HW2

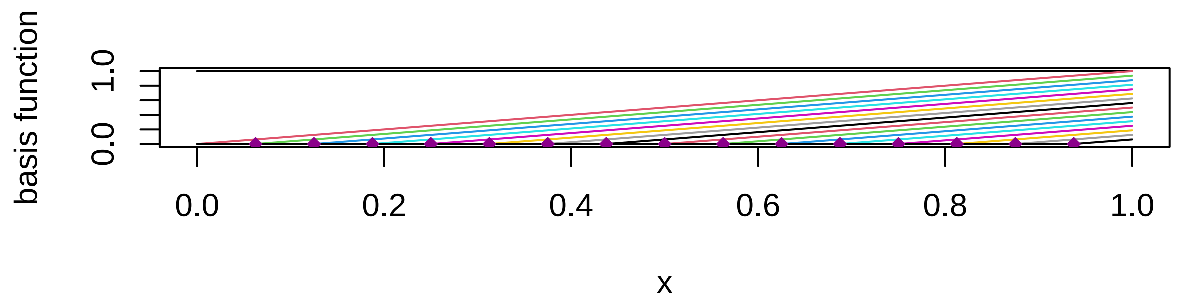
ASG

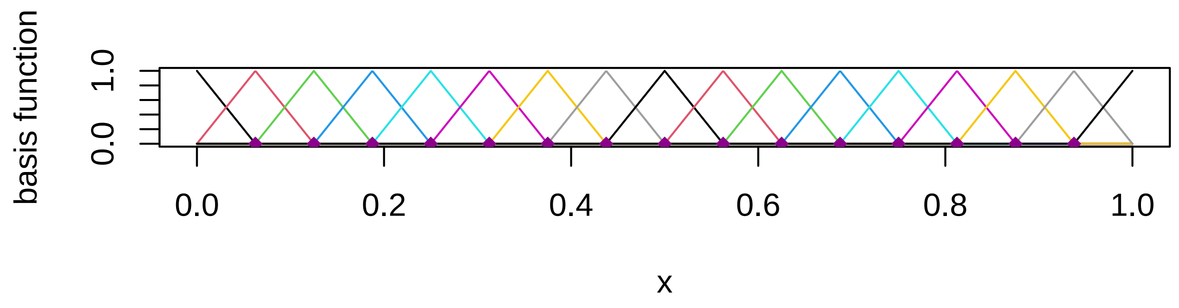
2023-03-14

2.3

a

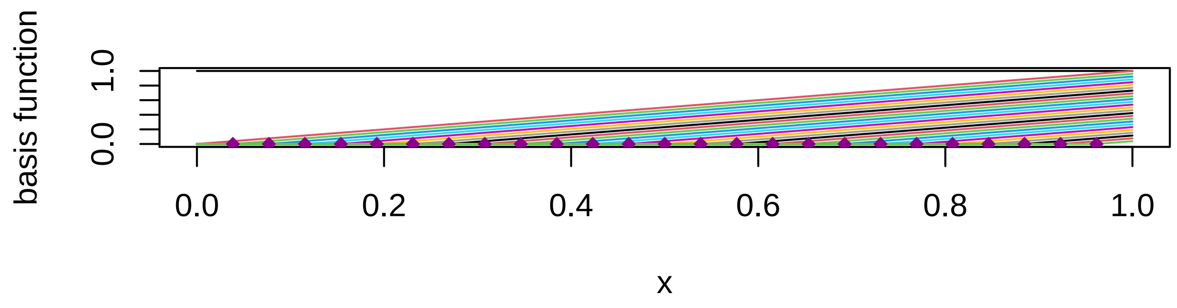
```r
library(splines) ; par(mfrow = c(2,1))
K <- 15 ; ng <- 1001
knots <- seq(0,1,length = (K+2))[-c(1,(K+2))]
xg <- seq(0,1,length = ng)
ZTg <- outer(xg,knots,"-") ; ZTg <- ZTg*(ZTg>0)
CTg <- cbind(1,xg,ZTg)
plot(0,type = "n",xlim = range(xg),ylim = range(CTg),xlab = "x",ylab = "basis function")
for (j in 1:(2+K)) lines(xg,CTg[,j],col = j)
points(knots,rep(0,K),col = "darkmagenta",pch = 18)
allKnots <- c(rep(0,2),knots,rep(1,2))
CBg <- spline.des(allKnots,xg,ord = 2,outer.ok = TRUE)$design
plot(0,type = "n",xlim = range(xg),ylim = range(CBg),xlab = "x",ylab = "basis function")
for (j in 1:(2+K)) lines(xg,CBg[,j],col = j)
points(knots,rep(0,K),col = "darkmagenta",pch = 18)
```

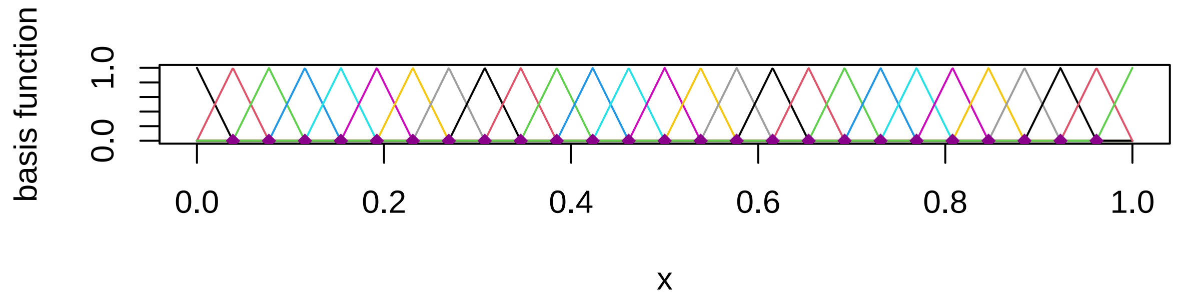




b

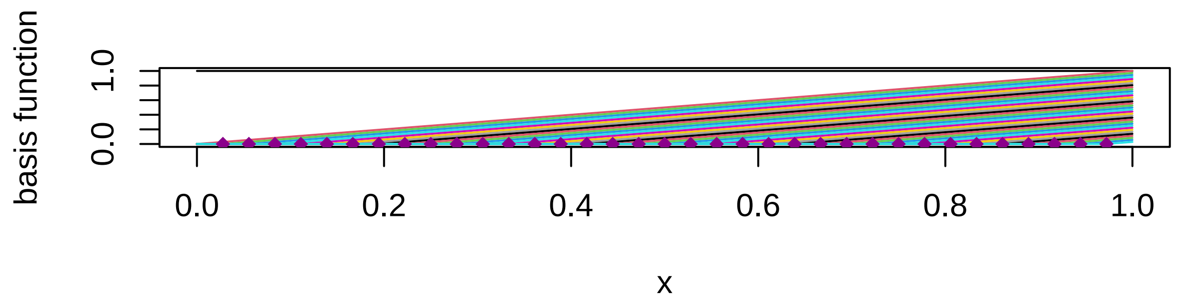
```r
par(mfrow = c(2,1))
K <- 25 ; ng <- 1001
knots <- seq(0,1,length = (K+2))[-c(1,(K+2))]
xg <- seq(0,1,length = ng)
ZTg <- outer(xg,knots,"-") ; ZTg <- ZTg*(ZTg>0)
CTg <- cbind(1,xg,ZTg)
plot(0,type = "n",xlim = range(xg),ylim = range(CTg),xlab = "x",ylab = "basis function")
for (j in 1:(2+K)) lines(xg,CTg[,j],col = j)
points(knots,rep(0,K),col = "darkmagenta",pch = 18)
allKnots <- c(rep(0,2),knots,rep(1,2))
CBg <- spline.des(allKnots,xg,ord = 2,outer.ok = TRUE)$design
plot(0,type = "n",xlim = range(xg),ylim = range(CBg),xlab = "x",ylab = "basis function")
for (j in 1:(2+K)) lines(xg,CBg[,j],col = j)
points(knots,rep(0,K),col = "darkmagenta",pch = 18)
```

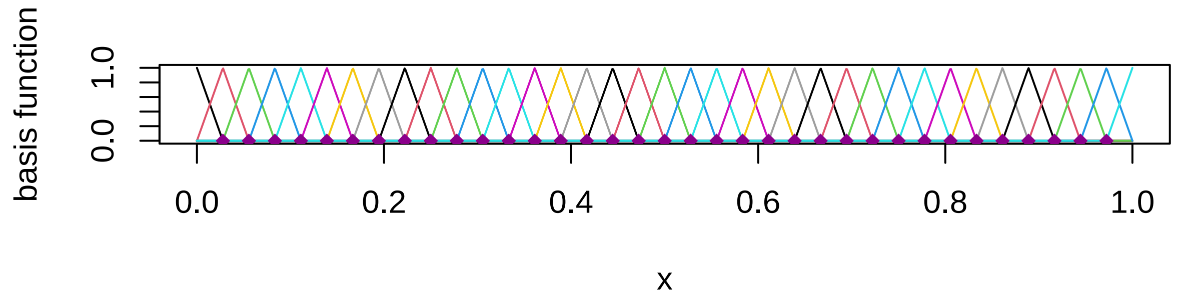




b

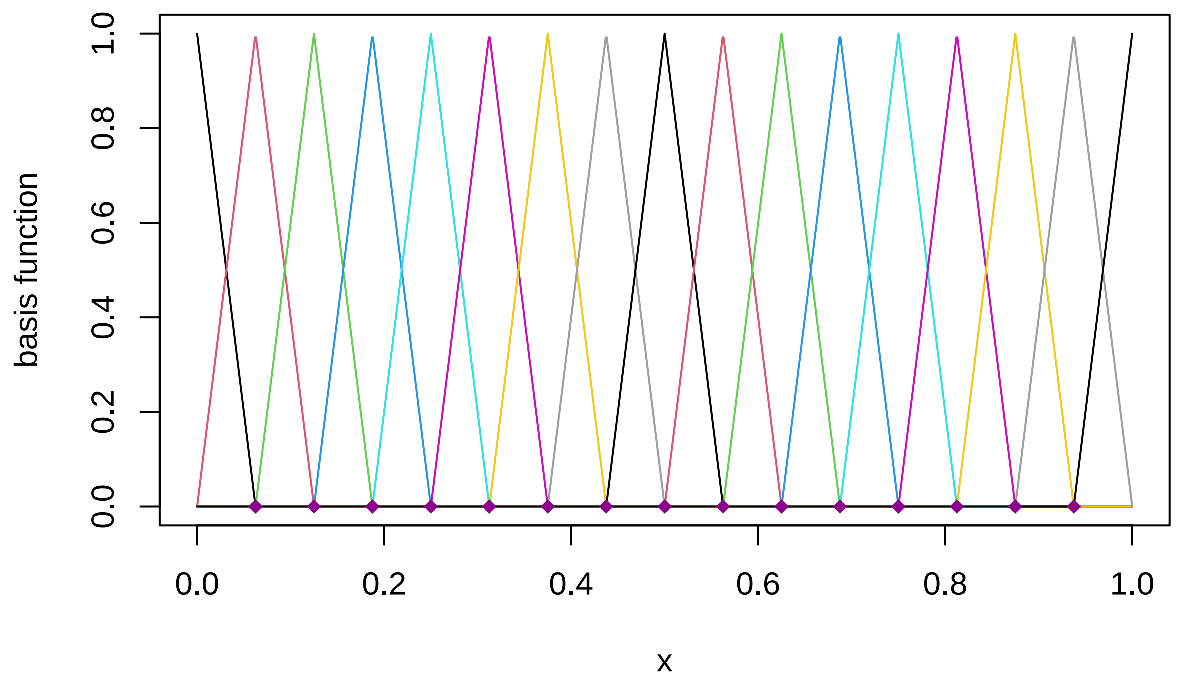
```
par(mfrow = c(2,1))
K <- 35 ; ng <- 1001
knots <- seq(0,1,length = (K+2))[-c(1,(K+2))]
xg <- seq(0,1,length = ng)
ZTg <- outer(xg,knots,"-") ; ZTg <- ZTg*(ZTg>0)
CTg <- cbind(1,xg,ZTg)
plot(0,type = "n",xlim = range(xg),ylim = range(CTg),xlab = "x",ylab = "basis function")
for (j in 1:(2+K)) lines(xg,CTg[,j],col = j)
points(knots,rep(0,K),col = "darkmagenta",pch = 18)
allKnots <- c(rep(0,2),knots,rep(1,2))
CBg <- spline.des(allKnots,xg,ord = 2,outer.ok = TRUE)$design
plot(0,type = "n",xlim = range(xg),ylim = range(CBg),xlab = "x",ylab = "basis function")
for (j in 1:(2+K)) lines(xg,CBg[,j],col = j)
points(knots,rep(0,K),col = "darkmagenta",pch = 18)
```





c

```
K <- 15 ; ng <- 1001
knots <- seq(0,1,length = (K+2))[-c(1,(K+2))]
xg <- seq(0,1,length = ng)
ZTg <- outer(xg,knots,"-") ; ZTg <- ZTg*(ZTg>0)
CTg <- cbind(1,xg,ZTg) ; LTB <- matrix(0,(K+2),(K+2))
LTB[1,1] <- 1 ; LTB[2,1:2] <- (K+1)*c(-1,1)
for (k in 3:(K+2)) LTB[k,((k-2):k)] <- (K+1)*c(1,-2,1)
CBviaLTBg <- CTg%*%LTB
plot(0,type = "n",xlim = range(xg),ylim = range(CBviaLTBg),xlab = "x",ylab = "basis function")
for (j in 1:(2+K)) lines(xg,CBviaLTBg[,j],col = j)
points(knots,rep(0,K),col = "darkmagenta",pch = 18)
```



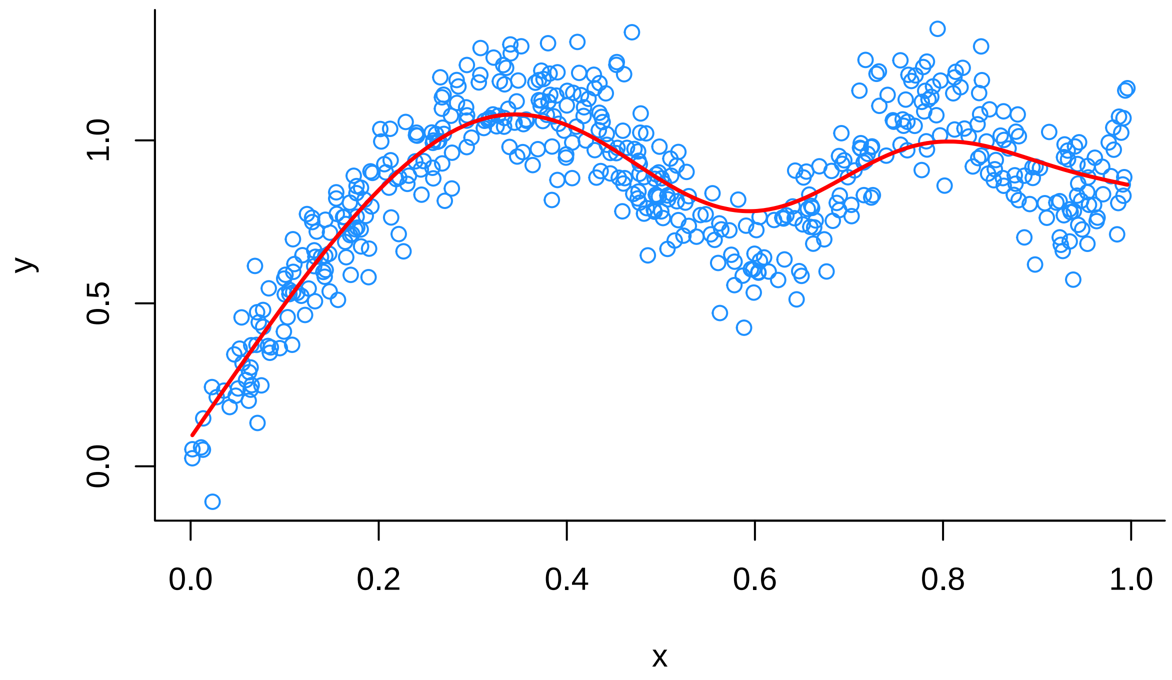
d

```
detL=det(LTB)
val=(K+1)**(K+1)
all.equal(val,detL)
```

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

e

```
set.seed(1) ; n <- 500 ; x <- sort(runif(n))
y <- (6*x + sin(4*pi*x^2))/(5*x+1) + 0.1*rnorm(n)
fit<-smooth.spline(x,y,nknots=35,lambda=0.01)
plot(x,y,bty = "l",col = "dodgerblue")
lines(fit,lwd=2,col="red")
```



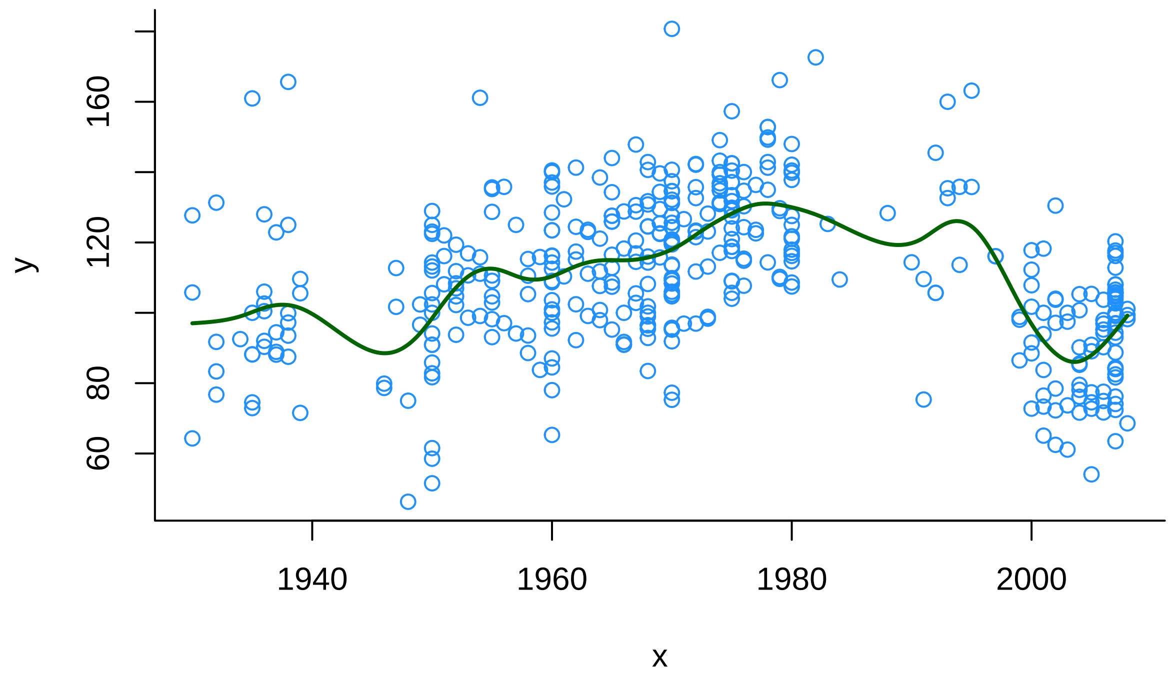
2.5

prep

```
library(HRW) ; library(mgcv) ; data(WarsawApts)
```

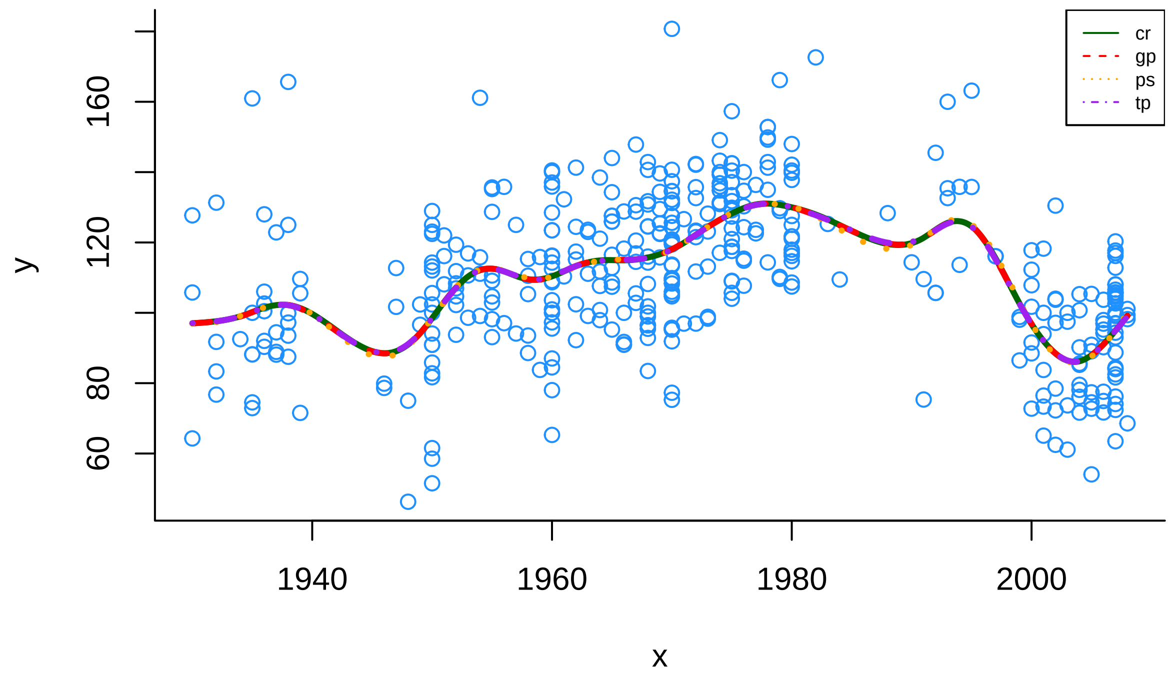
```
## Loading required package: nlme

## This is mgcv 1.8-42. For overview type 'help("mgcv-package")'.
```

```
x <- WarsawApts$construction.date
y <- WarsawApts$areaPerMzloty
plot(x,y,bty = "l",col = "dodgerblue")
fitGAMcr <- gam(y ~ s(x,bs = "cr",k = 30))
xg <- seq(min(x),max(x),length = 1001)
fHatgGAMcr <- predict(fitGAMcr,newdata = data.frame(x = xg))
lines(xg,fHatgGAMcr,col = "darkgreen",lwd=2)
```



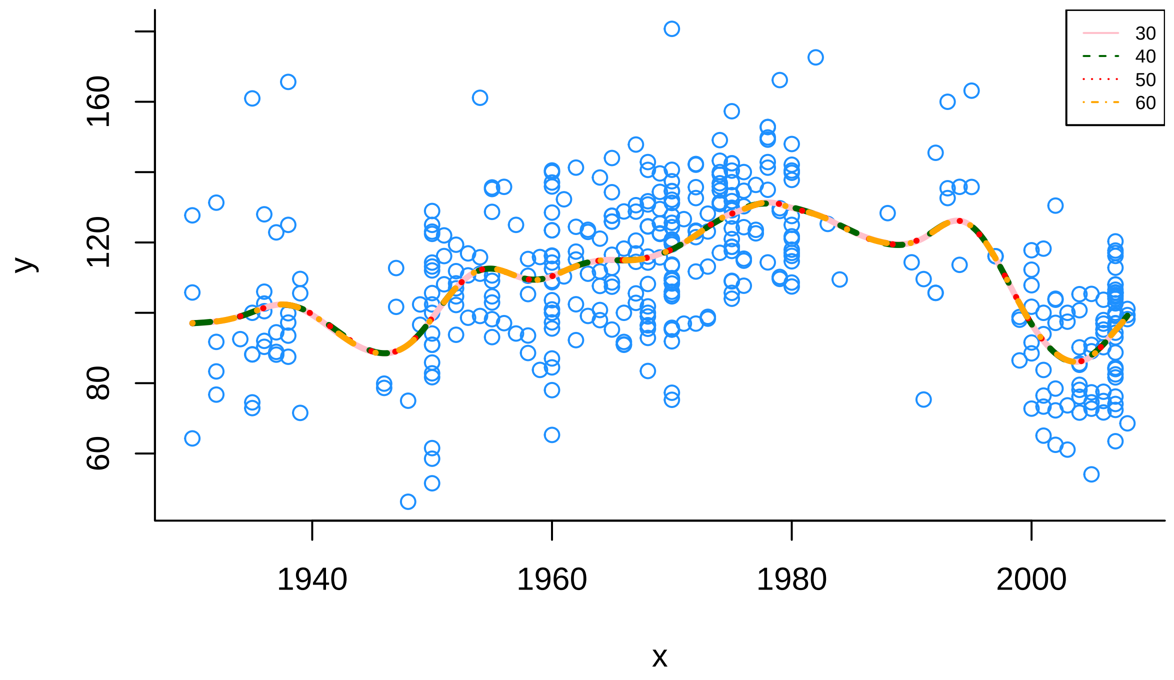
a

```
x <- WarsawApts$construction.date
y <- WarsawApts$areaPerMzloty
plot(x,y,bty = "l",col = "dodgerblue")
fitGAMcr <- gam(y ~ s(x,bs = "cr",k = 30))
fitGAMgp <- gam(y ~ s(x,bs = "gp",k = 30))
fitGAMps <- gam(y ~ s(x,bs = "ps",k = 30))
fitGAMtp <- gam(y ~ s(x,bs = "tp",k = 30))
xg <- seq(min(x),max(x),length = 1001)
fHatgGAMcr <- predict(fitGAMcr,newdata = data.frame(x = xg))
fHatgGAMgp <- predict(fitGAMgp,newdata = data.frame(x = xg))
fHatgGAMps <- predict(fitGAMps,newdata = data.frame(x = xg))
fHatgGAMtp <- predict(fitGAMtp,newdata = data.frame(x = xg))
lines(xg,fHatgGAMcr,col = "darkgreen",lty=1,lwd=3)
lines(xg,fHatgGAMgp,col = "red",lty=2,lwd=3)
lines(xg,fHatgGAMps,col = "orange",lty=3,lwd=3)
lines(xg,fHatgGAMtp,col = "purple",lty=4,lwd=3)
legend('topright',legend=c("cr","gp","ps","tp"),col=c("darkgreen","red","orange","purple"),lty = 1:4, c
```



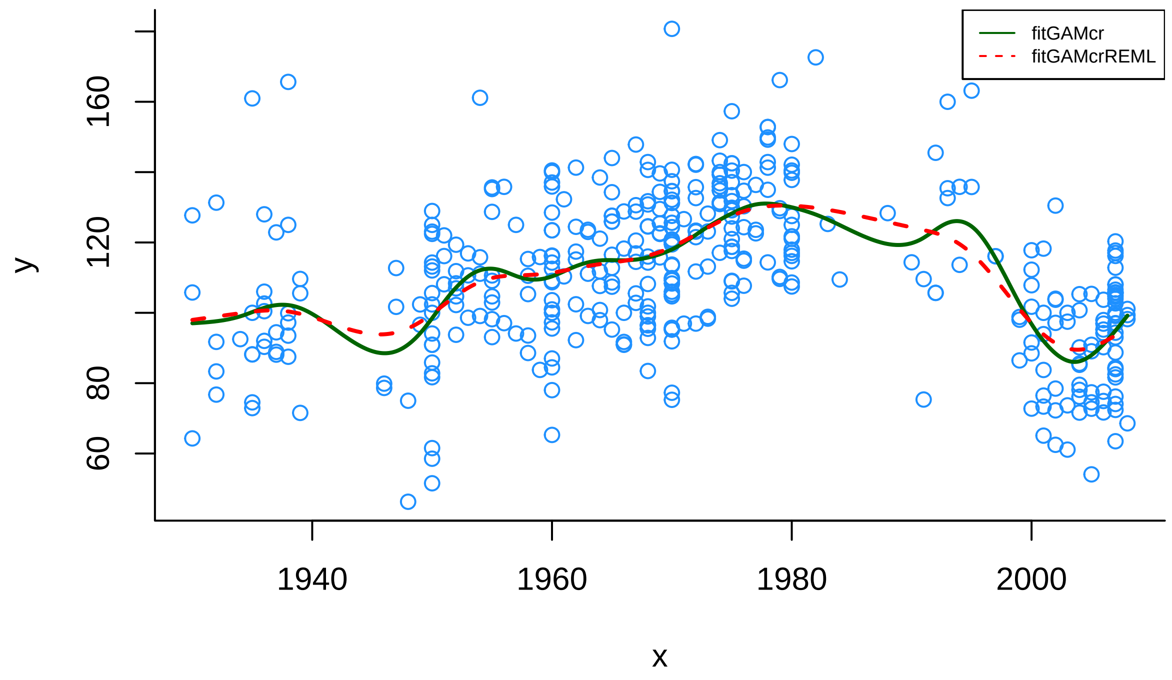
b

```
x <- WarsawApts$construction.date
y <- WarsawApts$areaPerMzloty
plot(x,y,bty = "l",col = "dodgerblue")
fitGAM30 <- gam(y ~ s(x,bs = "cr",k = 30))
fitGAM40 <- gam(y ~ s(x,bs = "cr",k = 40))
fitGAM50 <- gam(y ~ s(x,bs = "cr",k = 50))
fitGAM60 <- gam(y ~ s(x,bs = "cr",k = 60))
xg <- seq(min(x),max(x),length = 1001)
fHatgGAM30 <- predict(fitGAM30,newdata = data.frame(x = xg))
fHatgGAM40 <- predict(fitGAM40,newdata = data.frame(x = xg))
fHatgGAM50 <- predict(fitGAM50,newdata = data.frame(x = xg))
fHatgGAM60 <- predict(fitGAM60,newdata = data.frame(x = xg))
lines(xg,fHatgGAM60,col = "pink",lty=1,lwd=3)
lines(xg,fHatgGAM30,col = "darkgreen",lty=2,lwd=3)
lines(xg,fHatgGAM40,col = "red",lty=3,lwd=3)
lines(xg,fHatgGAM50,col = "orange",lty=4,lwd=3)
legend('topright',legend=c("30","40","50","60"),col=c("pink","darkgreen","red","orange"),lty=1:4, cex=0
```



c

```
x <- WarsawApts$construction.date
y <- WarsawApts$areaPerMzloty
plot(x,y,bty = "l",col = "dodgerblue")
fitGAMcr <- gam(y ~ s(x,bs = "cr",k = 30))
fitGAMcrREML <- gam(y ~ s(x,bs = "cr",k = 30),method = "REML")
xg <- seq(min(x),max(x),length = 1001)
fHatgGAMcr <- predict(fitGAMcr,newdata = data.frame(x = xg))
fHatgGAMcrREML <- predict(fitGAMcrREML,newdata = data.frame(x = xg))
lines(xg,fHatgGAMcr,col = "darkgreen",lwd=2)
lines(xg,fHatgGAMcrREML,col = "red",lty=2,lwd=2)
legend('topright',legend=c("fitGAMcr","fitGAMcrREML"),col=c("darkgreen","red"),lty=1:2, cex=0.6)
```



### d Accroding to the fits with variations we can draw the following conclusions: 1. Based on K: No variability in fit was observed as we kept on increasing K. The fits perfectly superimposed each other indicating that fitting plateaus after a certain K threshold. This means it cannot overfit more beyond a certain K value. 2. Based on Basis type: The basis type had a minimal effect for this dataset and K value. Ps type shows slight perturbations than the other 3. The other 3 more or less superimposes , thus showing not much signifanct difference between the types. It can be noted that the due to high value of K, overfitting may hide the variablity caused by different types. 3. Based on Smoothing parameter: There is a high degree of variability by the two fits. REML seems to have higher degree of bias thus cr based smoothing params are overfitting more wrt REML

3

a

The three tuning parameters are : 1. Knots i.e K 2. type of splines 3. Smoothing parameter

b

The smoothing parameter has the highest effect on the fit. As it the smoothing parameter penalizes for overfitting thus controlling the bias variance tradeoff. a very small smoothing parameter would lead to very high variance that is overfit the model and a veryhigh would lead to more bias that is underfit the model

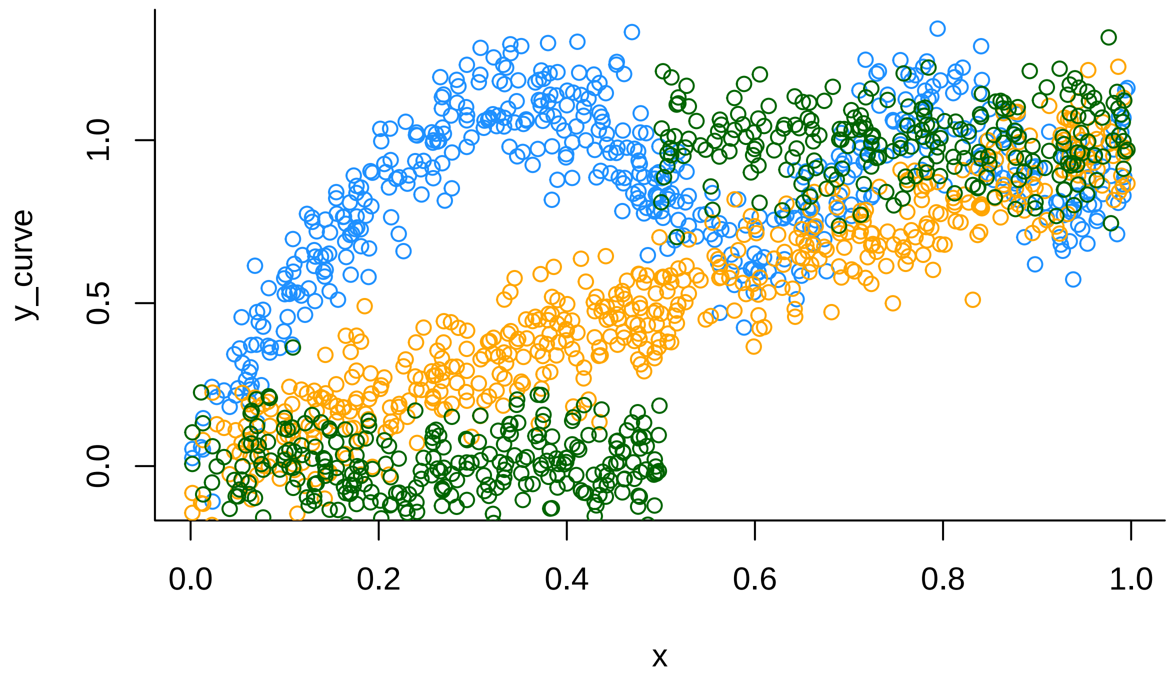
c

The GCV criterion function is one way employed by the packages to automatically choose the smoothing parameter.

4

pre

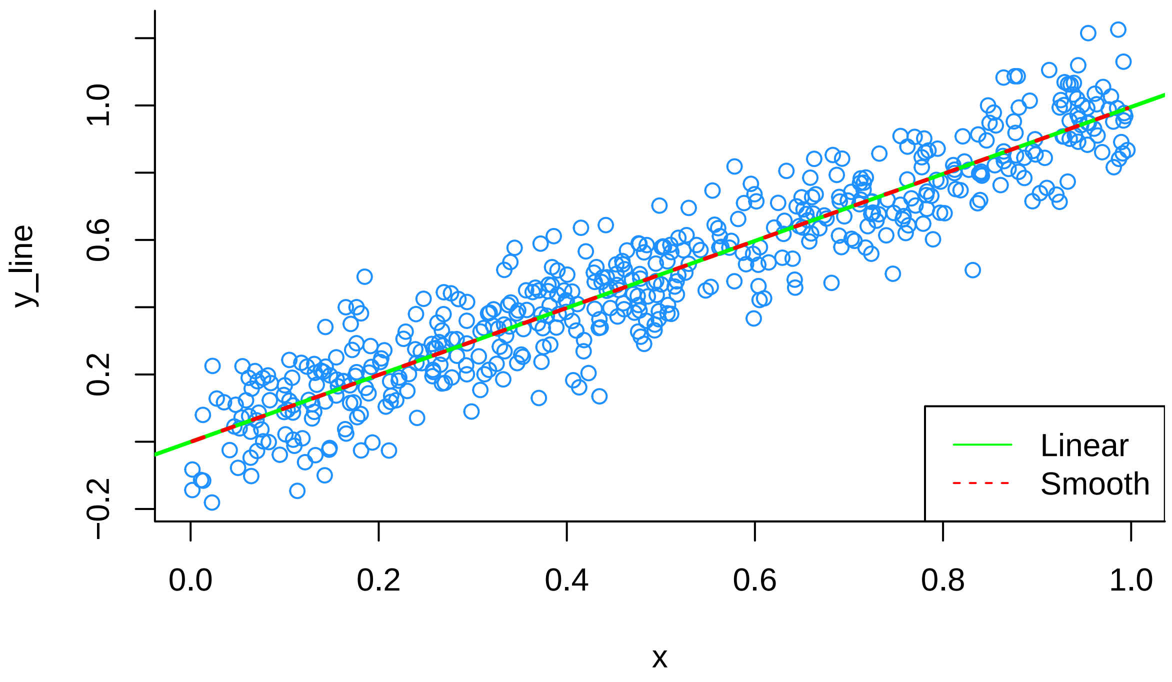
```
set.seed(1) ; n <- 500 ; x <- sort(runif(n))
y_curve <- (6*x + sin(4*pi*x^2))/(5*x+1) + 0.1*rnorm(n)
plot(x,y_curve,bty="l", col="dodgerblue")
y_line <- x + 0.1*rnorm(n)
points(x,y_line, bty="l", col="orange")
y_step <- as.numeric(x>.5) + 0.1*rnorm(n)

points(x,y_step, bty="l", col="darkgreen")
```



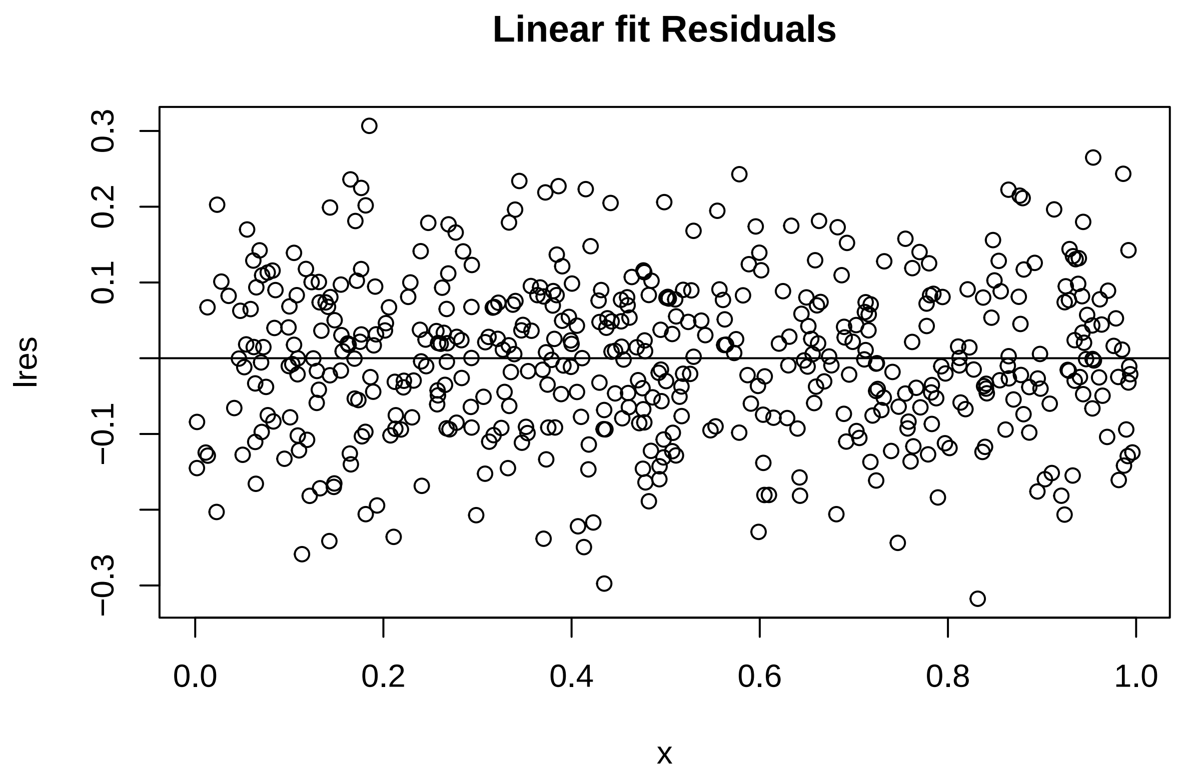
### a As per the above scatter plot, the first data set would be the best dataset to use penalized splines fit for.

b

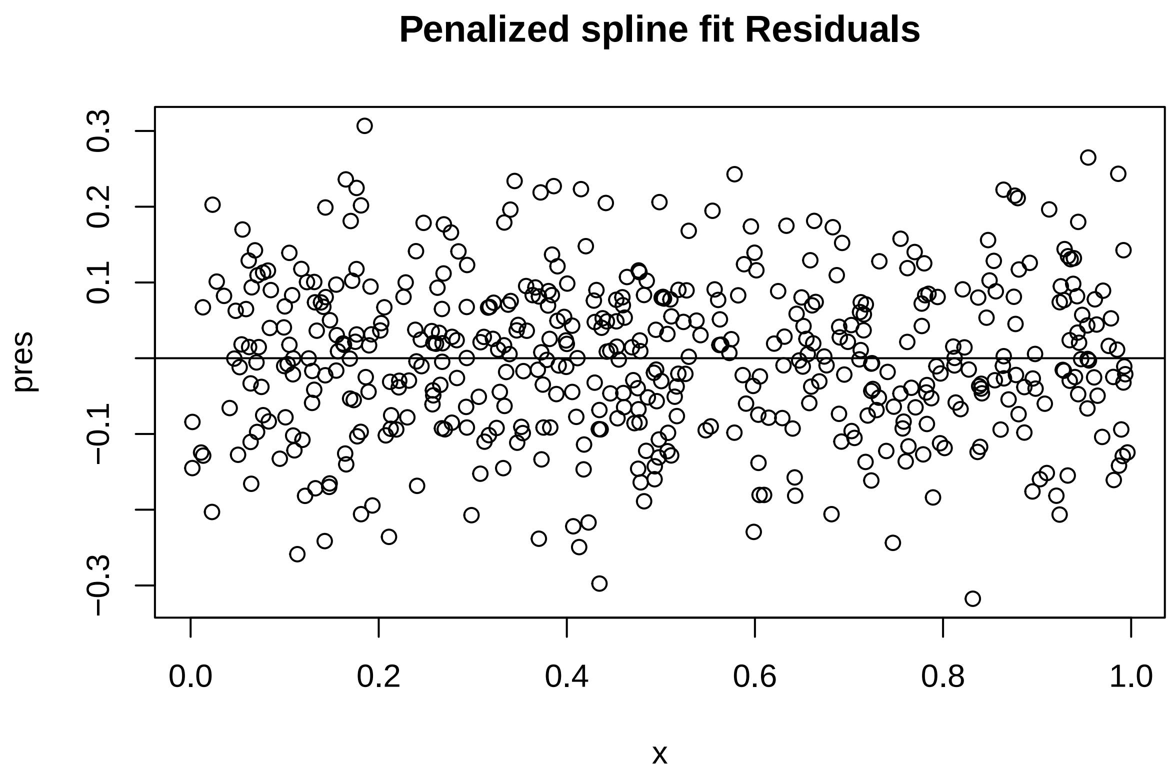
```
lfit1<-lm(y_line~x)
pfit1<-gam(y_line ~ s(x,bs = "cr"))
plot(x,y_line,bty="l", col="dodgerblue")
xg <- seq(min(x),max(x),length = 1001)
abline(lfit1, col="green",lwd=2)
lines(xg,predict(pfit1,newdata=data.frame(x=xg)),col="red",lty=2,lwd=2)
legend("bottomright",legend=c("Linear","Smooth"),col=c("green","red"),lty=1:2)
```



```
lres<-resid(lfit1)
pres<-residuals(pfit1)
plot(x, lres,main="Linear fit Residuals")
abline(0,0)
```



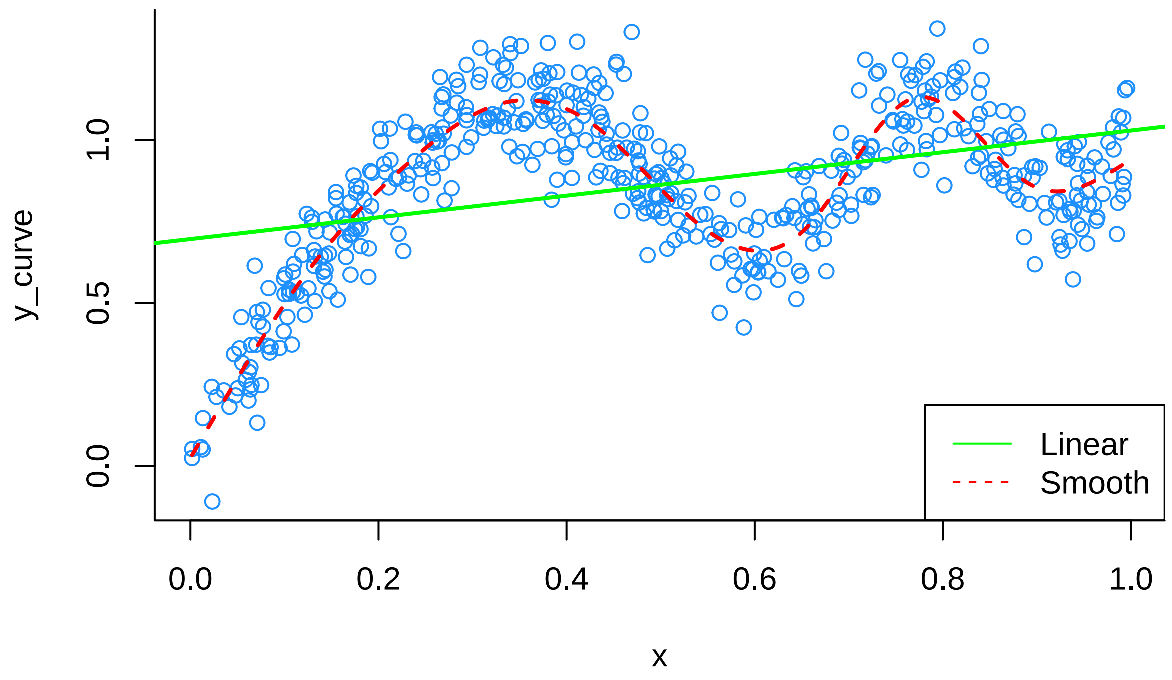
```
plot(x,pres,main="Penalized spline fit Residuals")
abline(0,0)
```



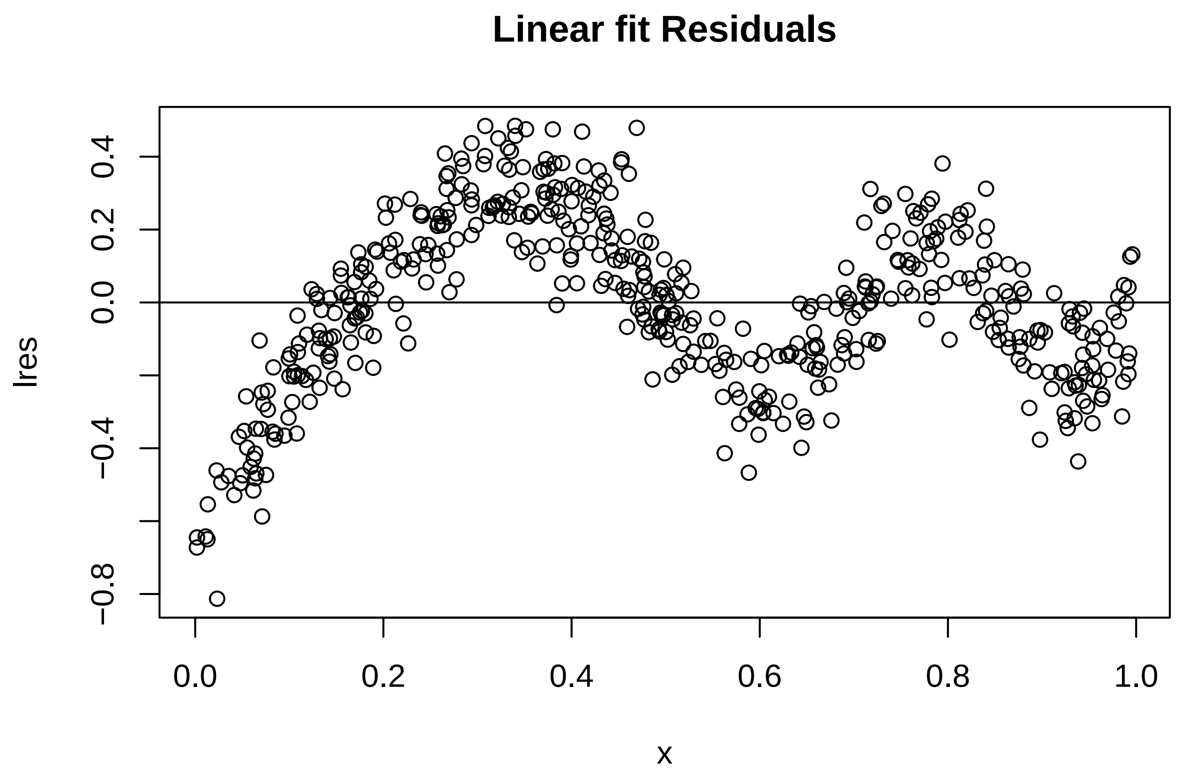
```
summary(pfit1)$edf
```

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

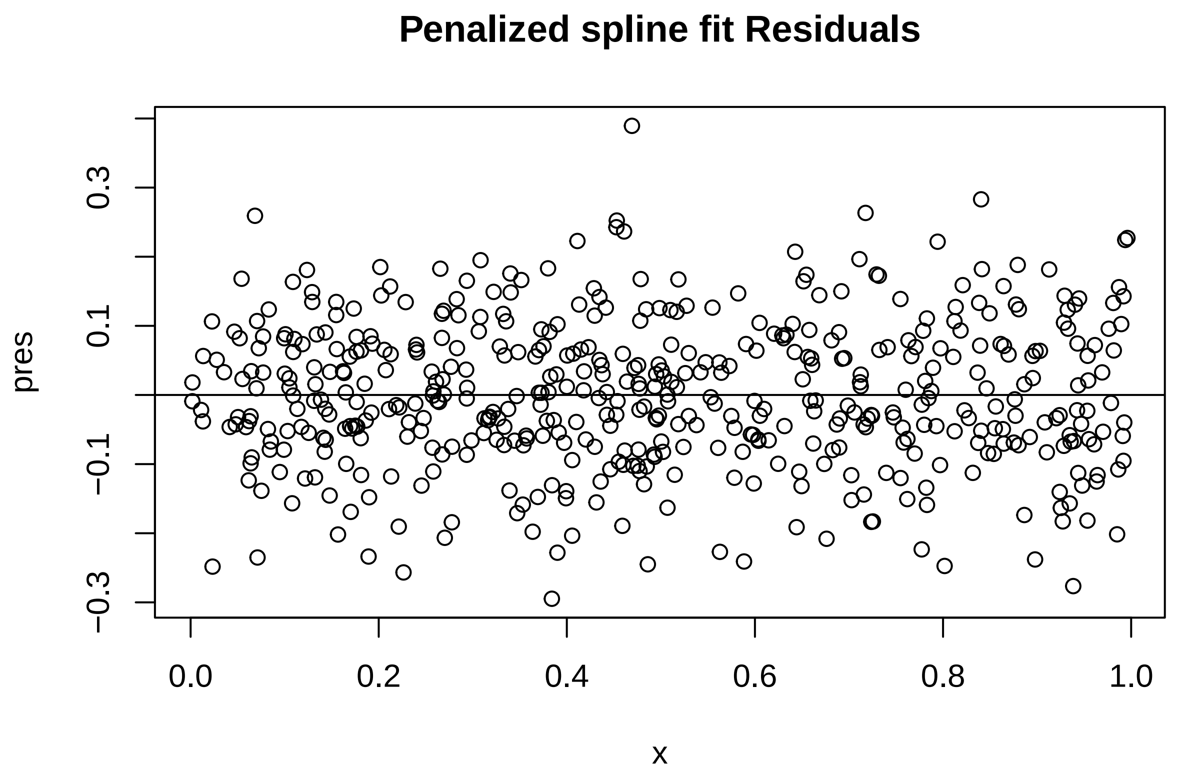
```
lfit1<-lm(y_curve~x)
pfit1<-gam(y_curve ~ s(x,bs = "cr"))
plot(x,y_curve,bty="l", col="dodgerblue")
xg <- seq(min(x),max(x),length = 1001)
lines(xg,predict(pfit1,newdata=data.frame(x=xg)),col="red",lty=2,lwd=2)
abline(lfit1, col="green",lwd=2)
legend("bottomright",legend=c("Linear","Smooth"),col=c("green","red"),lty=1:2)
```



```
lres<-resid(lfit1)
pres<-residuals(pfit1)
plot(x, lres,main="Linear fit Residuals")
abline(0,0)
```



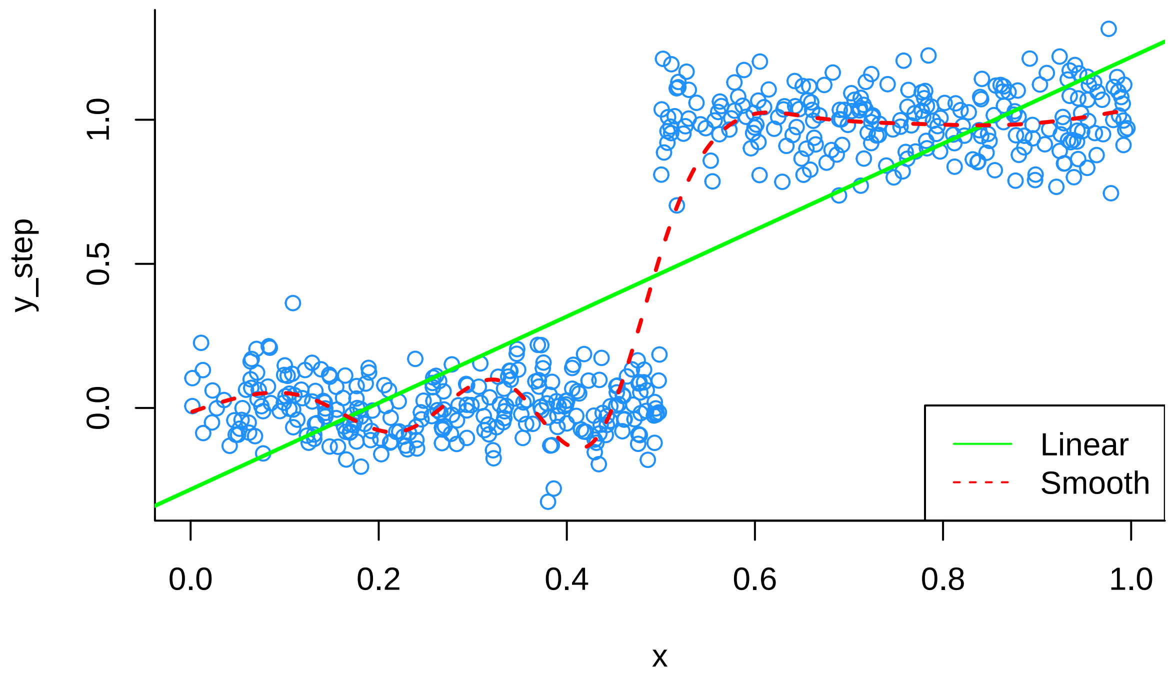
```
plot(x,pres,main="Penalized spline fit Residuals")
abline(0,0)
```



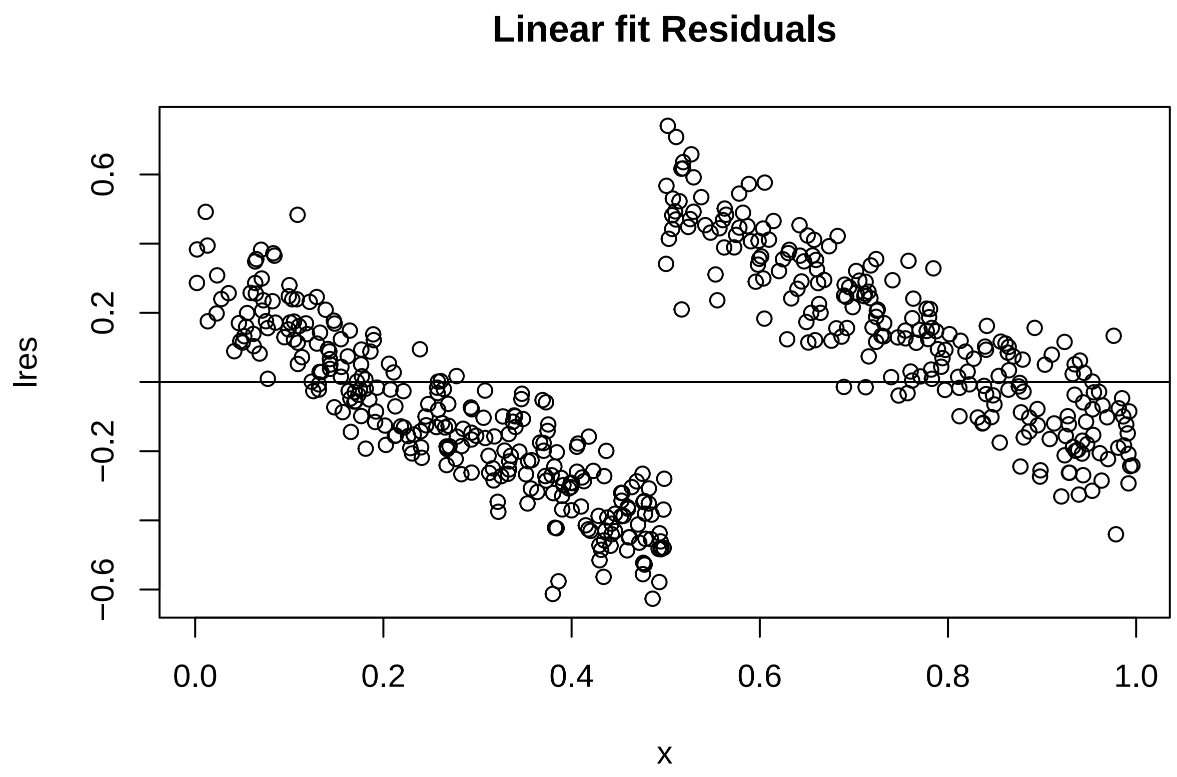
```
summary(pfit1)$edf
```

```
## [1] 8.851757
```

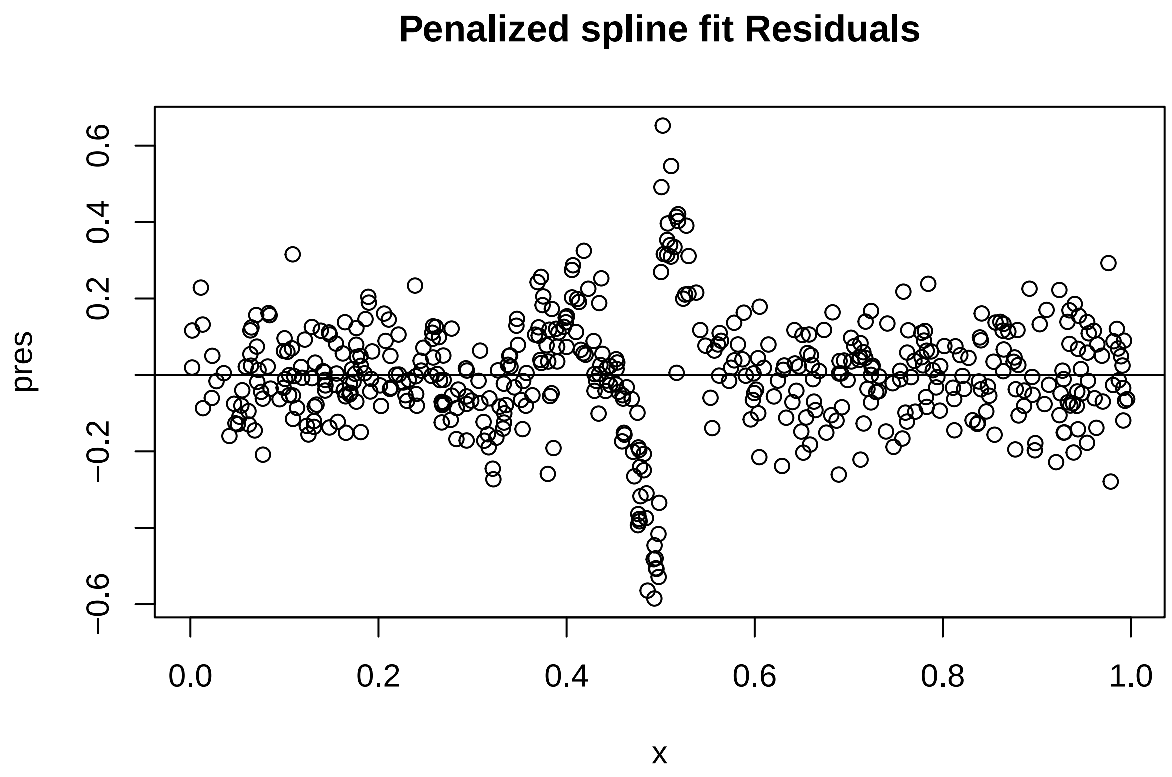
```
lfit1<-lm(y_step~x)
pfit1<-gam(y_step ~ s(x,bs = "cr"))
plot(x,y_step,bty="l", col="dodgerblue")
xg <- seq(min(x),max(x),length = 1001)
abline(lfit1, col="green",lwd=2)
lines(xg,predict(pfit1,newdata=data.frame(x=xg)),col="red",lty=2,lwd=2)
legend("bottomright",legend=c("Linear","Smooth"),col=c("green","red"),lty=1:2)
```



```
lres<-resid(lfit1)
pres<-residuals(pfit1)
plot(x,lres,main="Linear fit Residuals")
abline(0,0)
```



```
plot(x,pres,main ="Penalized spline fit Residuals")
abline(0,0)
```



```
summary(pfit1)$edf
```

```
## [1] 8.964206
```

c

1. linear data, both the linear and penalized splines yielded the same fit results. For both of them the residuals were very normally distributed. Thus we can conclude that , for linear data , both the methods would yeild very similar results
2. The curve i.e non linear data, it was no surprise that penalized spline was very effctive. The linear fit wasnt able to properly explain the variation and hence had abnormal and non linear residual plot. meanwhile the penalized spline fit modelled the data very well and had normally distributed resdiuals.
3. For the step function, the penalized spline did a decent job at modelling the step function data. The resdiuals were normally distributed apart from the place where the step occurred. Linear fit wasnt able to model the data very well and had an abnormal periodic kind of resdiual distribution due to step.

d

The EDF of linear data is 1 and can be explained as such since it is a linear data, only 1 spline basis function would be able to model it efficiently. and hence only one parameter was required to fit.

the EDF for the other two is nearly equal to 8.x. which means nearly 9ish basis spline function is needed to efficiently mdoel the data and hence 9 parameters needed to be estimated.

We can conclude that the the EDF tells about the complexity of penalized spline model. Linear has the least complexity while non linear data has to be modelled with higher complexity thus higher edf.