## [1] 0.7113402
```

```
# c)
# Diagnositc Plot is plot between residuals and comparision values.The
# Plot reveals trends or patterns and helps us understand if data is
# systematic or not from additive model.
# Before plotting diagnositc plot, we should have analyzed residuals
# and know that there is a problem with non-additivity and we already
# expect trends. The slope helps us in transformation to remove
# non-additivity.
# We plot the diagnostic plot between the residuals on y-axis and
# comparision values on x-axis.
source("myplotfit.r")
myplotfit(flyinsctble.MP)
```




```
#We see that the affect of (T3,A4) = 26 and (T2,A5) nearly 24 are
#almost similiar, they have the same affect.
```

Solution 7

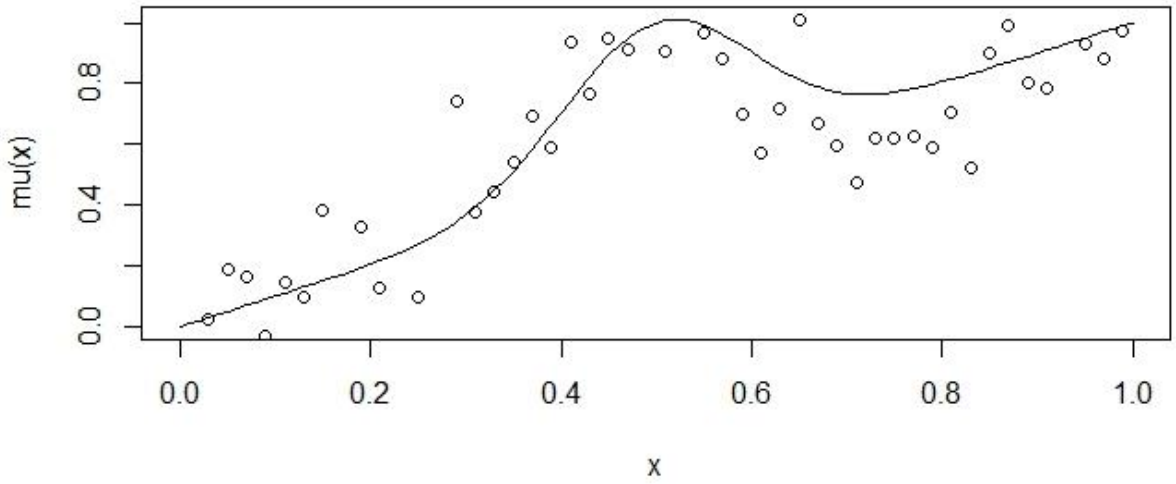
```
x = seq(0.01,0.99,by=0.02)
y=c(-0.0937,0.0247,0.1856,0.1620,-0.0316, 0.1442,0.0993,0.3823,-0.0624,
    0.3262,0.1271,-0.4158,0.0975,-0.0836,0.7410,0.3749,0.4446,0.5432,0.6946
    ,0.5869,0.9384,0.7647,0.9478,0.9134,1.2437,0.9070,1.2289,0.9638,0.8834,
    0.6982,0.5729,0.7160,1.0083,0.6681,0.5964,0.4759,0.6217,0.6221,0.6244,
    0.5918,0.7047,0.5234,0.9022,0.9930,0.8045,0.7858,1.1939,0.9272,0.8832,
    0.9751)
mu = function(t){t + 0.5 *exp(-50*(t-0.5)^2)}
curve(mu(x),0,1)
points(x,y)
kernel_cv = function(x, y, lam)
{
  n = length(x)
```

```
  cv=numeric(n)
  for(i in 1:n)
  {
    fit = ksmooth(x[-i], y[-i], kernel = "normal", bandwidth = lam, n.points
= n)
    cv[i] = (y[i] - fit$y[i])^(2)
  }
  cv = mean(cv, na.rm = T)
  return(cv)

}

spline_cv = function(x, y, lam)
{
  fit = smooth.spline(x, y,spar = lam ,cv=TRUE)
  return(fit$cv.crit)
}

spline_gcv = function(x, y, lam)
{
  fit = smooth.spline(x, y,spar = lam ,cv=FALSE)
  return(fit$cv.crit)
}

# part f

plot_cv = function(x, y, spline = 0)
{
  lam = seq(0.01, 1, by = 0.01)
  cv3 = c()
```

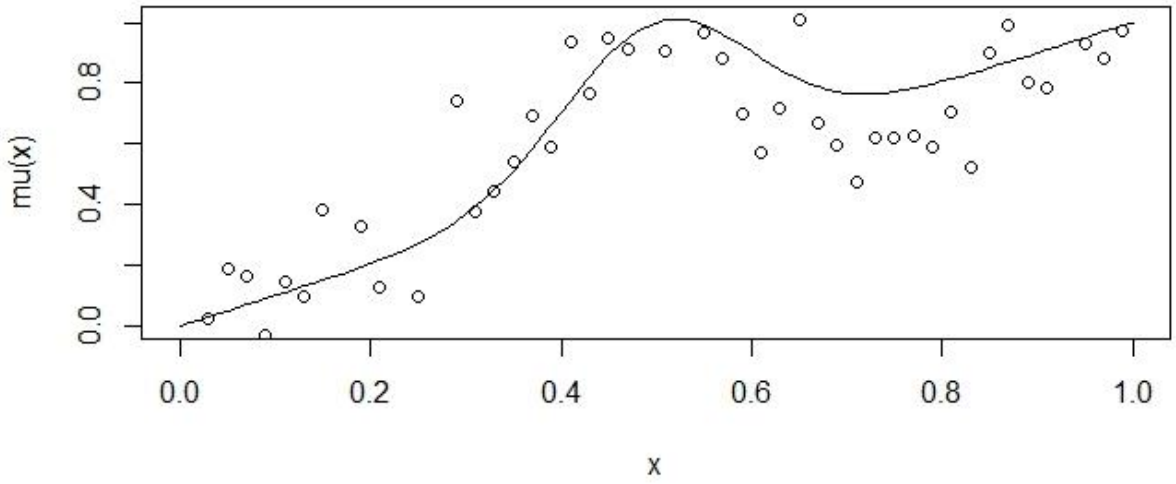
```
  for(i in 1:length(lam))
  {
    if(spline == 0)
    {
      ylab = "CV"
      cv2 = kernel_cv(x, y, lam[i])
      cv3 = c(cv3, cv2)
    }
    else
    {
      if(spline == 1)
      {
        ylab = "CV"
        cv2 = spline_cv(x, y, lam[i])
        cv3 = c(cv3, cv2)
      }
      else
      {
        ylab = "GCV"
        cv2 = spline_gcv(x, y, lam[i])
        cv3 = c(cv3, cv2)
      }
    }


  }
  plot(lam, cv3, xlab = "Lambda", ylab = ylab, main = "Lambda vs cross
validation", pch = 20, col = "green")
  lam_min = lam[which.min(cv3)]
  return(lam_min)
}
```

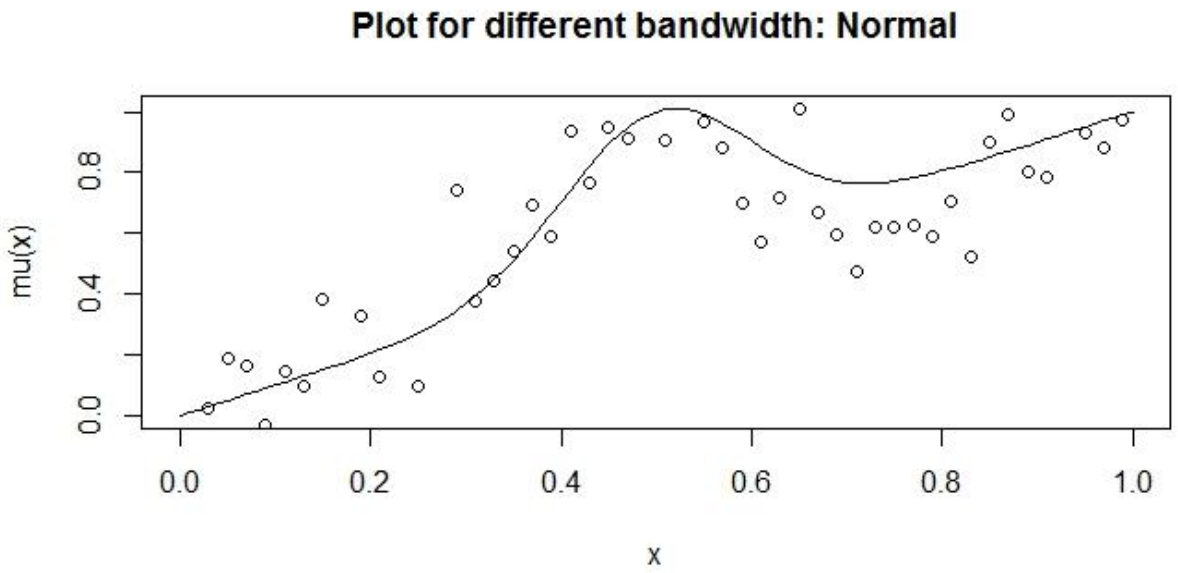
```
plot_ksmooth = function(x, y, mu, method)
{
  lam = c(0.1, 0.3, 0.5, 0.7, 0.9)
  col = c("red", "yellow", "green", "blue", "orange")
  titl = paste("Plot for different bandwidth:", method)
  curve(mu(x), 0, 1, main = titl)
  points(x, y)
  for(i in 1:length(lam))
  {
    fit = ksmooth(x, y, kernel = method, bandwidth = lam[i])
    lines(fit$x, fit$y, lty = 4, col = col[i])

  }
  legend('bottomright', legend = lam , lty=1, col= col, bty='n', cex=.75)
}

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)

#plot(x, y, col = "purple")
mu = function(t){t + 0.5 *exp(-50*(t-0.5)^2)}
u = expression(t + 0.5 *exp(-50*(t-0.5)^2))
curve(mu(x), 0, 1)
points(x, y)

# part b
```

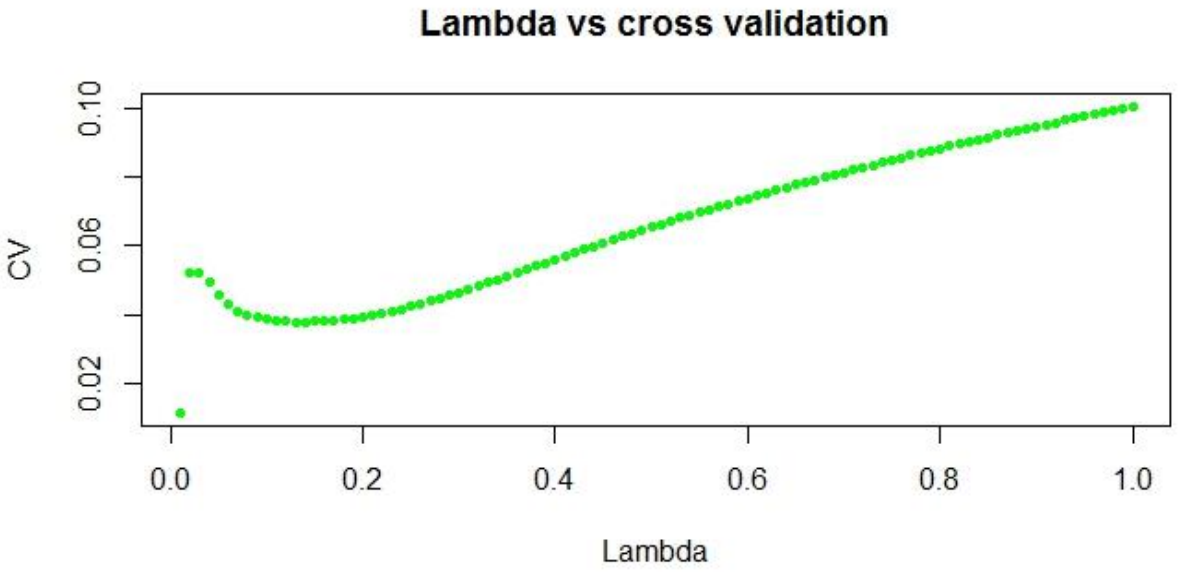
```
plot_ksmooth(x, y, mu, "Box")
plot_ksmooth(x, y, mu, "Normal")

# part c
lam_min = plot_cv(x, y, spline = 0)
ans = paste("The Optimized Lambda Value is", lam_min)
print(ans)

# part e

plot_spline = function(x, y, mu)
{
  lam = c(0.1, 0.3, 0.5, 0.7, 0.9)
  col = c("red", "yellow", "green", "blue", "orange")
  curve(mu(x), 0, 1, ylim = range(-0.6, 1.5), main = "Smoothing Spline")
  points(x, y)
  for(i in 1:length(lam))
  {
    fit = smooth.spline(x, y, spar = lam[i])
    lines(fit$x, fit$y, lty = 4, col = col[i])
  }
  legend('bottomright', legend = lam , lty=1, col= col, bty='n', cex=.75)
}

plot_spline(x, y, mu)
min_lam = plot_cv(x, y, spline = 1)
min_gcv = plot_cv(x, y, spline = 2)
```

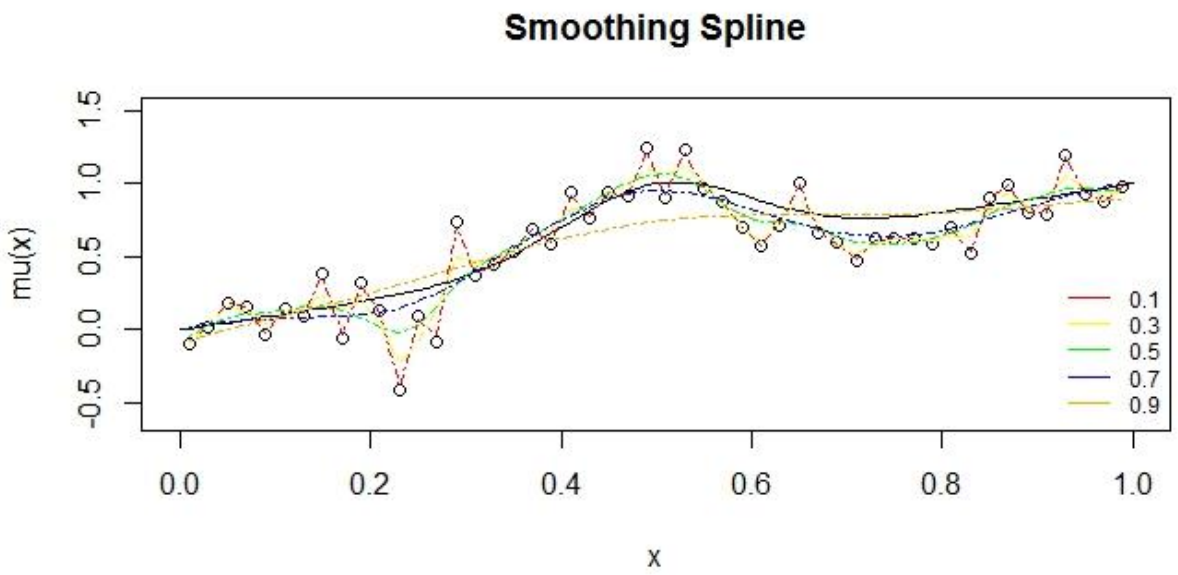


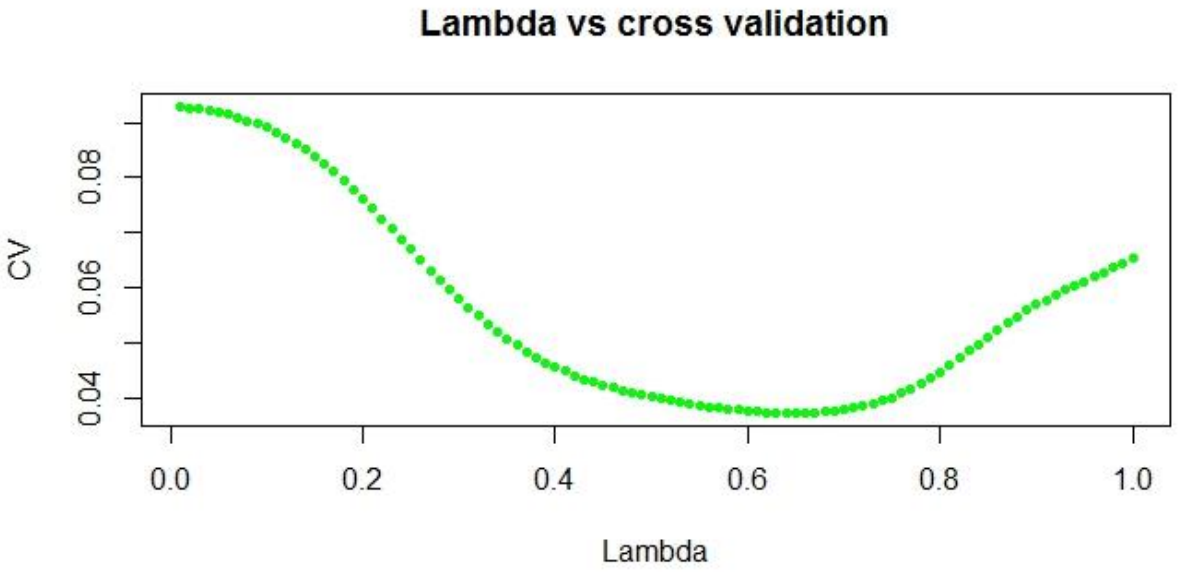
Plot for different bandwidth: Box

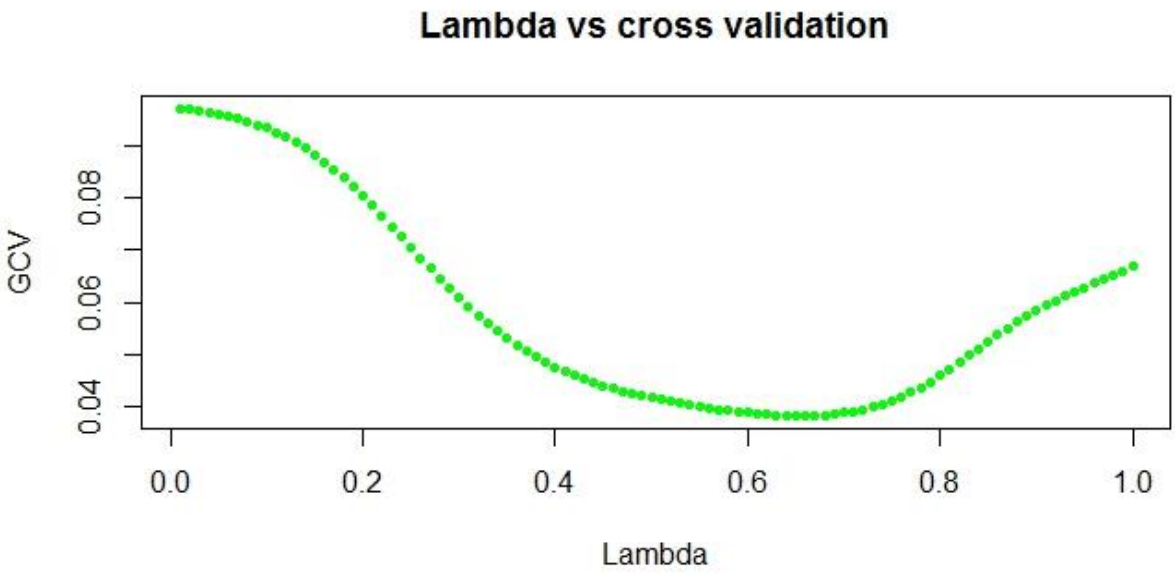



Plot for different bandwidth: Normal
mu(x)
0.0
0.4
0.8
0.0
0.2
0.4
0.6
0.8
1.0
x


Lambda vs cross validation
CV
0.02
0.06
0.10
0.0
0.2
0.4
0.6
0.8
1.0
Lambda


Smoothing Spline
mu(x)
-0.5
0.0
0.5
1.0
1.5
0.0
0.2
0.4
0.6
0.8
1.0
x
0.1
0.3
0.5
0.7
0.9


Lambda vs cross validation
CV
0.04
0.06
0.08
0.0
0.2
0.4
0.6
0.8
1.0
Lambda


Lambda vs cross validation
GCV
0.04
0.06
0.08
0.0
0.2
0.4
0.6
0.8
1.0
Lambda