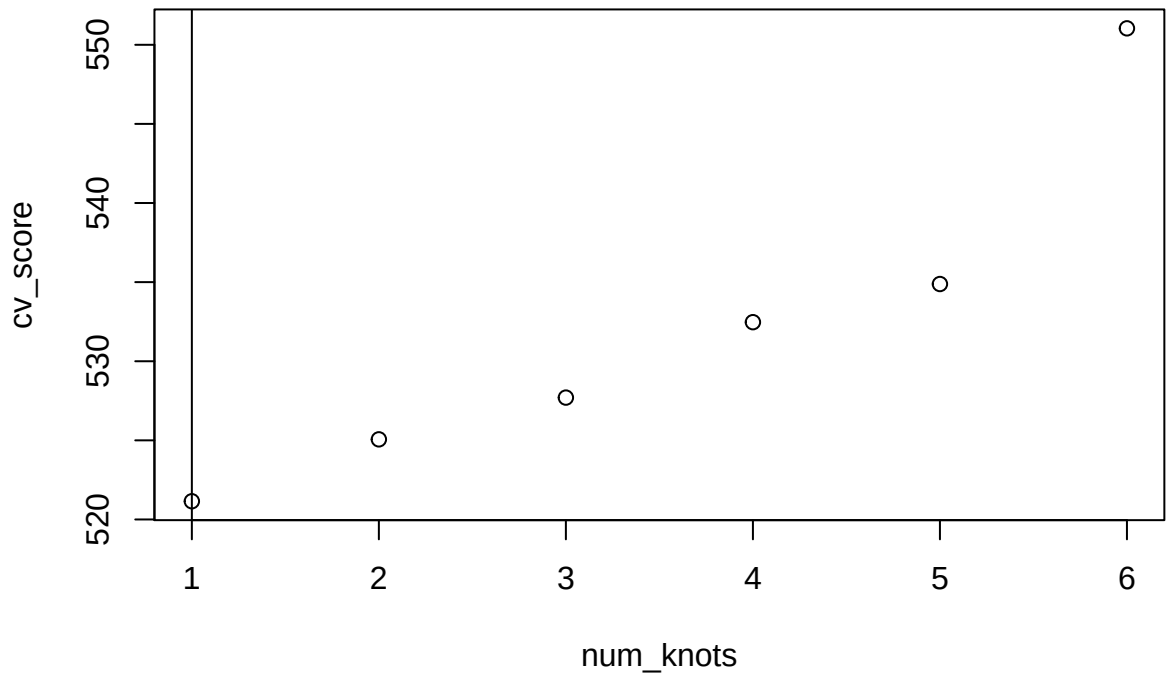
a3

```
# knitr::opts_chunk$set(echo = TRUE, cache = FALSE)
options(warn=-1)
```

1(a)

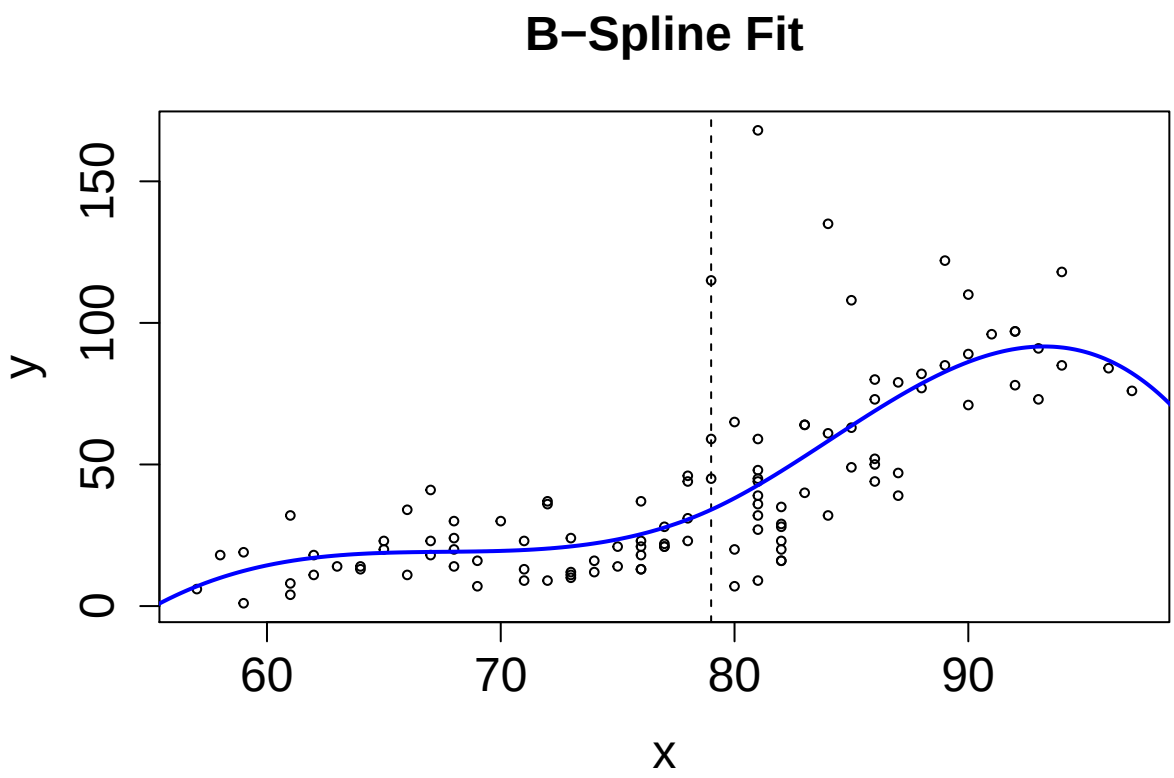
```
data <- read.table("ozone.txt", header=T)
data = data.frame(x=data$temperature, y=data$ozone)
library(boot)
library(splines)

index=order(data$x)
x=data$x[index]
y=data$y[index]

k_knots <- function(k, x){
  locs = seq(1/(k+1), 1-1/(k+1), 1/(k+1))
  quantile(x, locs)
}

num_knots = c(1:6)
cv_score = rep(0,6)
for (i in num_knots){
  knots_p=k_knots(i, x)
  fit.bs <- glm(y ~ bs(x, degree= 3, knots=knots_p),data = data)
  set.seed(444)
  CV=cv.glm(data, fit.bs, K=10)
  cv_score[i]=CV$delta[1]

}
plot(num_knots, cv_score)

k=num_knots[which.min(cv_score)]
abline(v=k)
```




```
k
```

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

```
knots_p=k_knots(k, data$x)
fit.bs_a <- glm(y ~ bs(x, degree= 3, knots=knots_p), data = data)
xrange <- extendrange(x)
xnew <- seq(min(xrange), max(xrange), length.out=500)
ypred.bs <- predict(fit.bs_a, newdata= data.frame(x=xnew))

plot(x,y,
     pch=1, cex=0.6,
     main = "B-Spline Fit",
     cex.axis=1.5 , cex.main=1.5, cex.lab=1.5,
)
abline(v=knots_p , lty=2 )
lines(xnew, ypred.bs, col="blue", lwd=2)
```

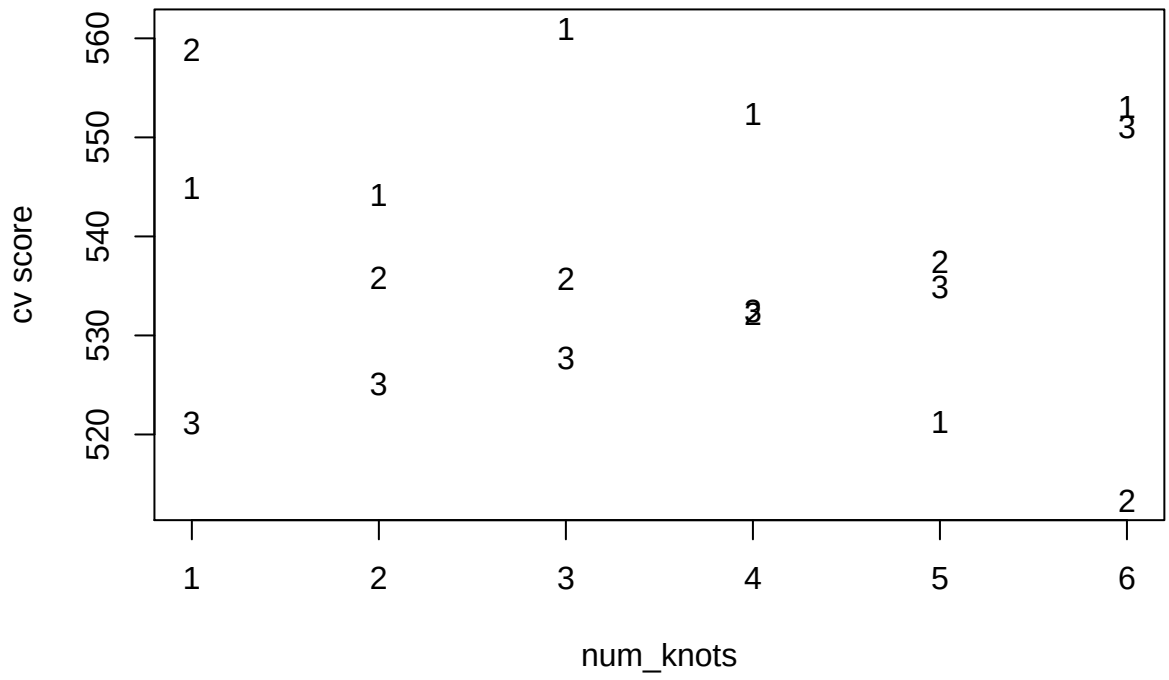

B−Spline Fit
y
0
50
100
150
60
70
80
90
x

1(b)

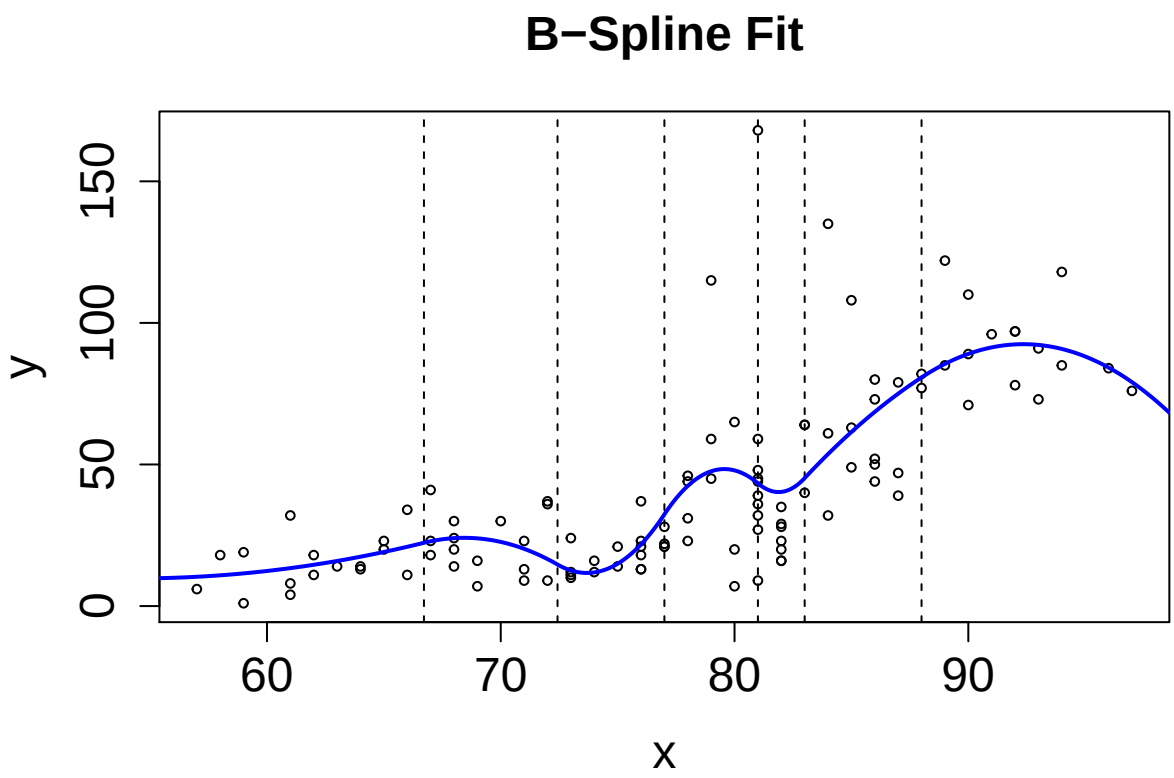
```
cvs = matrix(1:18,nrow=3,ncol=6)
num_knots = c(1:6)
cv_score = rep(0,6)
degrees = c(1:3)
for (i in num_knots)
  {
    for (j in degrees)
    {
      knots_p=k_knots(i, x)
      set.seed(444)
      fit.bs <- glm(y ~ bs(x, degree= j, knots=knots_p),data = data)
      CV=cv.glm(data, fit.bs, K=10)
      cvs[j,i]=CV$delta[1]
    }
}
cvs
```

```
##          [,1]     [,2]     [,3]     [,4]     [,5]     [,6]
## [1,] 544.9239 544.1610 561.0100 552.3368 521.2414 553.0459
## [2,] 558.8231 535.7830 535.7246 532.1627 537.3980 513.2574
## [3,] 521.1519 525.0623 527.7018 532.4686 534.8796 551.0386
```

```
min_index= which(cvs == min(cvs), arr.ind = TRUE)
dmin = cvs[which(cvs == min(cvs), arr.ind = TRUE)]
dmax = cvs[which(cvs == max(cvs), arr.ind = TRUE)]
ytemp=c(rep(dmin,5),dmax)
plot(ytemp~num_knots, type='n' , ylab = 'cv score')
for (i in num_knots)
  {
    for (j in degrees)
    {
        text(i,cvs[j,i],label=j)
    }
}
```




```
degree = min_index[1]
knots = min_index[2]
knots_p=k_knots(knots, data$x)
fit.bs_b <- glm(y ~ bs(x, degree= degree, knots=knots_p), data = data)
xrange <- extendrange(x)
xnew <- seq(min(xrange), max(xrange), length.out=500)
ypred.bs <- predict(fit.bs_b, newdata= data.frame(x=xnew))

plot(x,y,
     pch=1, cex=0.6,
     main = "B-Spline Fit",
     cex.axis=1.5 , cex.main=1.5, cex.lab=1.5,
)
abline(v=knots_p , lty=2 )
lines(xnew, ypred.bs, col="blue", lwd=2)
```


B−Spline Fit
y
0
50
100
150
60
70
80
90
x

(c)

```
print('degree of freedom in a')
```

```
## [1] "degree of freedom in a"
```

```
fit.bs_a$df.null-fit.bs_a$df.residual
```

```
## [1] 4
```

```
print('degree of freedom in b')
```

```
## [1] "degree of freedom in b"
```

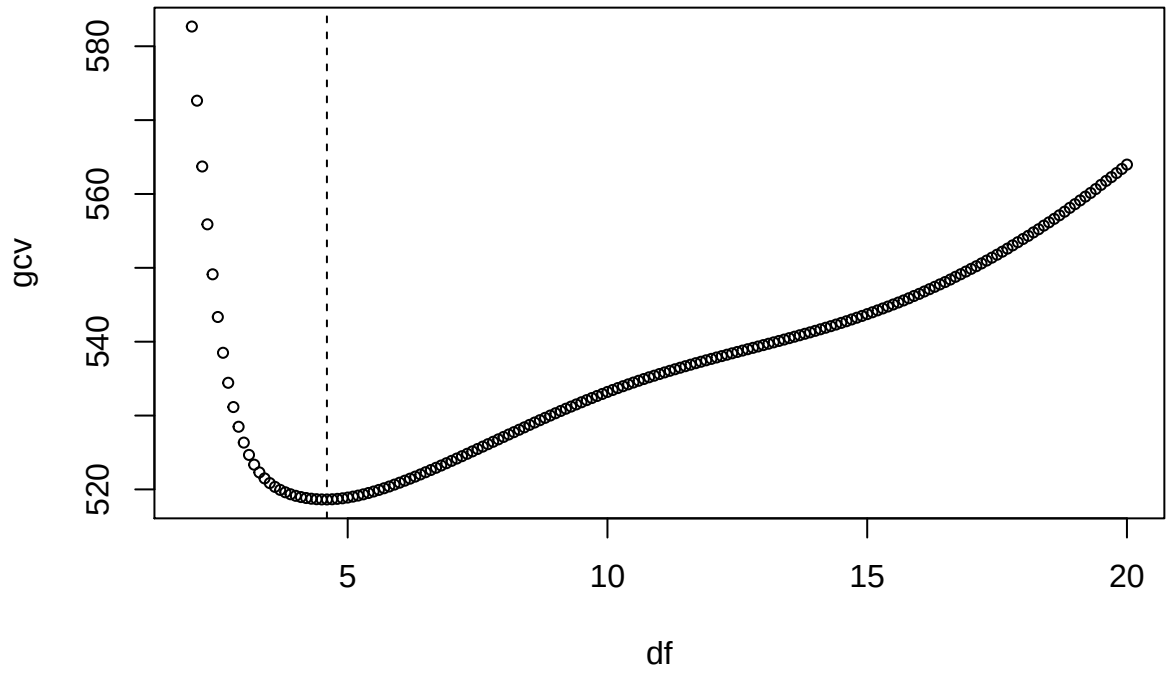
```
fit.bs_b$df.null-fit.bs_b$df.residual
```

```
## [1] 8
```

we observe part b has way more degrees of freedom than part a, hence the graph looks more flexible and more local. this make sense because c has way more knots than b, so local models for c are more “local”.

2(a)

```
df = seq(2,20,0.1)
gcv = c()
j=1
for (i in df){
  sm = smooth.spline(x,y,df=i,cv=FALSE)
  gcv[j]=sm$cv.crit
  j = j+1
}
plot(df, gcv, pch=1, cex=0.7)
best_df = df[which.min(gcv)]
abline(v=best_df , lty=2 )
```

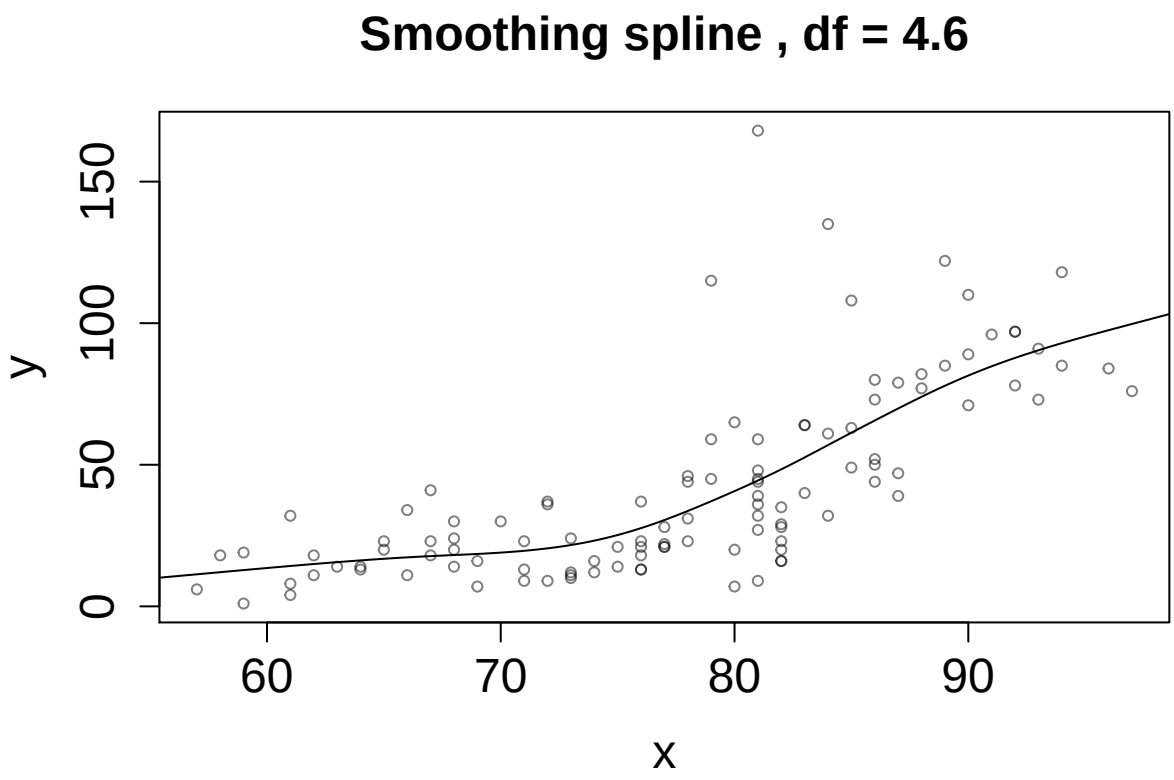



```
best_df
```

```
## [1] 4.6
```

```
sm <- smooth.spline(x, y, df = best_df)
#help("smooth.spline")
# plotting the resutls
xrange <- extendrange(x)
xnew <- seq(min(xrange), max(xrange), length.out=500)
ypred.sm <- predict(sm, x=xnew)$y # fitted values
plot(x,y,
     pch=1, cex=0.7,
     main = paste("Smoothing spline , df =", best_df),
     cex.axis=1.5 , cex.main=1.5, cex.lab=1.5,
```

```
     col=adjustcolor("black" , alpha=0.5)
)
lines(xnew, ypred.sm)
```

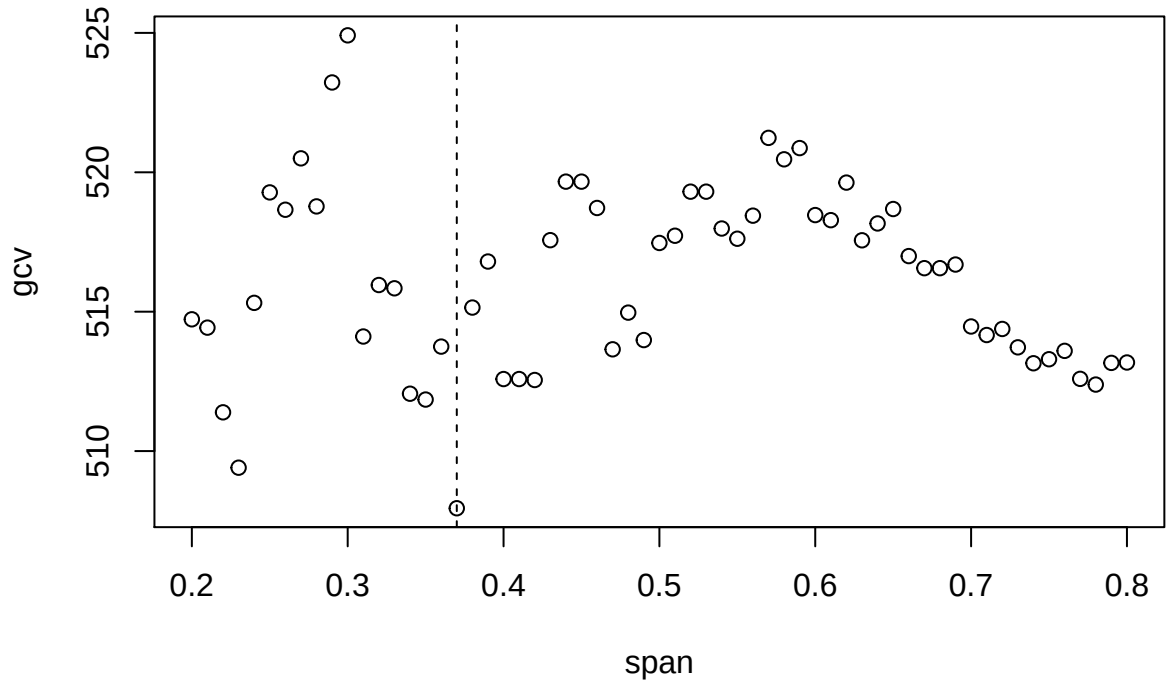


2(b)

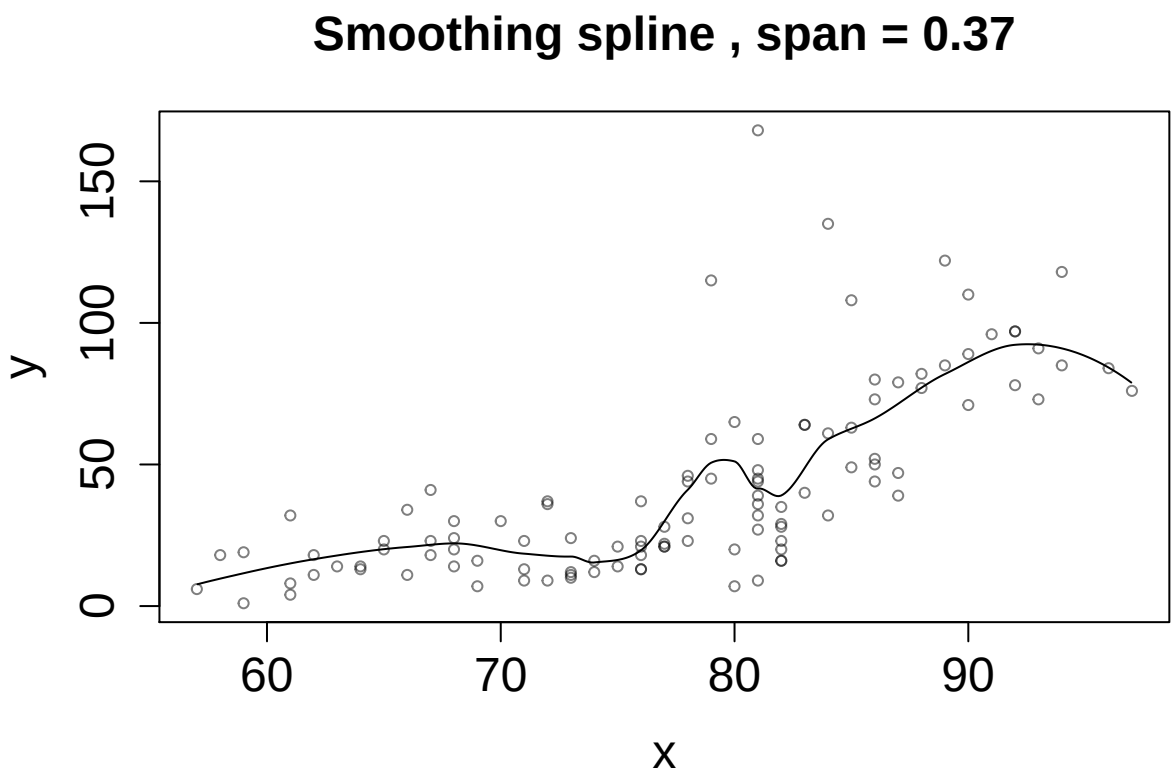
```
# A function to generate the indices of the k-fold sets

kfold <- function(N, k=N, indices=NULL){
  # get the parameters right:
  if (is.null(indices)) {
    # Randomize if the index order is not supplied
    indices <- sample(1:N, N, replace=FALSE)
  } else {
    # else if supplied, force N to match its length
    N <- length(indices)
  }
  # Check that the k value makes sense.
  if (k > N) stop("k must not exceed N")
  #

  # How big is each group?
  gsize <- rep(round(N/k), k)

  # For how many groups do we need odjust the size?
  extra <- N - sum(gsize)

  # Do we have too few in some groups?
  if (extra > 0) {
    for (i in 1:extra) {
      gsize[i] <- gsize[i] +1
    }
  }
  # Or do we have too many in some groups?
  if (extra < 0) {
    for (i in 1:abs(extra)) {
      gsize[i] <- gsize[i] - 1
    }
  }

  running_total <- c(0,cumsum(gsize))

  # Return the list of k groups of indices
  lapply(1:k,
         FUN=function(i) {
           indices[seq(from = 1 + running_total[i],
                       to = running_total[i+1],
                       by = 1)
                   ]
         }
  )
}


# A function to form the k samples
getKfoldSamples <- function (x, y, k, indices=NULL){
  groups <- kfold(length(x), k, indices)
  Ssamples <- lapply(groups,
```

```
                    FUN=function(group) {
                      list(x=x[-group], y=y[-group])
                    })
  Tsamples <- lapply(groups,
                    FUN=function(group) {
                      list(x=x[group], y=y[group])
                    })
  list(Ssamples = Ssamples, Tsamples = Tsamples)
}


get_gcv_score <- function(x,y,span,degree){
  set.seed(444)
  samples_kfold <-  getKfoldSamples(x, y, k=length(x))
  Ssamples <- samples_kfold$Ssamples # change this accorcing to the number of folds
  Tsamples <- samples_kfold$Tsamples # change this accorcing to the number of folds
  MSE = c()
  # help(loess)

  for(j in 1:length(Ssamples)){
    x.temp = Ssamples[[j]]$x
    y.temp = Ssamples[[j]]$y
    temp=data.frame(x_h=x.temp, y_h=y.temp)
    model = loess(y_h~x_h, span=span, degree=degree, data=temp)

    pred = predict(model , newdata=data.frame(x_h=Tsamples[[j]]$x))
    MSE[j] = mean((Tsamples[[j]]$y-pred)^2)
  }
  MSE[MSE=="N/A"] = NA
  MSE=na.omit(MSE)
  mean(MSE)
}
```

```
span = seq(0.2,0.8,0.01)
gcv = c()
j=1

for (i in span){
  gcv_score=get_gcv_score(x,y,i,2)
  # print(gcv_score)
  gcv[j]=gcv_score
  j=j+1
}

best_span = span[which.min(gcv)]
model_b = loess(y~x, span=best_span, degree=2, data=data)
plot(span, gcv)
abline(v=best_span , lty=2 )
```

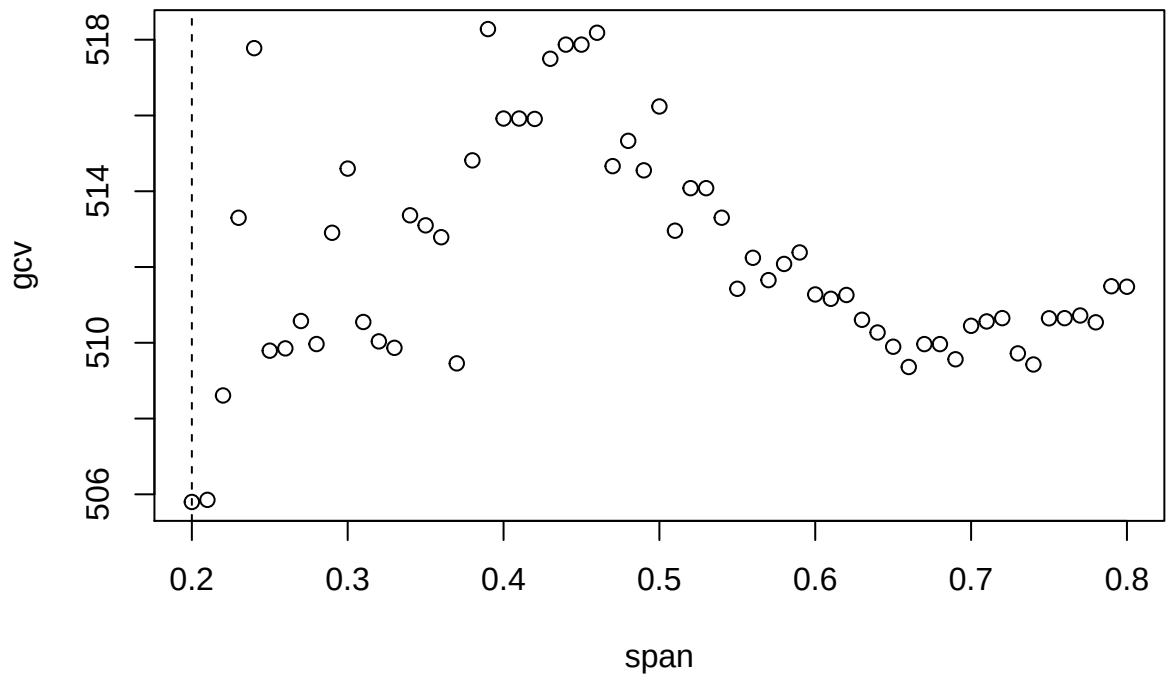



```
range <- extendrange(x)
xnew <- seq(min(xrange), max(xrange), length.out=500)
ypred.sm <- predict(model_b, data.frame(x=xnew)) # fitted values
plot(x,y,
     pch=1, cex=0.7,
     main = paste("Smoothing spline , span =", best_span),
     cex.axis=1.5 , cex.main=1.5, cex.lab=1.5,
     col=adjustcolor("black" , alpha=0.5)
)
lines(xnew, ypred.sm)
```


Smoothing spline , span = 0.37
y
0
50
100
150
60
70
80
90
x

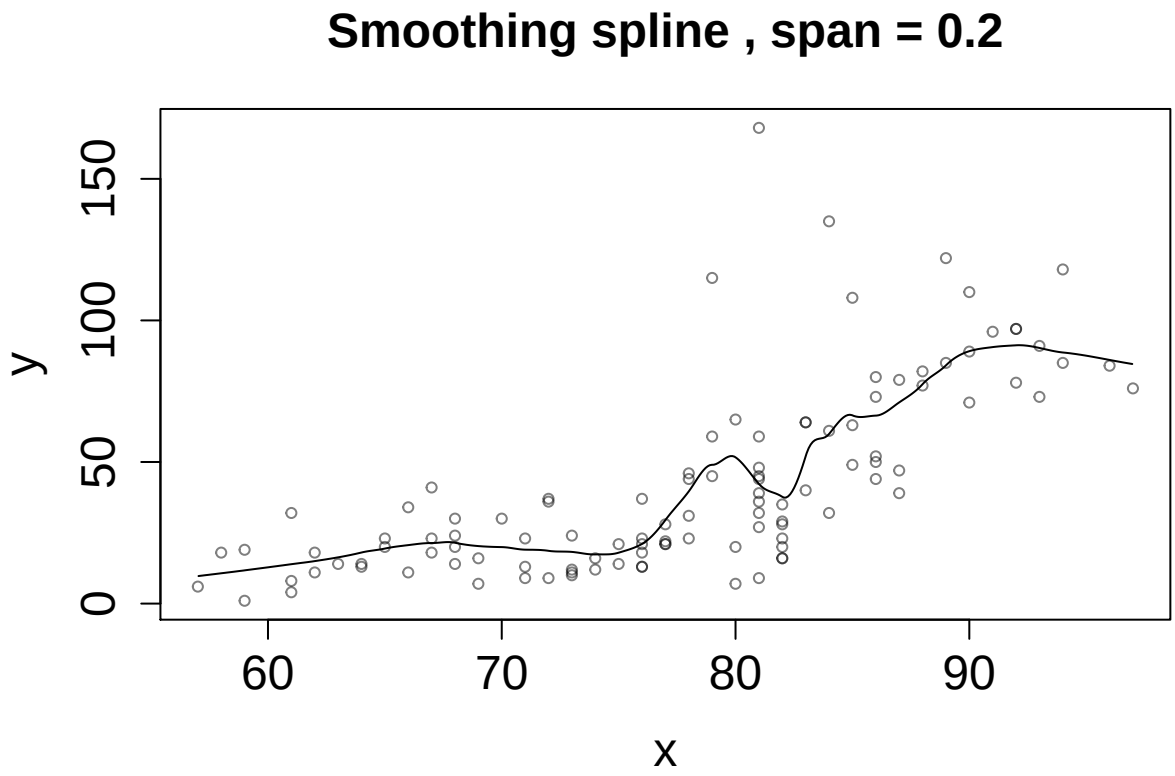
2(c)

```
span = seq(0.2,0.8,0.01)
gcv = c()
j=1

for (i in span){
  gcv_score=get_gcv_score(x,y,i,1)
  # print(gcv_score)
  gcv[j]=gcv_score
  j=j+1
}

best_span = span[which.min(gcv)]
model_c = loess(y~x, span=best_span, degree=1, data=data)
plot(span, gcv)
abline(v=best_span , lty=2 )
```




```
range <- extendrange(x)
xnew <- seq(min(xrange), max(xrange), length.out=500)
ypred.sm <- predict(model_c, data.frame(x=xnew)) # fitted values
plot(x,y,
     pch=1, cex=0.7,
     main = paste("Smoothing spline , span =", best_span),
     cex.axis=1.5 , cex.main=1.5, cex.lab=1.5,
     col=adjustcolor("black" , alpha=0.5)
)
```

```
lines(xnew, ypred.sm)
```



2(d)

```
print('df in part a')
```

```
## [1] "df in part a"
```

```
best_df
```

```
## [1] 4.6
```

```
print('df in part b')
```

```
## [1] "df in part b"
```

```
model_b$enp
```

```
## [1] 9.083582
```

```
print('df in part c')
```

```
## [1] "df in part c"
```

```
model_c$enp
```

```
## [1] 8.691655
```

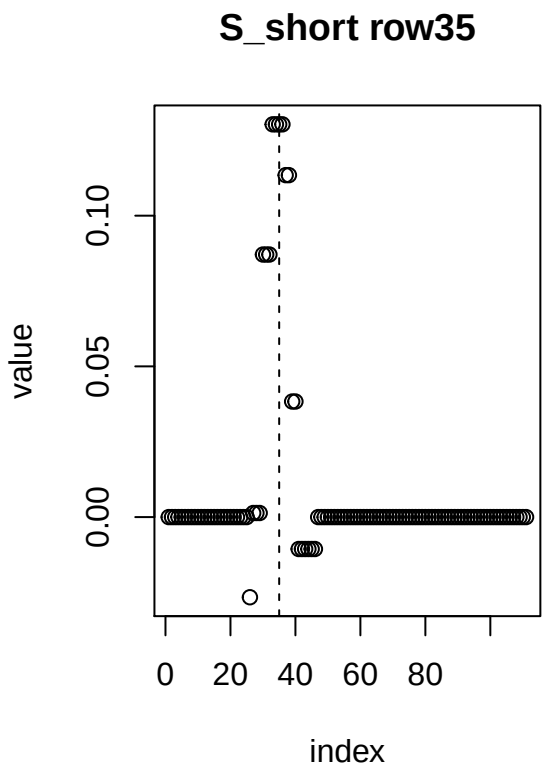
```
help(loess)
```

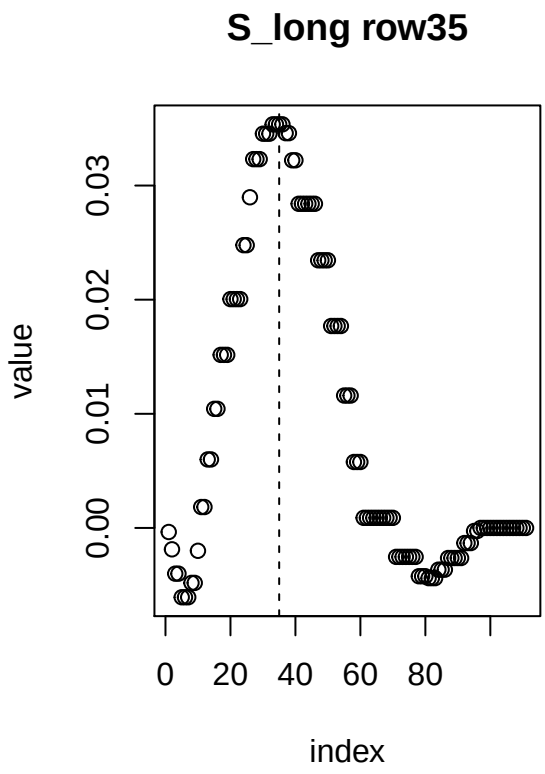
```
## starting httpd help server ... done
```

we observe df in a is significantly smaller than b and c. this is because smoothing splines is punishing the flexibility of the data whereas b and c do not punish flexibility as long as there is no overfitting. For c the span is smaller, so the model is more local than c, this result in the flexibility of the model increase, this is a bonus to the df. However c has less degree than b, thus less parameters, results in smaller df due to number of parameters. thus b and c have roughly the same df.

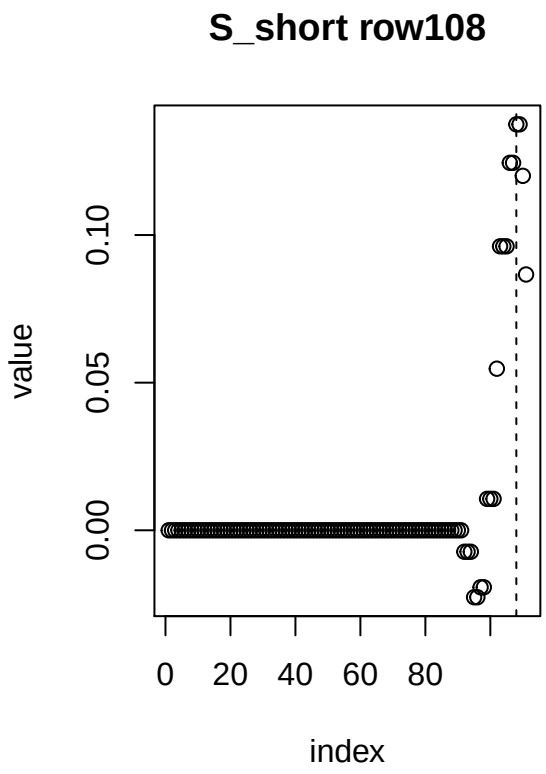
3(a)

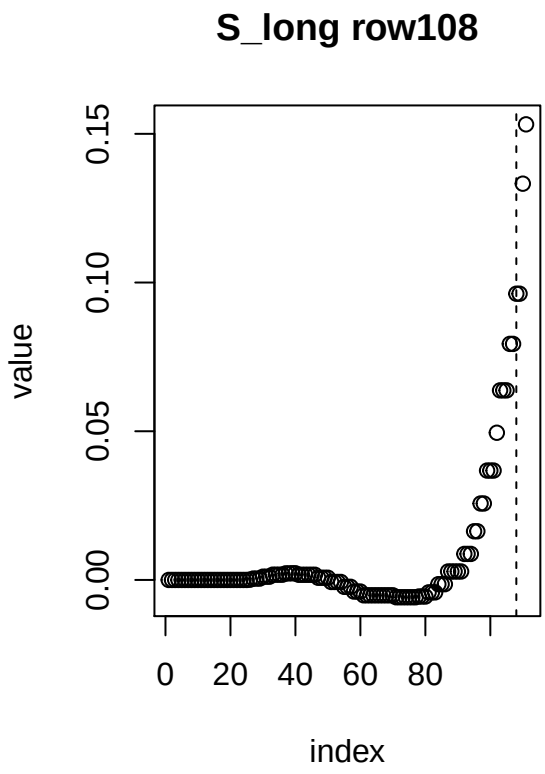
```
help("smooth.spline")

par(mfrow=c(1,2))
smootherMatrix <- function(x,span) {
  n <- length(x)
  S <- matrix(0, n, n)
  for (i in 1:n) {
    ei = rep(0, n)
    ei[i] <- 1
    # insert the fit into the i'th column of S
    temp = data.frame(x=x, y=ei)
    model = loess(y~x, span=span, data = temp)
    S[,i]  <- predict(model, data.frame(x=x))
  }
  # To make sure the result is (numerically) symmetric
  #S <- (S + t(S))/2
  # and return the symmetric matrix
  S
}
S=smootherMatrix(x,0.2)
S_2=smootherMatrix(x,0.8)
r35=S[35,]
plot(r35, xlab = 'index', ylab='value', main='S_short row35')
abline(v=35 , lty=2 )

plot(S_2[35,], xlab = 'index', ylab='value', main='S_long row35')
abline(v=35 , lty=2 )
```



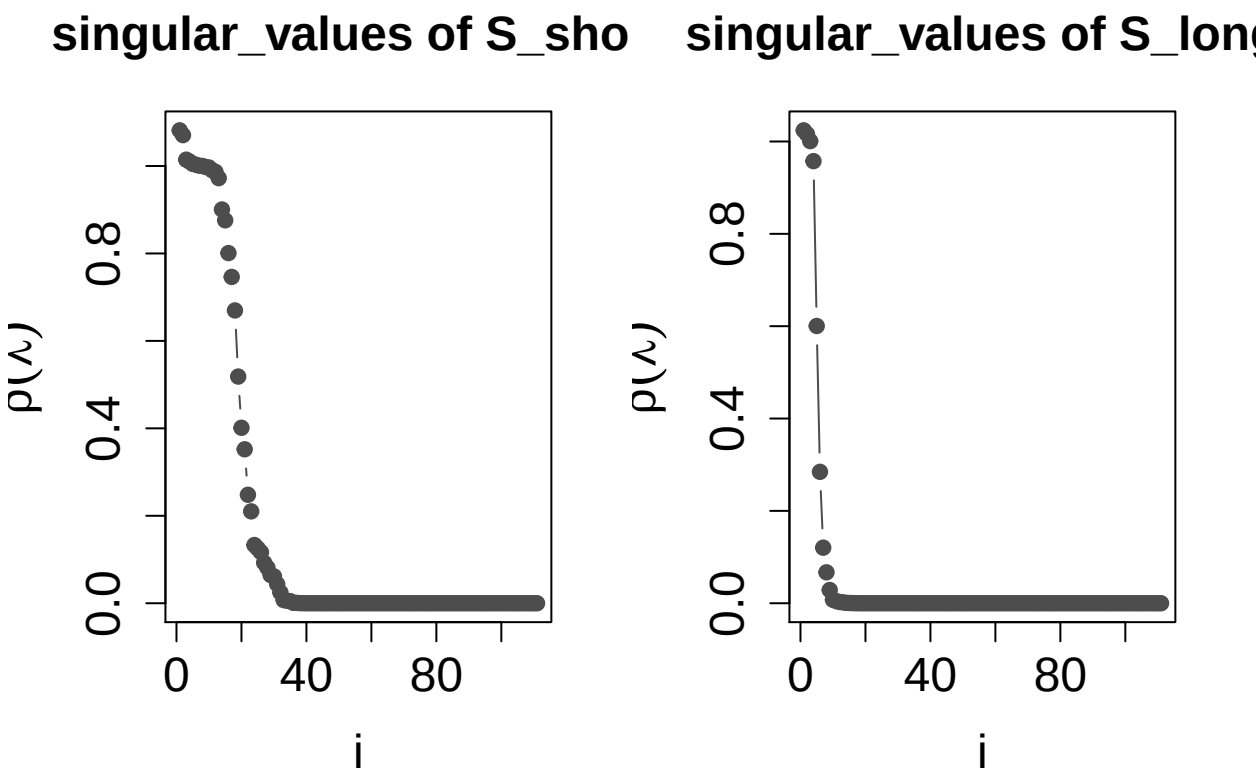



```
r108 = S[108,]
plot(r108, xlab = 'index', ylab='value', main='S_short row108')
abline(v=108 , lty=2 )

plot(S_2[108,], xlab = 'index', ylab='value', main='S_long row108')
abline(v=108 , lty=2 )
```

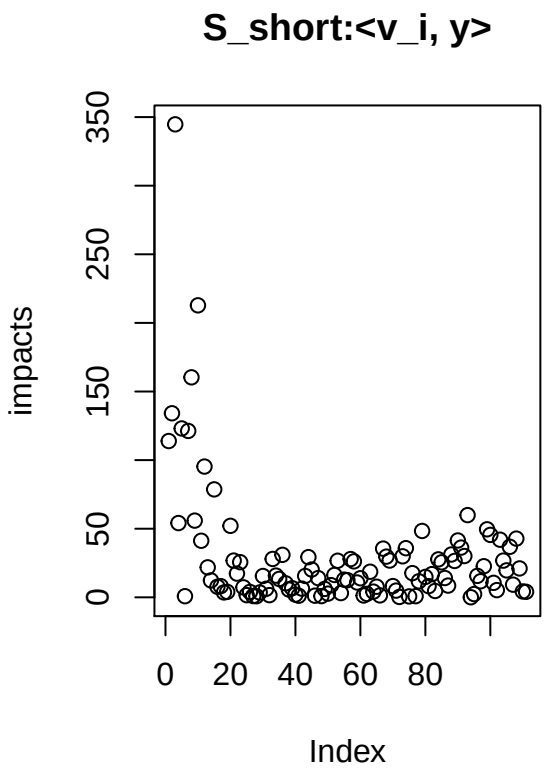


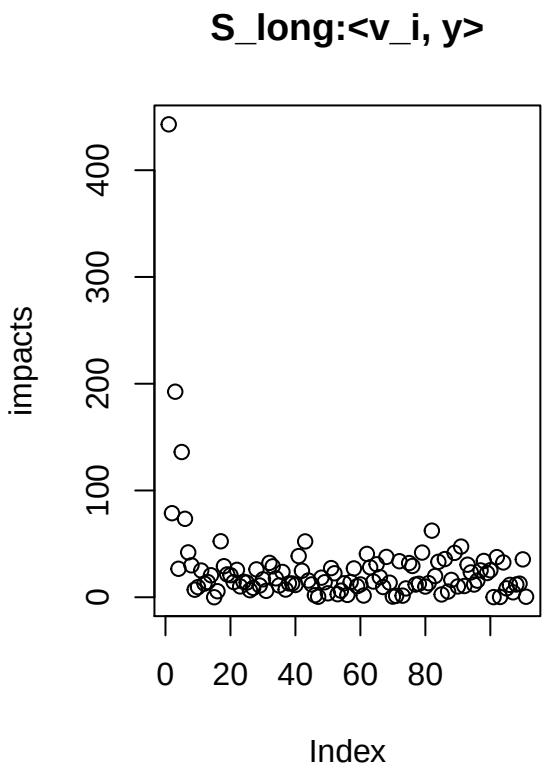



```
sing= svd(S)
sing_2= svd(S_2)
plot(sing$d, type="b",
     main=paste("singular_values of S_short"),
     ylab = expression(rho(lambda)),
     xlab="i",
     pch=19, col="grey30",
     cex.axis=1.5 , cex.main=1.5, cex.lab=1.5)
plot(sing_2$d, type="b",
     main=paste("singular_values of S_long"),
     ylab = expression(rho(lambda)),
     xlab="i",
     pch=19, col="grey30",
     cex.axis=1.5 , cex.main=1.5, cex.lab=1.5)
```

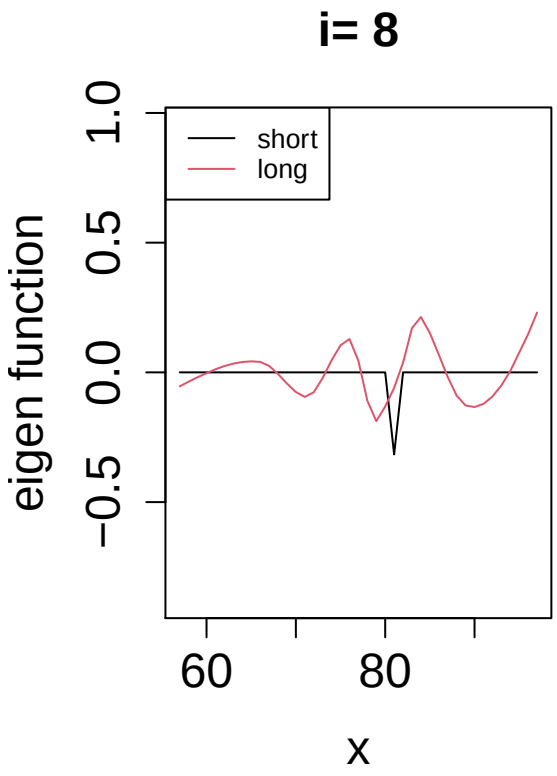


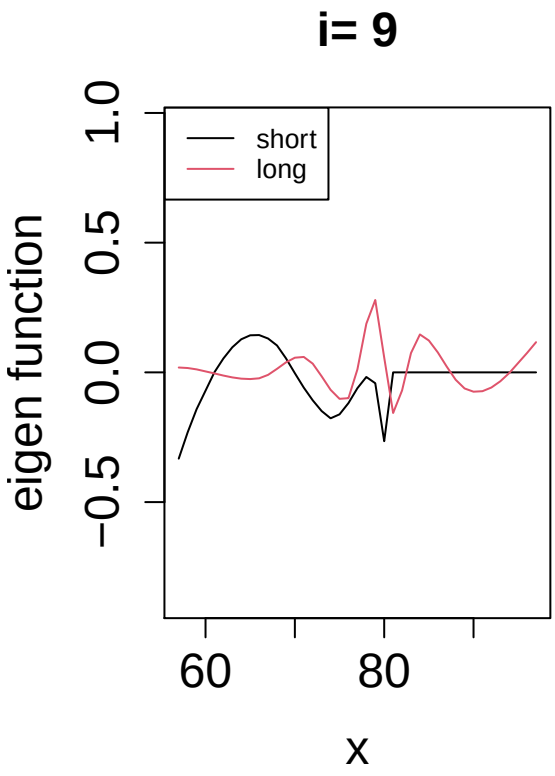

```
impacts = c()
j=1
for(i in 1:length(y)){
  impacts[j]= abs(t(sing$v[,i]) %*% y)
  j=j+1
}
plot(impacts, main='S_short:<v_i, y>')

impacts = c()
j=1
for(i in 1:length(y)){
  impacts[j]= abs(t(sing_2$v[,i]) %*% y)
  j=j+1
}
plot(impacts,main='S_long:<v_i, y>')
```

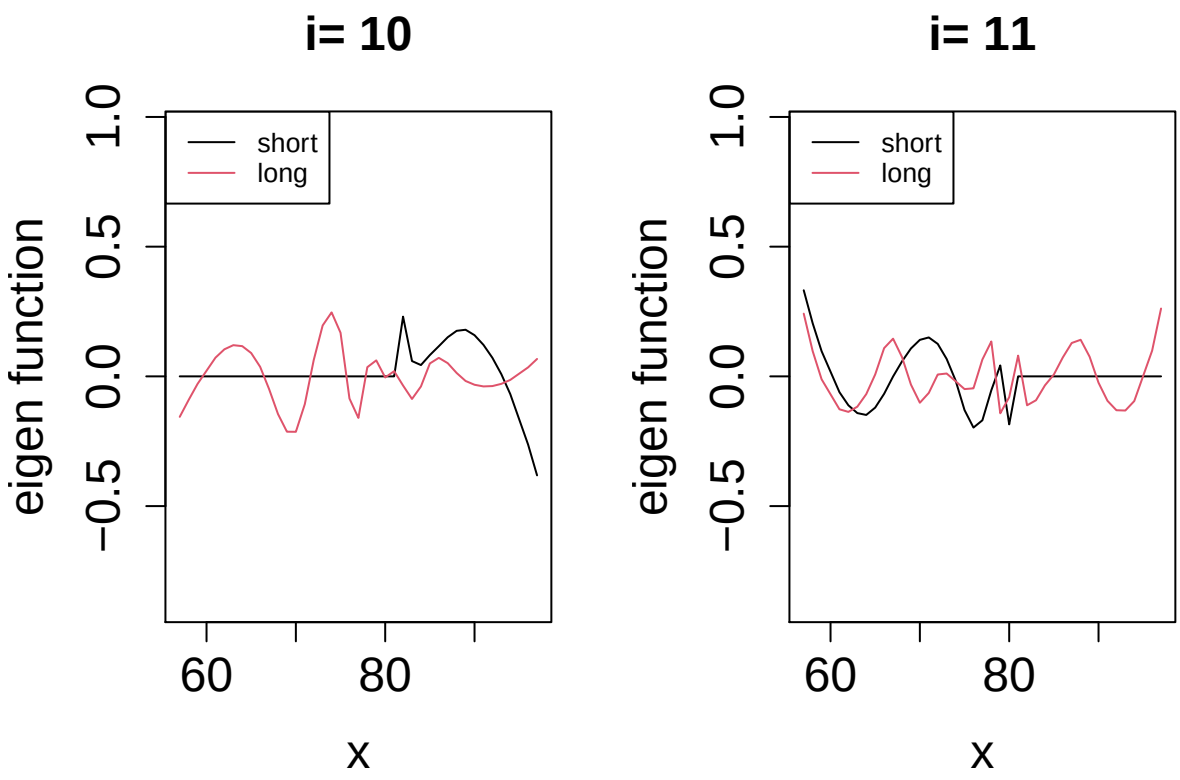


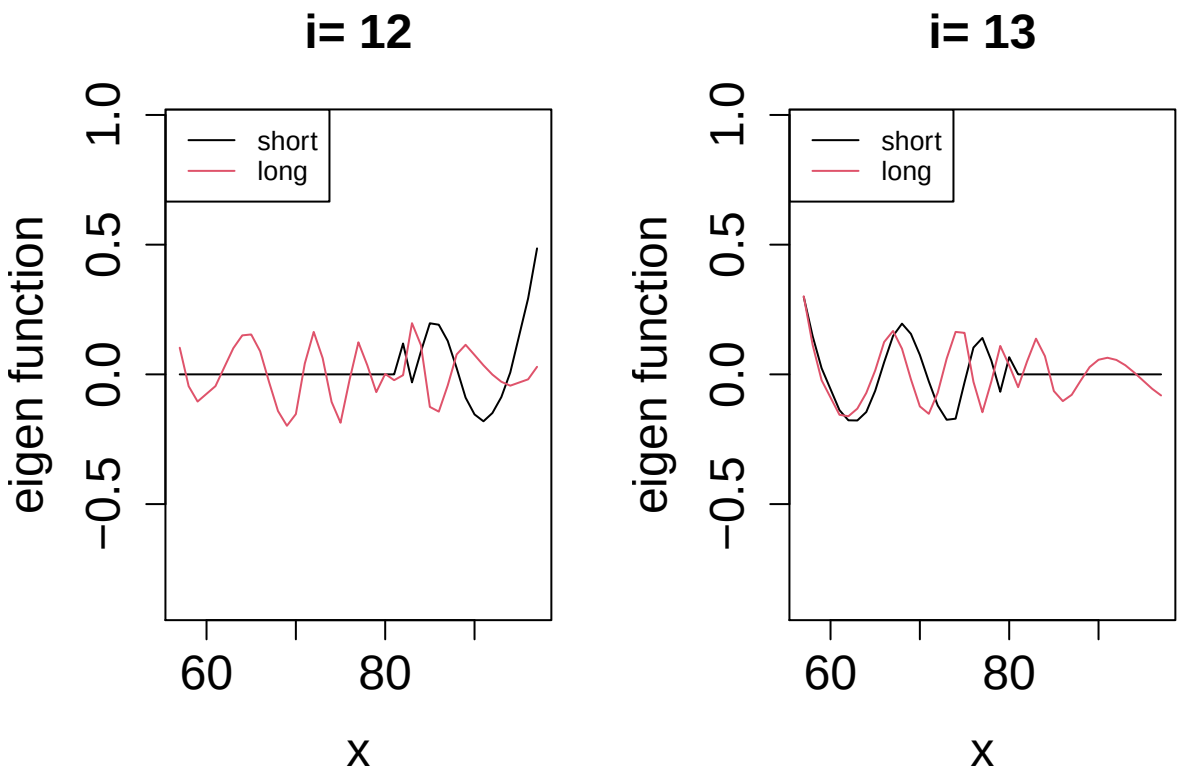


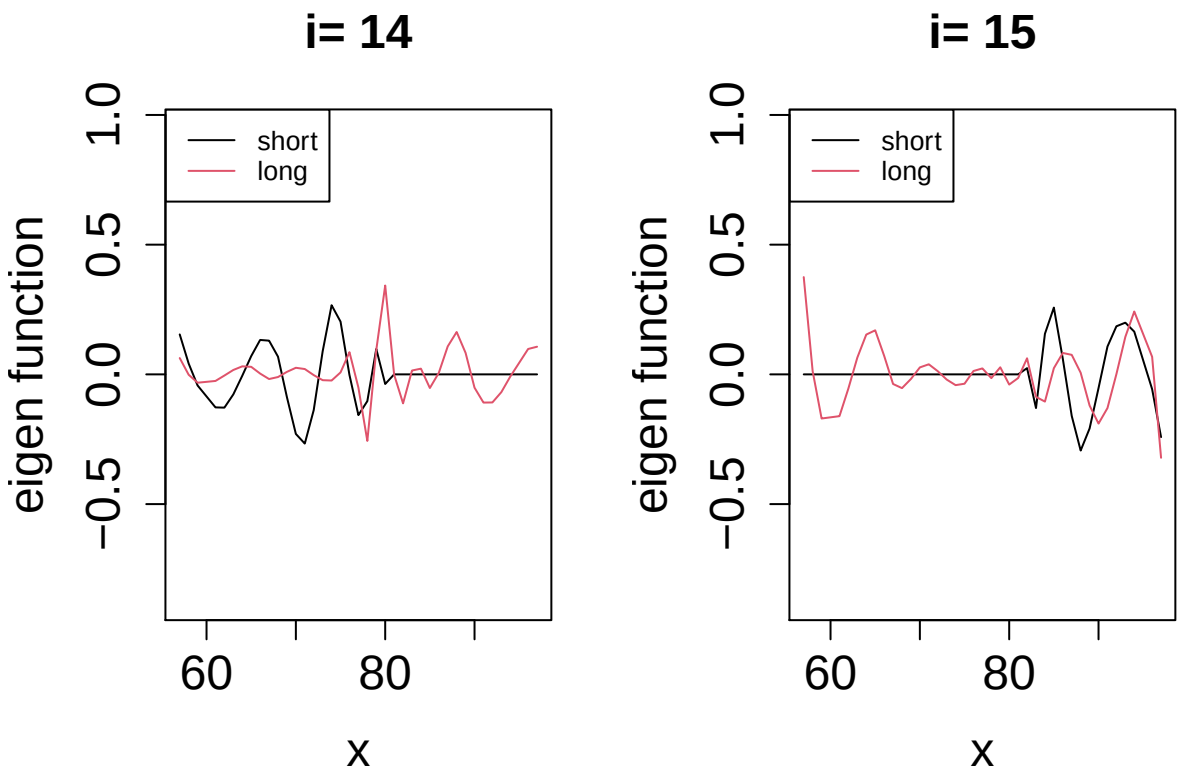

```
plotEigenBases <- function(x,
                           eigenDecomp, eigenDecomp2,
                           indices){

  Xorder <- order(x)
  orderedX <- x[Xorder]
  vectors <- eigenDecomp$u
  vectors2 <- eigenDecomp2$u
  ylim <- range(vectors)
  values <- eigenDecomp$d
  values2 <- eigenDecomp2$d
  for (i in indices){
    plot(orderedX, vectors[Xorder, i],
         type="l", col=1,
         xlab="x", ylab="eigen function",
         ylim = ylim,
         main=paste("i=", i),
         cex.axis=1.5 , cex.main=1.5, cex.lab=1.5)
    lines(orderedX, vectors2[Xorder, i], col=2)
    legend("topleft",legend=c("short", "long"),col=c(1, 2), lty=c(1,1), cex=0.8)
  }


}

plotEigenBases(x,sing,sing_2, c(8:16))
```


i= 8
short
long
eigen function
1.0
0.5
0.0
−0.5
60
80
x


i= 9
short
long
eigen function
1.0
0.5
0.0
−0.5
60
80
x

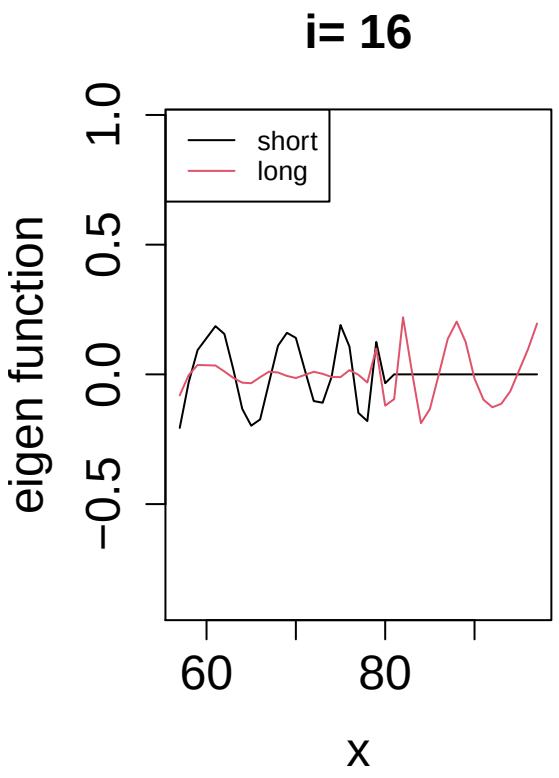

i= 10
i= 11
eigen function
x
short
long
1.0
0.5
0.0
−0.5
60
80


i= 12
i= 13
short
long
eigen function
x
60
80
−0.5
0.0
0.5
1.0




```
dim(S)
```

```
## [1] 111 111
```


i= 16
short
long
eigen function
1.0
0.5
0.0
−0.5
60
80
x

(b) for i), we observe as expected, as we go to S_{ii}, the value becomes larger, S_short has a sharper increase than S_long. Also, for i=108, the peak has only one end for S_long.
for ii), we observe singular value of S_short goes to 0 quicker than S_long, this is because if the model become more global, the pattern is more consistent, whereas each local model differs from one another significantly. for iii) we observe more basic vectors are impacted (by considerable amount) by y for S_short, this is because different local model of S_short impacts defferent basic vectors. We also observe the maxium impact on basic vector of S_long is significantly higher than S_short.

(c) we observe for S_short, there is a lot of x that does not has direction on the basis, this is independent of index of basic function. But as index increase, for x that does has directions, the direction becomes larger. For S_long, some x has slightly noise than the rest, but overall, the pattern is consistent throughout x axis. As index increase, there is more noise on the directions. S_long is giving higher weight to these directions because there is direction on all value of x.

4(a)

by partial derivative: $\int_a^b \mu^{''}(x)h^{''}(x)dx = \mu^{''}(x)h^{'}(x)|_a^b - \int_a^b \mu^{'''}(x)h^{'}(x)dx$

because $\mu^{''}(b) = \mu^{''}(a) = 0$ (linear at ends)
$\mu^{''}(x)h^{'}(x)|_a^b = 0 * h(b) - 0 * h(a) = 0$
$\int_a^b \mu^{''}(x)h^{''}(x)dx = -\int_a^b \mu^{'''}(x)h^{'}(x)dx$
$= -\left(\sum_{i=1}^{n-1}(\int_{x_i}^{x_{i+1}} \mu^{'''}(x)h^{'}(x)dx) + \int_a^{x_1} \mu^{'''}(x)h^{'}(x)dx + \int_{x_n}^b \mu^{'''}(x)h^{'}(x)dx\right)$

for the last two terms, $\mu^{'''}(x) = 0$ (linear) so we are left with $\int_a^b \mu^{''}(x)h^{''}(x)dx = -\left(\sum_{i=1}^{n-1}(\int_{x_i}^{x_{i+1}} \mu^{'''}(x)h^{'}(x))dx\right)$
as required

(b) for $x \in [a, x_1]$ and $x \in [x_n, b]$, because linear, $\mu^{'''}(x) = 0$ because cubic spline is $f_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i$, third deriative is $6a_i$(constant) at (x_i, x_{i+1}). because third derivative is not nessecarily same at boundary of knots, they are discontinuous at knots. so for $x \in (x_1, x_2 ... x_n)$, $\mu^{'''}(x)$ doesnot exist for $x \in (x_i, x_{i+1})$, $\mu^{'''}(x) = 6a_i$

(c) explained in (b), $\mu^{'''}(x)$ is $6a_i$ within (x_i, x_{i+1}), so $\int_{x_i}^{x_{i+1}} \mu^{'''}(x)h^{'}(x)dx = \int_{x_i}^{x_{i+1}} 6a_i h^{'}(x)dx = 6a_i \int_{x_i}^{x_{i+1}} h^{'}(x)dx = 6a_i(h(x_{i+1}) - h(x_i))$

(d) from (a),

$\int_a^b \mu^{''}(x)h^{''}(x)dx$
$= -\left(\sum_{i=1}^{n-1}(\int_{x_i}^{x_{i+1}} \mu^{'''}(x)h^{'}(x))dx\right)$
$= \sum_{i=1}^{n-1} c_i(h(x_{i+1}) - h(x_i))$ because$h(x) = g(x) - \mu(x)$ and $g(x_i) = \mu(x_i)$. we have
$\int_a^b \mu^{''}(x)h^{''}(x)dx = \sum_{i=1}^{n-1} c_i(h(x_{i+1}) - h(x_i)) = \sum c_i(0-0) = 0$

(e) first we prove the inequality. from(d),$\int_a^b \mu^{''}(x)h^{''}(x)dx = 0$
$\int_a^b \mu^{''}(x)(g^{''}(x) - \mu^{''}(x))dx = 0$
$\int_a^b \mu^{''}(x)g^{''}(x)dx = \int_a^b(\mu^{''}(x))^2dx$

consider $\int_a^b(g^{''}(x))^2dx - \int_a^b(\mu^{''}(x))^2dx$
$= \int_a^b(g^{''}(x))^2dx - (\mu^{''}(x))^2dx$
$= \int_a^b[(g^{''}(x))^2 - (\mu^{''}(x))(g^{''}(x))]dx$
$= \int_a^b(g^{''}(x)(g^{''}(x) - \mu^{''}(x))dx$
$= \int_a^b(g^{''}(x)h^{''}(x))dx$
$= \int_a^b(\mu^{''}(x) + h^{''}(x))h^{''}(x)dx$
$= \int_a^b(h^{''}(x))^2dx + \int_a^b \mu^{''}(x)h^{''}(x)dx$ from last question, $\int \mu^{''}(x)h^{''}(x)dx = 0\backslash$ we get $\int_a^b(g^{''}(x))^2 - \int_a^b(\mu^{''}(x))^2 = \int_a^b(h^{''}(x))^2dx$ \rhs is nonnegative so $\int_a^b(g^{''}(x))^2dx \geq \int_a^b(\mu^{''}(x))^2dx$

the backward direction is trival

for the forward direction

we have $\int_a^b(h^{''}(x))^2dx = 0$ thus we must have $h^{''}(x) = 0$ for all x.

we conclude that $h^{'}(x) = c_1$ and $h(x) = c_1x + c_2$

yet we know $g(x_1) - \mu(x_1) = h(x_1) = 0$ and $g(x_2) - \mu(x_2) = h(x_2) = 0$, we can solve the previous equation $c_1 = c_2 = 0$

so $h(x) = 0$, we conclude $g(x) - \mu(x) = 0$ for all x in [a,b]

(f)

$min_g \left[\sum_{i=1}^{n}(y_i - g(x_i))^2 + \lambda \int_a^b (g^{''}(x))^2 dx\right]$

from(e), we know there is no polynomial that has smaller $A = \int_a^b (f^{''}(x))^2 dx$ then the natural cubic splines, we also observe for natural cubic splines, $B = \sum_{i=1}^{n}(y_i - g(x_i))^2 = 0$ as $\mu(x_i) = y_i$ this is also the smallest possible value for B. natral cubic splines has the uniquely smallest A and B value, we conclude it has uniquely smallest C=A+B value.