Module 7 - Nonlinear regression.

Pawel Chilinski

December 22, 2013

Exercise 1.

Data set USPop in package car contains information on population of USA within years 1790-2000.

```
> library(car)
> data(USPop)
```

- Fit a population growth model of the form:

$$y_i = \frac{\beta_1}{1 + e^{\beta_2 + \beta_3 x_i}} + \epsilon_i$$

where y_i is a population in year x_i, using nonlinear least squares method (function nls()). Take initial values of parameters as $\beta_1 = 350$, $\beta_2 = 4.5$, $\beta_3 = -0.3$. What are the estimated values $\hat{\beta}_1$, $\hat{\beta}_2$, $\hat{\beta}_3$?

```
> population.st=c(b1=350,b2=4.5,b3=-0.3)
> time <- 0:(nrow(USPop)-1)
> population.nls<-nls(population~b1/(1+exp(b2+b3*time)), data=USPop, start=population.st, trace=T)

13217.35 :  350.0   4.5  -0.3
1861.728 :  370.4674576   3.7839326  -0.2153424
538.287 :  430.4181666   4.0390595  -0.2174231
457.8314 :  440.9958648   4.0313233  -0.2159307
457.8057 :  440.7942364   4.0324813  -0.2160736
457.8056 :  440.8363258   4.0323905  -0.2160578
457.8056 :  440.8331803   4.0324001  -0.2160592

> population.nls.sum<-summary(population.nls)
```

The estimated values are $\hat{\beta}_1 = 440.83$, $\hat{\beta}_2 = 4.03$, $\hat{\beta}_3 = -0.22$.

- Draw the fitted curve on the plot of the data.

```
> plot(USPop$year,USPop$population,main="Population",xlab="year",ylab="population")
> b1<-population.nls.sum$coef["b1",1]
> b2<-population.nls.sum$coef["b2",1]
> b3<-population.nls.sum$coef["b3",1]
> lines(USPop$year,fitted.values(population.nls),col="blue")
> legend(x="topleft",lty=c(-1,1),col=c("black","blue"),
+                 legend=c("Population data","Fitted model"),pch = c(1,-1))
```

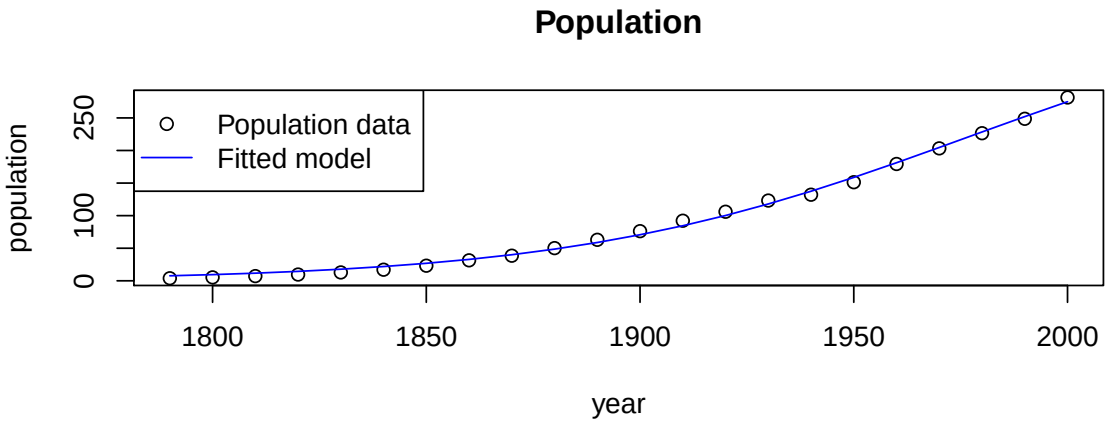

Figure 1: NLS fit for the population of USA within years 1790-2000

- Using the fitted model make a prediction of the population of USA in year 2015.

```
> relative_2015<-(2015-range(USPop$year)[1])/10
> (population_2015_predicted<-predict(population.nls,data.frame(time=relative_2015)))

[1] 306.8767
```

```
> plot(time,USPop$population,main="Population",xlab="year",ylab="population",xlim=c(0,relative_2015+1),
+                 ylim=c(0,population_2015_predicted+1000))
> b1<-population.nls.sum$coef["b1",1]
> b2<-population.nls.sum$coef["b2",1]
> b3<-population.nls.sum$coef["b3",1]
> curve(b1/(1+exp(b2+b3*x)),add=T,from=0,to=relative_2015,col="blue")
> points(relative_2015,population_2015_predicted,col="red")
> legend(x="topleft",lty=c(-1,1,-1),col=c("black","blue","red"),
+                 legend=c("Population data","Fitted model","Predicted population in 2015"),pch = c(1,-1,1))
```

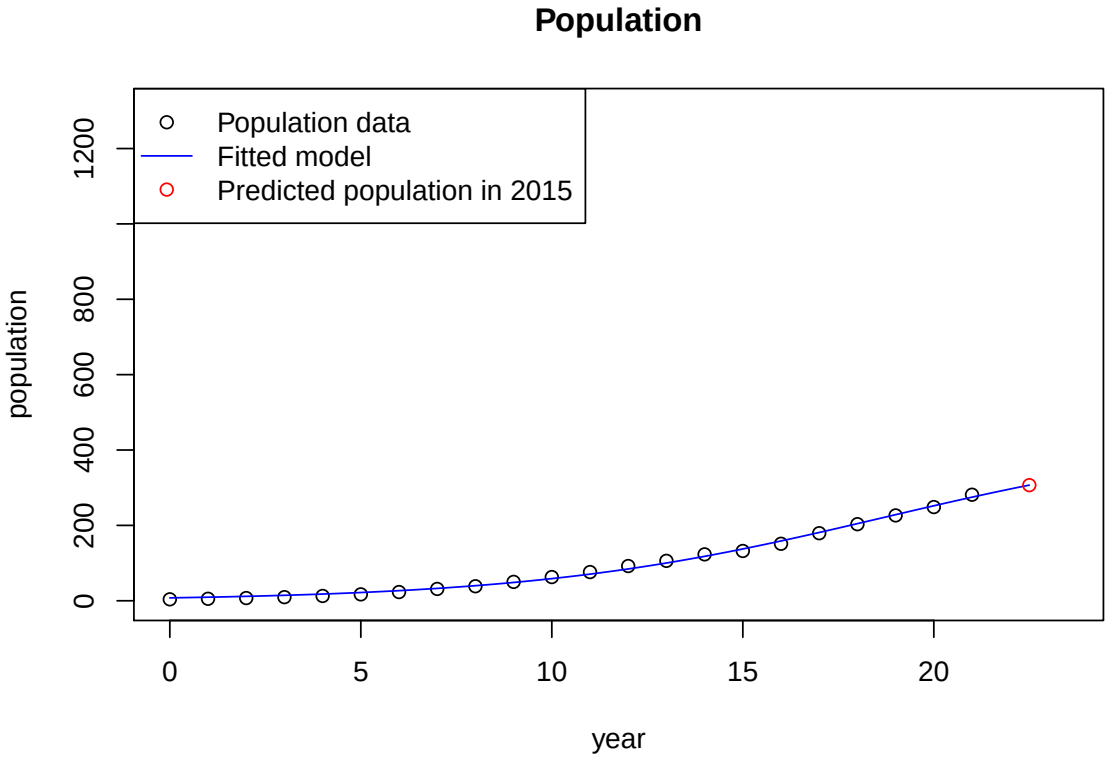

Figure 2: Predicted population in 2015

Exercise 2.

Data set Prestige in package car contains information on prestige, average income and education for 102 occupations in Canada.

```
> data(Prestige)
```

- Plot variable prestige against income.

```
> plot(Prestige$income,Prestige$prestige,xlab="income",ylab="prestige")
```

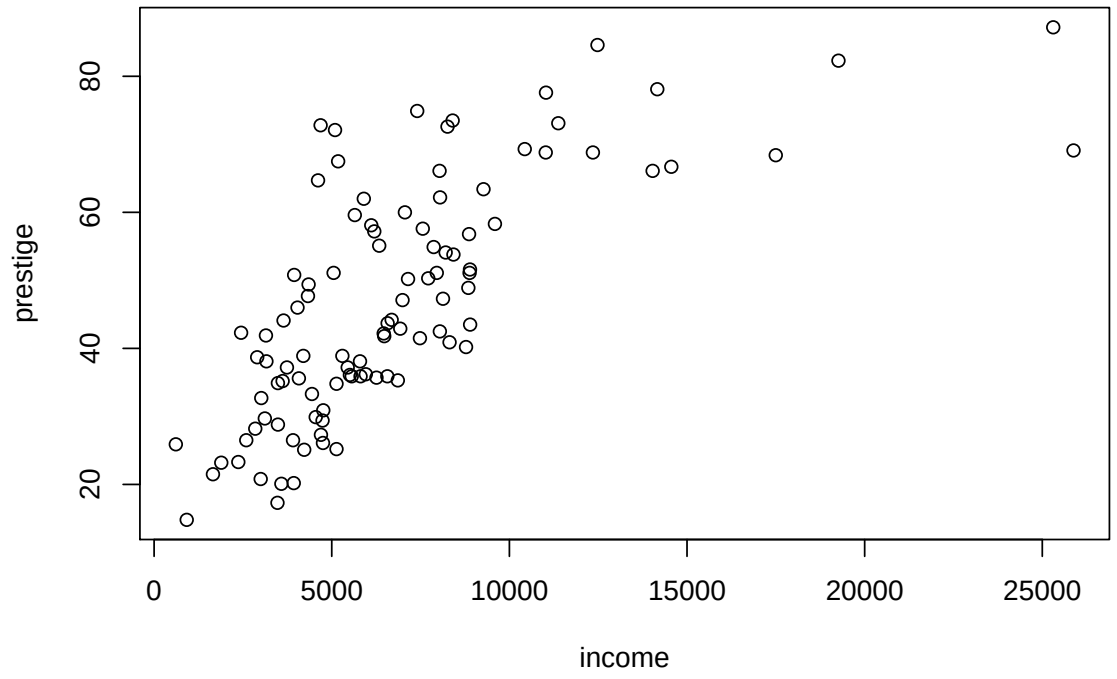

Figure 3: prestige against income

- Fit a moving average estimator of regression function. Use function loess() with parameter degree=0. Choose the best value of smoothing parameter span (visually). Draw the fitted curve.

  Fitting models with different values of span:

```
> prestige.mas<-list()
> for(s in seq(from=0.1,to=0.8,by=0.1)){
+         prestige.mas[[as.character(s)]]<-loess(prestige~income,Prestige,degree=0,span=s)
+ }
```

  It can be seen that small values of smoothing parameter results in the model that tries to fits the data too closely and is over-fitted.

```
> plot(Prestige$income,Prestige$prestige,xlab="income",ylab="prestige")
> colors<-1:length(prestige.mas)
> idx<-0
> for(s in seq(from=0.1,to=0.8,by=0.1)){
+         idx<-idx+1
+         s.char<-as.character(s)
+         sorted_idx<-sort(prestige.mas[[s.char]]$x,index.return=T)$ix
+         lines(prestige.mas[[s.char]]$x[sorted_idx],prestige.mas[[s.char]]$fitted[sorted_idx],
+                         col=colors[idx])
+ }
> legend(x="bottomright",lty=c(-1,rep(1,length(prestige.mas))),col=c(1,colors),
+                 legend=c("data",as.character(seq(from=0.1,to=0.8,by=0.1))),
+                 pch = c(1,rep(-1,length(prestige.mas))))
```

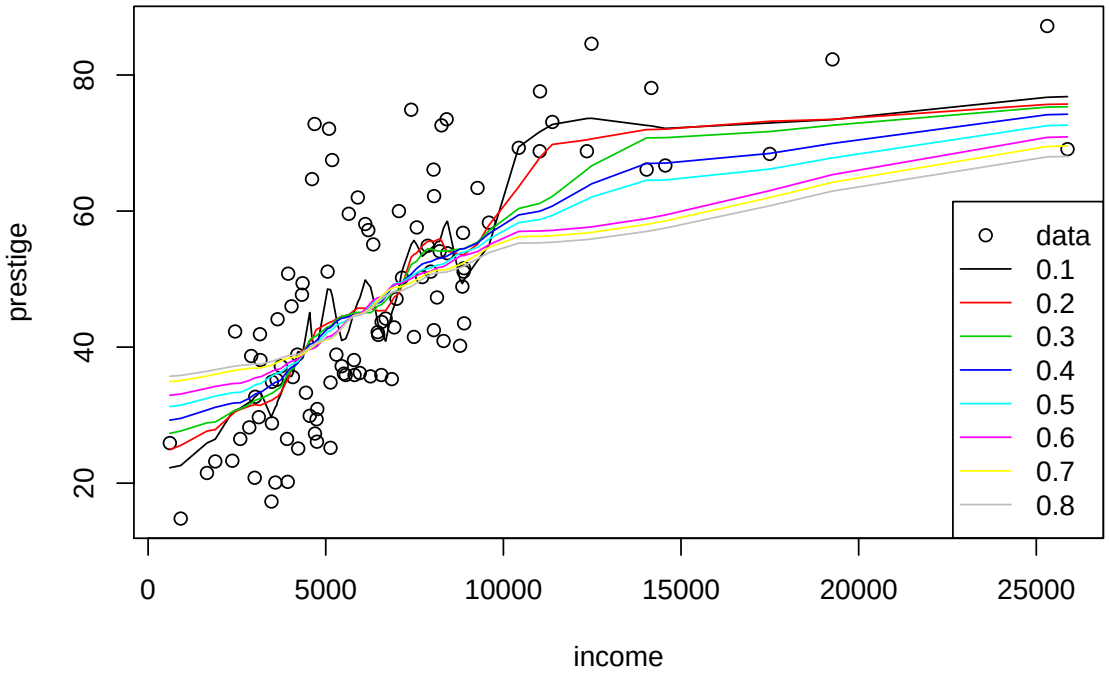

Figure 4: moving average model for different values of smoothing parameter

We should choose the model with smother line going through the data points (chosen 0.3):

```
> choosed.smoothing.param<-0.3
> prestige.ma<-prestige.mas[[as.character(choosed.smoothing.param)]]
> plot(Prestige$income,Prestige$prestige,xlab="income",ylab="prestige")
> sorted_idx<-sort(prestige.ma$x,index.return=T)$ix
> lines(prestige.ma$x[sorted_idx],prestige.ma$fitted[sorted_idx],col=2)
> legend(x="bottomright",lty=c(-1,1),col=c(1,2), legend=c("data",as.character(choosed.smoothing.param)),pch =
```

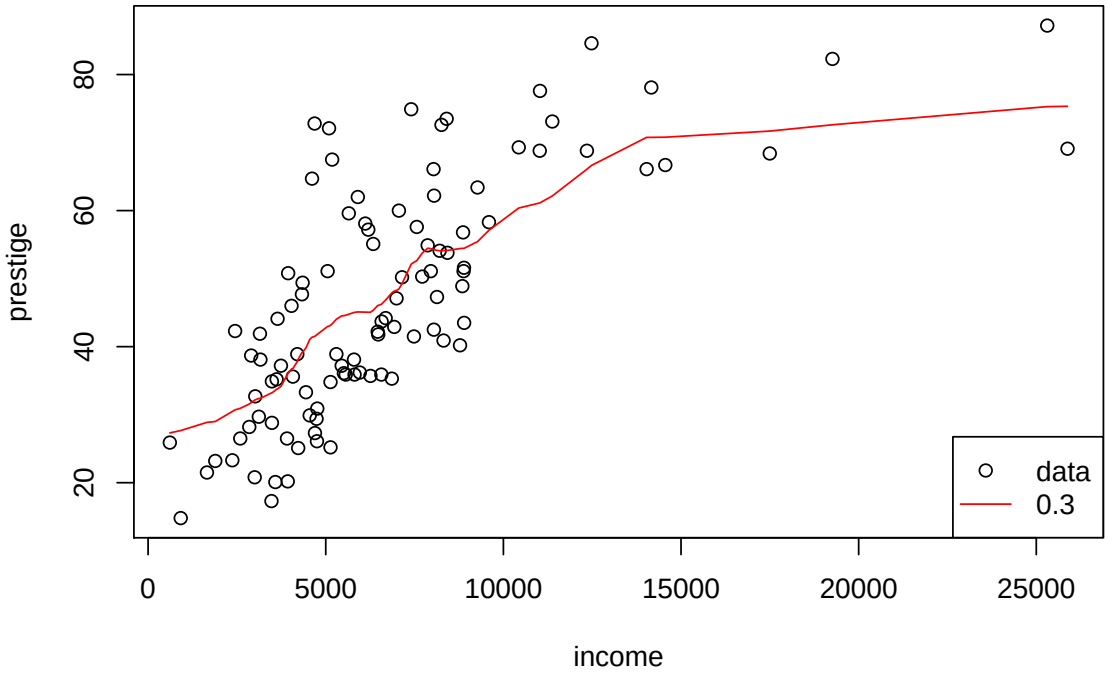

Figure 5: Chosen moving average model with smoothing parameter = 0.3

- Fit a local linear estimator of regression function using function loess() with parameter degree=1. Choose the best value of smoothing parameter span (visually). Draw the fitted curve.

  Fitting models with different values of span:

  ```
  > prestige.mas<-list()
  > for(s in seq(from=0.1,to=0.8,by=0.1)){
  +         prestige.mas[[as.character(s)]]<-loess(prestige~income,Prestige,degree=1,span=s)
  + }
  ```

  It can be seen that small values of smoothing parameter results in the model that tries to fits the data too closely and is over-fitted.

```
> plot(Prestige$income,Prestige$prestige,xlab="income",ylab="prestige")
> colors<-1:length(prestige.mas)
> idx<-0
> for(s in seq(from=0.1,to=0.8,by=0.1)){
+         idx<-idx+1
+         s.char<-as.character(s)
+         sorted_idx<-sort(prestige.mas[[s.char]]$x,index.return=T)$ix
+         lines(prestige.mas[[s.char]]$x[sorted_idx],prestige.mas[[s.char]]$fitted[sorted_idx],
+                         col=colors[idx])
+ }
> legend(x="bottomright",lty=c(-1,rep(1,length(prestige.mas))),col=c(1,colors),
+                 legend=c("data",as.character(seq(from=0.1,to=0.8,by=0.1))),
+                 pch = c(1,rep(-1,length(prestige.mas))))
```

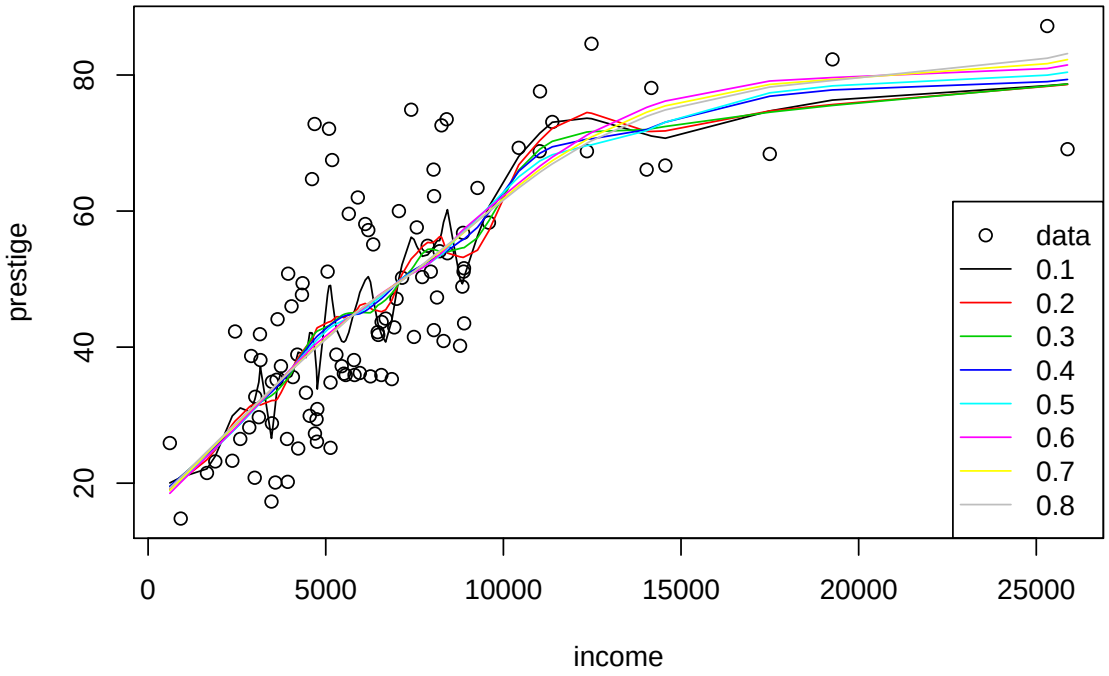

Figure 6: local linear estimator for different values of smoothing parameter

We should choose the model with smother line going through the data points (chosen 0.7):

```
> choosed.smoothing.param<-0.7
> prestige.ma<-prestige.mas[[as.character(choosed.smoothing.param)]]
> plot(Prestige$income,Prestige$prestige,xlab="income",ylab="prestige")
> sorted_idx<-sort(prestige.ma$x,index.return=T)$ix
> lines(prestige.ma$x[sorted_idx],prestige.ma$fitted[sorted_idx],col=2)
> legend(x="bottomright",lty=c(-1,1),col=c(1,2), legend=c("data",as.character(choosed.smoothing.param)),pch =
```

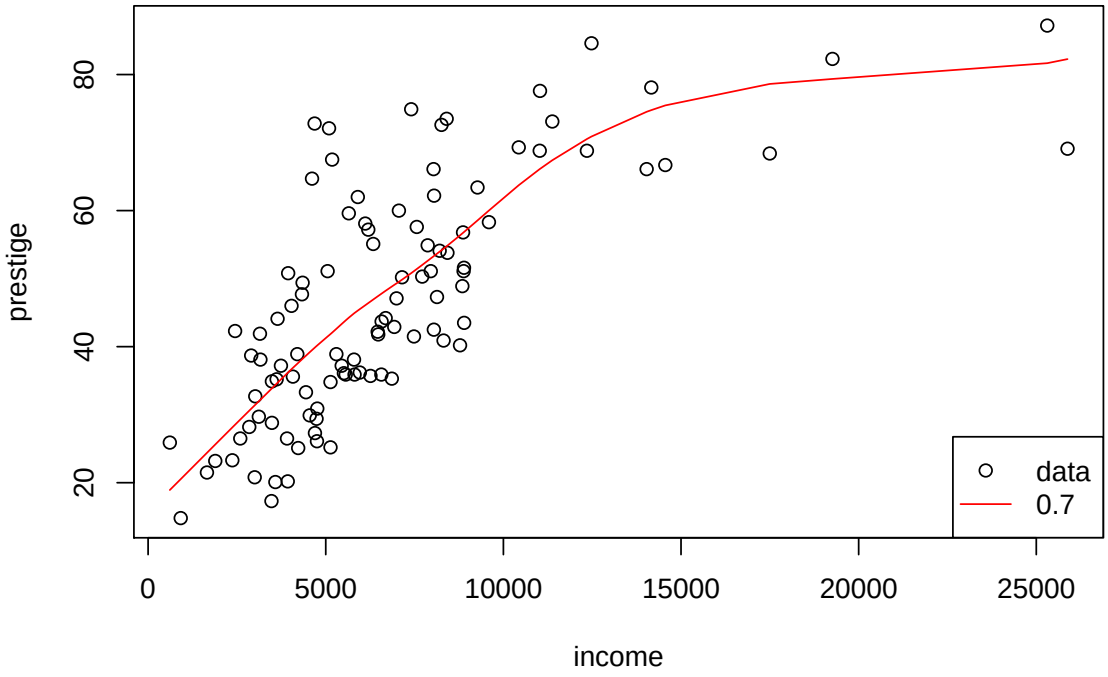

Figure 7: Chosen local linear estimator model with smoothing parameter = 0.7

- Fit a local quadratic estimator of regression function using function loess() with parameter degree=2. Choose the best value of smoothing parameter span (visually). Draw the fitted curve.

  Fitting models with different values of span:

```
> prestige.mas<-list()
> for(s in seq(from=0.1,to=0.8,by=0.1)){
+         prestige.mas[[as.character(s)]]<-loess(prestige~income,Prestige,degree=2,span=s)
+ }
```

  It can be seen that small values of smoothing parameter results in the model that tries to fits the data too closely and is over-fitted.

```
> plot(Prestige$income,Prestige$prestige,xlab="income",ylab="prestige")
> colors<-1:length(prestige.mas)
> idx<-0
> for(s in seq(from=0.1,to=0.8,by=0.1)){
+         idx<-idx+1
+         s.char<-as.character(s)
+         sorted_idx<-sort(prestige.mas[[s.char]]$x,index.return=T)$ix
+         lines(prestige.mas[[s.char]]$x[sorted_idx],prestige.mas[[s.char]]$fitted[sorted_idx],
+                         col=colors[idx])
+ }
> legend(x="bottomright",lty=c(-1,rep(1,length(prestige.mas))),col=c(1,colors),
+                 legend=c("data",as.character(seq(from=0.1,to=0.8,by=0.1))),
+                 pch = c(1,rep(-1,length(prestige.mas))))
```

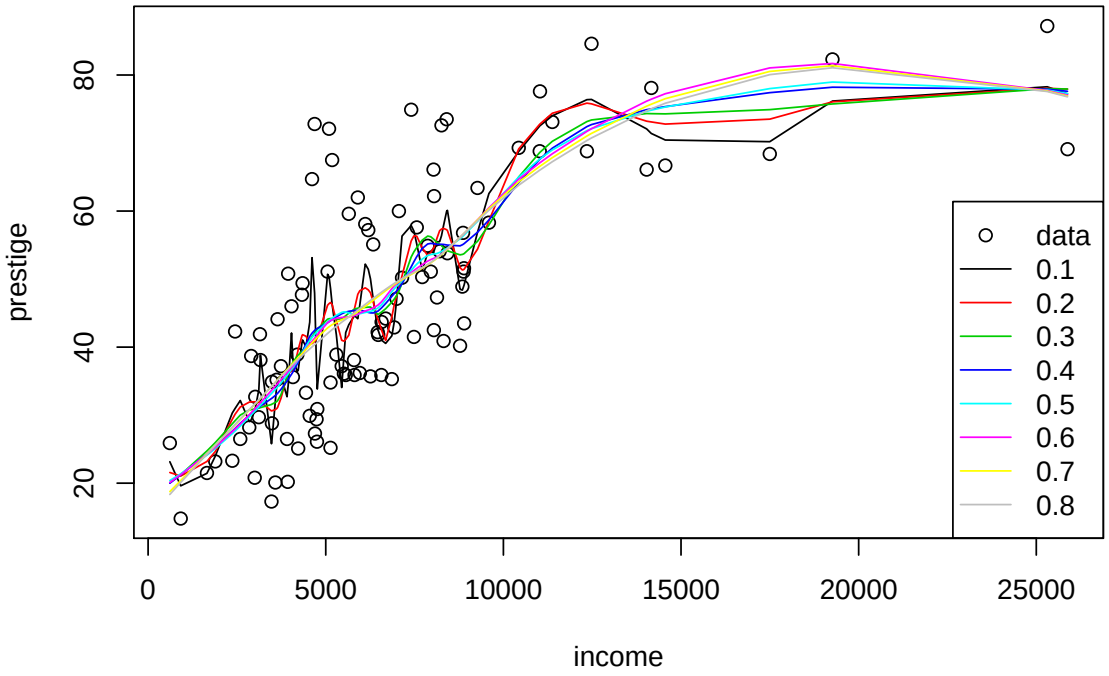

Figure 8: local quadratic estimator model for different values of smoothing parameter

We should choose the model with smother line going through the data points (chosen 0.8):

```
> choosed.smoothing.param<-0.8
> prestige.ma<-prestige.mas[[as.character(choosed.smoothing.param)]]
> plot(Prestige$income,Prestige$prestige,xlab="income",ylab="prestige")
> sorted_idx<-sort(prestige.ma$x,index.return=T)$ix
> lines(prestige.ma$x[sorted_idx],prestige.ma$fitted[sorted_idx],col=2)
> legend(x="bottomright",lty=c(-1,1),col=c(1,2), legend=c("data",as.character(choosed.smoothing.param)),pch =
```

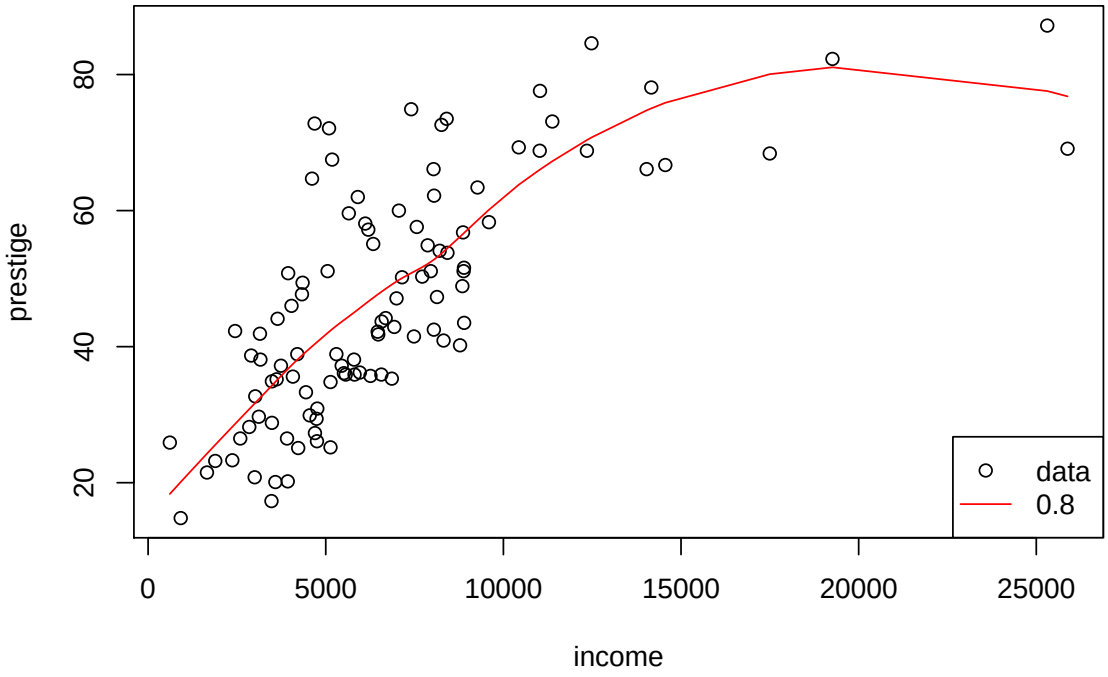

Figure 9: Chosen local quadratic estimator model with smoothing parameter = 0.8

- Fit a local polynomial estimator of regression function using normal kernel K (function locpoly() in package KernSmooth with parameter kernel='normal'). Choose the most adequate degree of the polynomial and value of the smoothing parameter (bandwidth) in this case (visually). Draw the fitted curve.

  For this method we have to change the scale of income, computing models for different bandwidth and degree:

```
> library(KernSmooth)
> Prestige$incomeScaled<-Prestige$income/1000
> locpoly.models<-list()
> for(deg in 1:4){
+         locpoly.models[[as.character(deg)]]=list()
+         for(band in seq(0.1,0.8,0.1)){
+                 locpoly.models[[as.character(deg)]][[as.character(band)]]<-
+                                 locpoly(x=Prestige$incomeScaled,y=Prestige$prestige,kernel="normal",
+                                                 bandwidth=band,degree=deg)
+         }
+ }
```

  Plotting those models:

```
> plot(Prestige$incomeScaled,Prestige$prestige,xlab="income",ylab="prestige")
> deg=1
> bands<-seq(0.1,0.8,0.1)
> colors<-1:length(bands)
> idx<-0
> for(band in bands){
+         idx<-idx+1
+         lines(locpoly.models[[as.character(deg)]][[as.character(band)]],col=colors[idx])
+ }
> legend(x="bottomright",lty=c(-1,rep(1,length(bands))),col=c(1,colors),
+                 legend=c("data",as.character(bands)),pch = c(1,rep(-1,length(bands))))
>
```

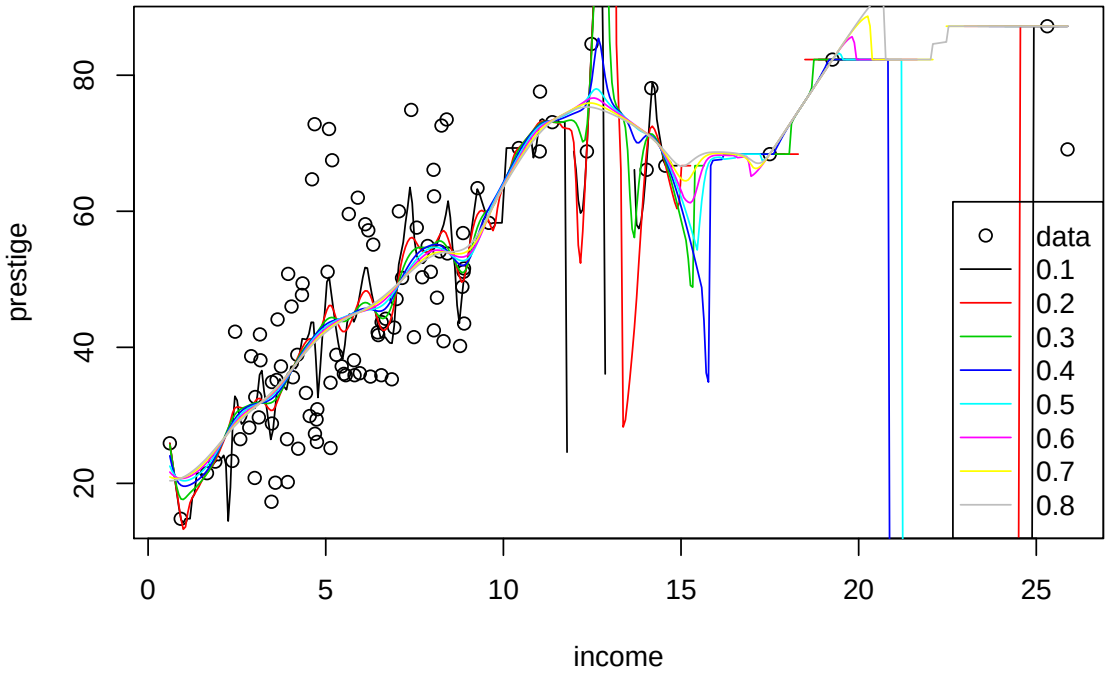

Figure 10: locpoly models with degree 1 and bandwidth from 0.1 to 0.8

```
> plot(Prestige$incomeScaled,Prestige$prestige,xlab="income",ylab="prestige")
> deg=2
> bands<-seq(0.1,0.8,0.1)
> colors<-1:length(bands)
> idx<-0
> for(band in bands){
+         idx<-idx+1
+         lines(locpoly.models[[as.character(deg)]][[as.character(band)]],col=colors[idx])
+ }
> legend(x="bottomright",lty=c(-1,rep(1,length(bands))),col=c(1,colors),
+                 legend=c("data",as.character(bands)),pch = c(1,rep(-1,length(bands))))
>
```

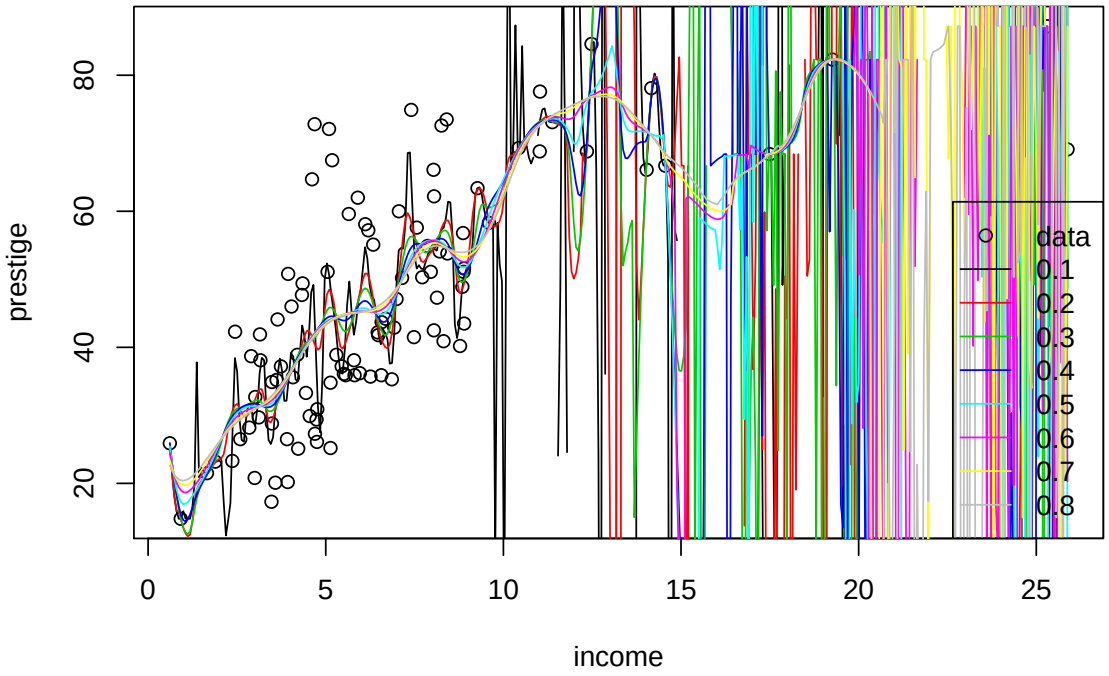

Figure 11: locpoly models with degree 2 and bandwidth from 0.1 to 0.8

```
> plot(Prestige$incomeScaled,Prestige$prestige,xlab="income",ylab="prestige")
> deg=3
> bands<-seq(0.1,0.8,0.1)
> colors<-1:length(bands)
> idx<-0
> for(band in bands){
+         idx<-idx+1
+         lines(locpoly.models[[as.character(deg)]][[as.character(band)]],col=colors[idx])
+ }
> legend(x="bottomright",lty=c(-1,rep(1,length(bands))),col=c(1,colors),
+                 legend=c("data",as.character(bands)),pch = c(1,rep(-1,length(bands))))
>
```

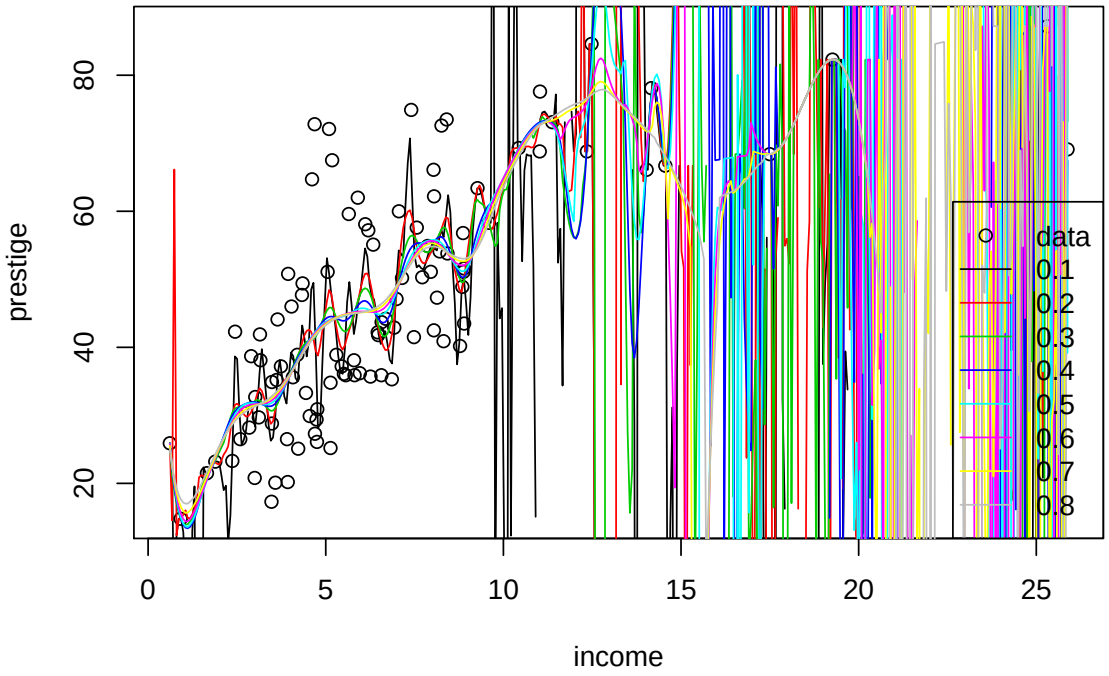

Figure 12: locpoly models with degree 3 and bandwidth from 0.1 to 0.8

```
> plot(Prestige$incomeScaled,Prestige$prestige,xlab="income",ylab="prestige")
> deg=4
> bands<-seq(0.1,0.8,0.1)
> colors<-1:length(bands)
> idx<-0
> for(band in bands){
+         idx<-idx+1
+         lines(locpoly.models[[as.character(deg)]][[as.character(band)]],col=colors[idx])
+ }
> legend(x="bottomright",lty=c(-1,rep(1,length(bands))),col=c(1,colors),
+                 legend=c("data",as.character(bands)),pch = c(1,rep(-1,length(bands))))
>
```

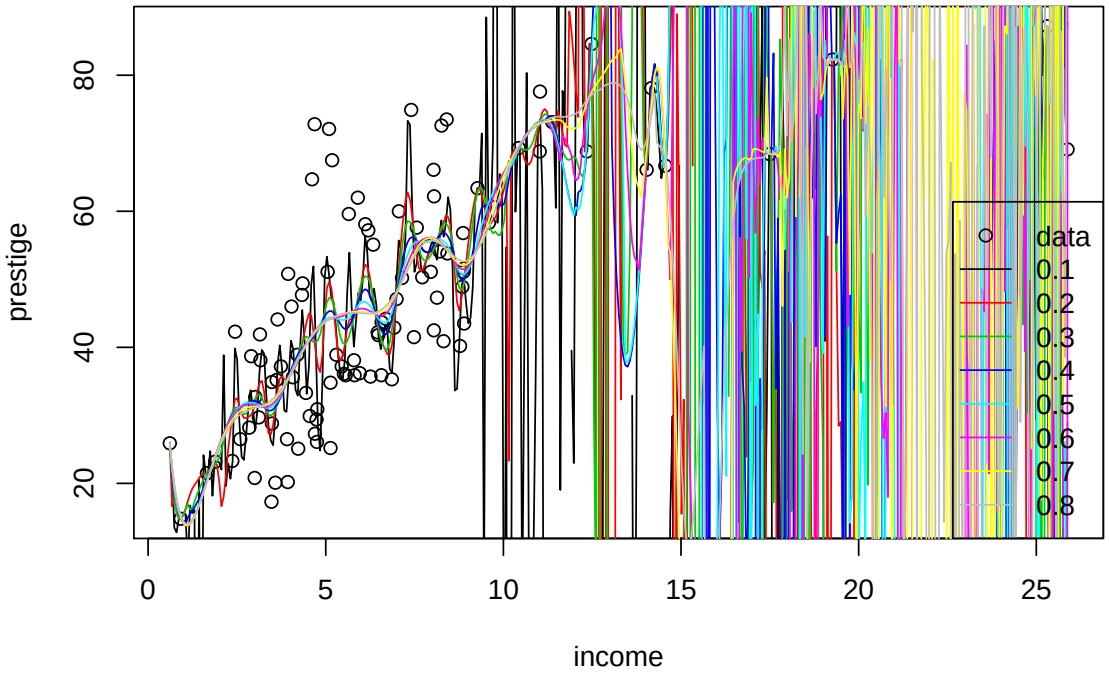

Figure 13: locpoly models with degree 4 and bandwidth from 0.1 to 0.8

Choosing model with degree 1 and bandwidth 0.8 but really we should remove outlying values in income variable.

```
> plot(Prestige$incomeScaled,Prestige$prestige,xlab="income",ylab="prestige")
> deg=1
> bands<-0.8
> colors<-1:length(bands)
> idx<-0
> for(band in bands){
+         idx<-idx+1
+         lines(locpoly.models[[as.character(deg)]][[as.character(band)]],col=colors[idx])
+ }
> legend(x="bottomright",lty=c(-1,rep(1,length(bands))),col=c(1,colors),
+                 legend=c("data",as.character(bands)),pch = c(1,rep(-1,length(bands))))
>
```

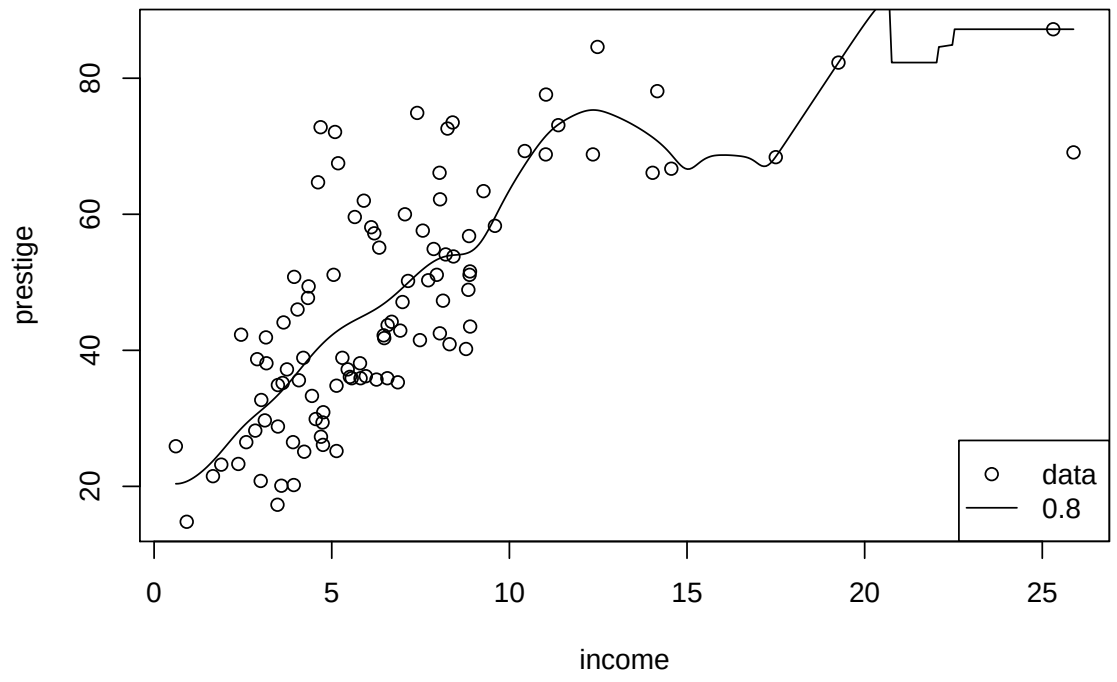

Figure 14: locpoly models with degree 1 and bandwidth 0.8

- Fit a cubic spline to the data using function smooth.spline().

  ```
  > prestige.smooth.spline<-smooth.spline(x=Prestige$income,y=Prestige$prestige)
  ```

  Ploting the data and the model:

```
> plot(Prestige$income,Prestige$prestige,xlab="income",ylab="prestige")
> lines(prestige.smooth.spline,col="blue")
> legend(x="bottomright",lty=c(-1,1),col=c("black","blue"),
+                 legend=c("Prestige data","Fitted model"),pch = c(1,-1))
```

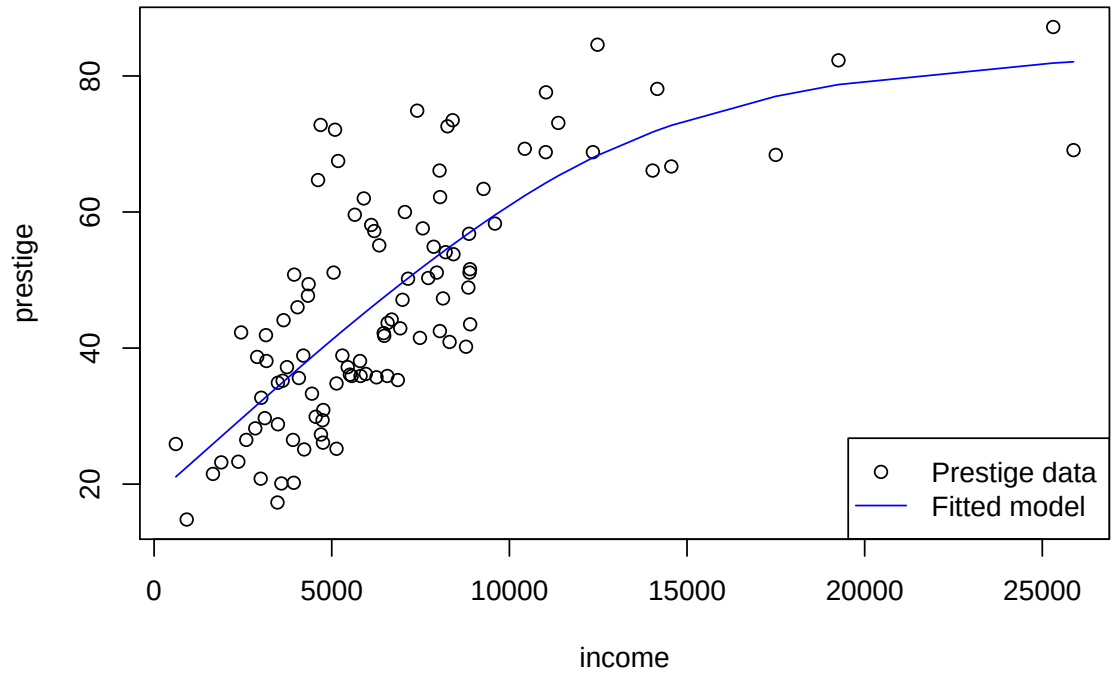

Figure 15: smooth.spline model

Exercise 3.

Generate a random sample which follows the equation:

$$y_i = sin(4x_i) + \epsilon_i \ , \ \epsilon_i \sim N(0, \sigma), \sigma = 1/3,$$

for i = 1, . . . , 100 and xi are uniofrmly distributed in interval [0, 1.6].

```
> set.seed(103)
> xi<-runif(100,0,1.6)
> eps.i<-rnorm(100,0,1/3)
> yi<-sin(4*xi)+eps.i
```

Fit a local polynomial estimator to the data choosing the appropriate degree and value of smoothing parameter.

```
> locpoly.models<-list()
> for(deg in 1:4){
+         locpoly.models[[as.character(deg)]]=list()
+         for(band in seq(0.1,0.8,0.1)){
+                 locpoly.models[[as.character(deg)]][[as.character(band)]]<-
+                                 locpoly(x=xi,y=yi,kernel="normal",bandwidth=band,degree=deg)
+         }
+ }
```

Plotting those models:

```
> plot(xi,yi,xlab="x",ylab="y")
> deg=1
> bands<-seq(0.1,0.8,0.1)
> colors<-1:length(bands)
> idx<-0
> for(band in bands){
+         idx<-idx+1
+         lines(locpoly.models[[as.character(deg)]][[as.character(band)]],col=colors[idx])
+ }
> legend(x="topright",lty=c(-1,rep(1,length(bands))),col=c(1,colors),
+                 legend=c("data",as.character(bands)),pch = c(1,rep(-1,length(bands))))
>
```

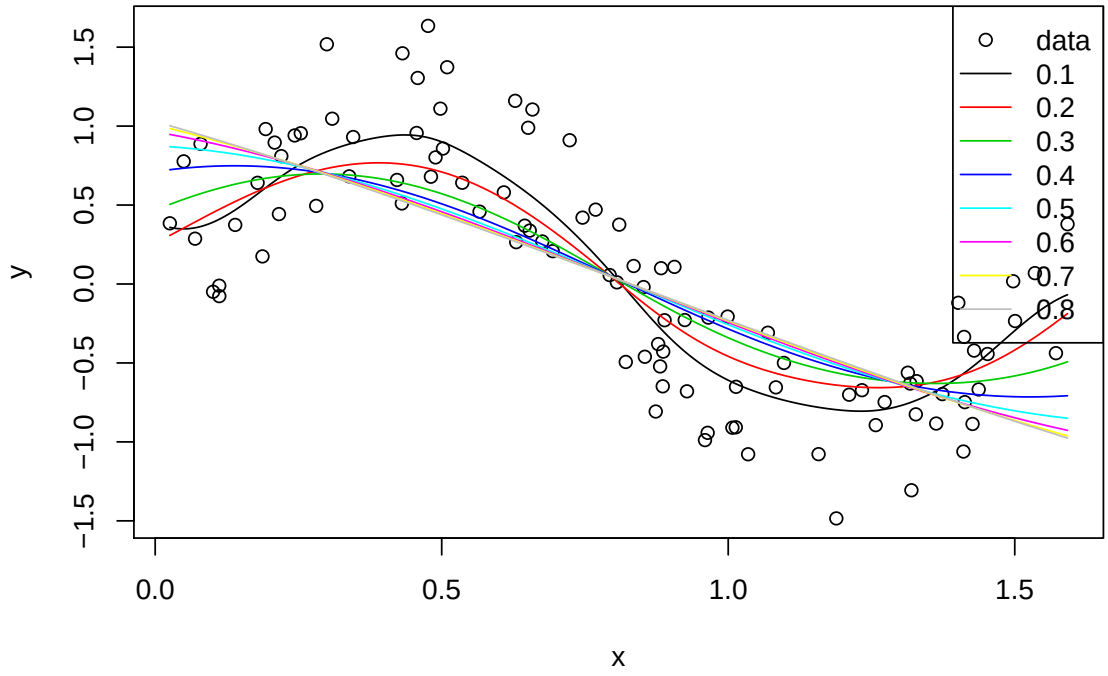

Figure 16: locpoly models with degree 1 and bandwidth from 0.1 to 0.8

```
> plot(xi,yi,xlab="x",ylab="y")
> deg=2
> bands<-seq(0.1,0.8,0.1)
> colors<-1:length(bands)
> idx<-0
> for(band in bands){
+         idx<-idx+1
+         lines(locpoly.models[[as.character(deg)]][[as.character(band)]],col=colors[idx])
+ }
> legend(x="topright",lty=c(-1,rep(1,length(bands))),col=c(1,colors),
+                 legend=c("data",as.character(bands)),pch = c(1,rep(-1,length(bands))))
>
```

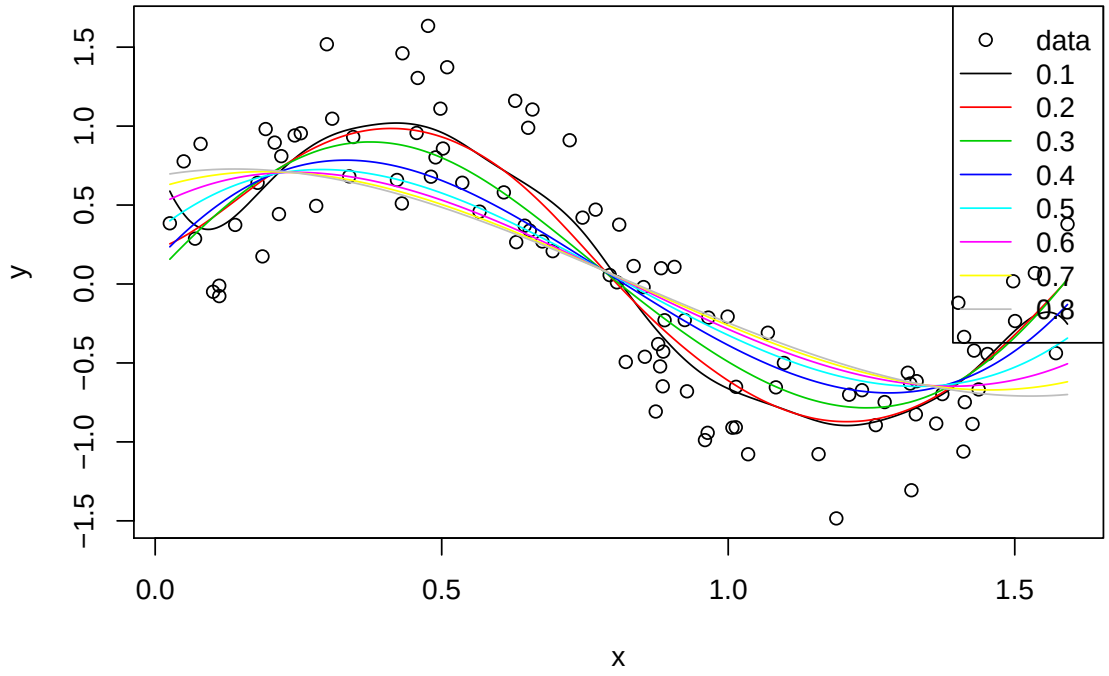

Figure 17: locpoly models with degree 2 and bandwidth from 0.1 to 0.8

```
> plot(xi,yi,xlab="x",ylab="y")
> deg=3
> bands<-seq(0.1,0.8,0.1)
> colors<-1:length(bands)
> idx<-0
> for(band in bands){
+         idx<-idx+1
+         lines(locpoly.models[[as.character(deg)]][[as.character(band)]],col=colors[idx])
+ }
> legend(x="topright",lty=c(-1,rep(1,length(bands))),col=c(1,colors),
+                 legend=c("data",as.character(bands)),pch = c(1,rep(-1,length(bands))))
>
```

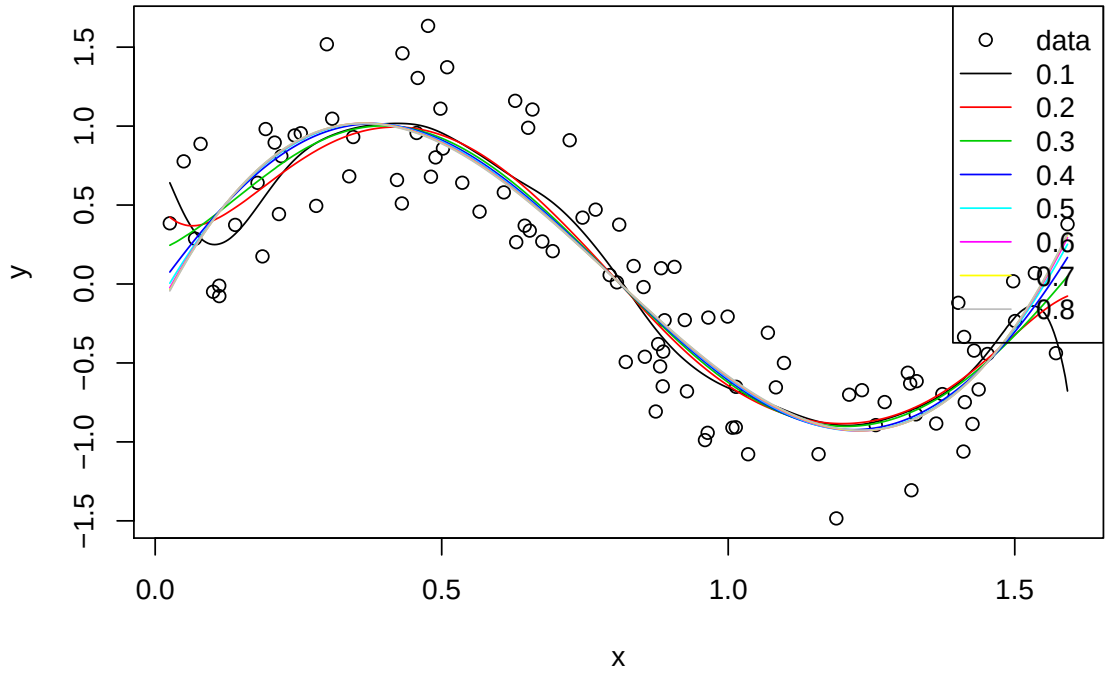

Figure 18: locpoly models with degree 3 and bandwidth from 0.1 to 0.8

```
> plot(xi,yi,xlab="x",ylab="y")
> deg=4
> bands<-seq(0.1,0.8,0.1)
> colors<-1:length(bands)
> idx<-0
> for(band in bands){
+         idx<-idx+1
+         lines(locpoly.models[[as.character(deg)]][[as.character(band)]],col=colors[idx])
+ }
> legend(x="topright",lty=c(-1,rep(1,length(bands))),col=c(1,colors),
+                 legend=c("data",as.character(bands)),pch = c(1,rep(-1,length(bands))))
>
```

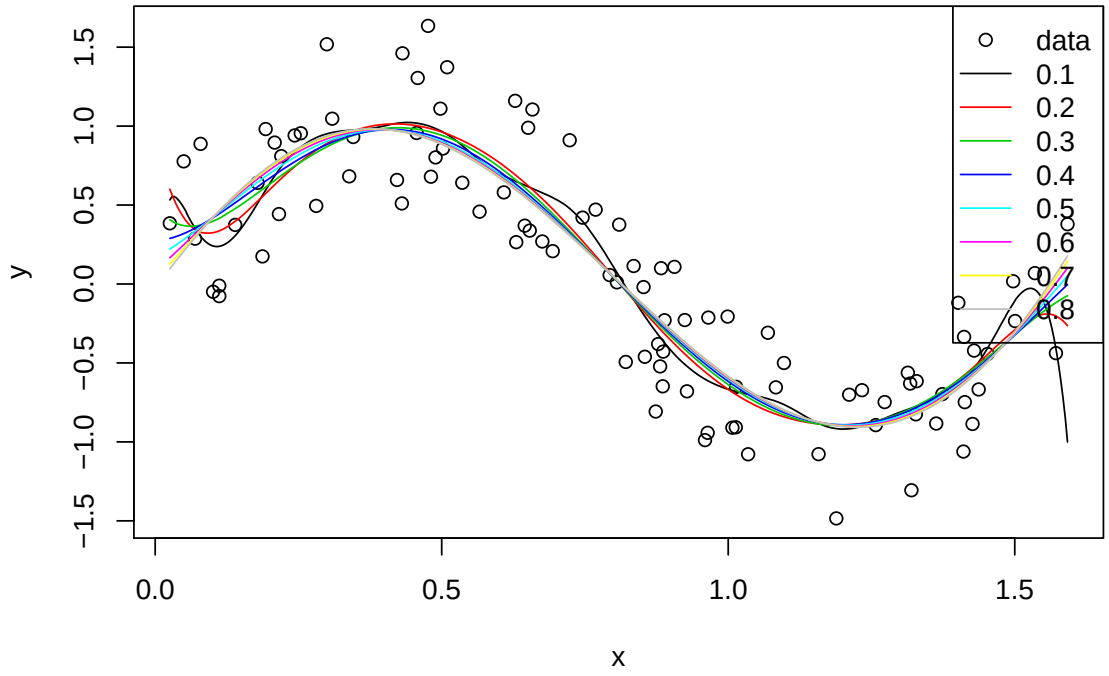

Figure 19: locpoly models with degree 4 and bandwidth from 0.1 to 0.8

Based on the graphs choosing model with degree 2 and bandwidth 0.2:

```
> plot(xi,yi,xlab="x",ylab="y")
> lines(locpoly.models[[as.character(2)]][[as.character(0.2)]],col="blue")
> legend(x="topright",lty=c(-1,1),col=c(1,"blue"),
+                 legend=c("data",as.character(0.2)),pch = c(1,-1))
>
```

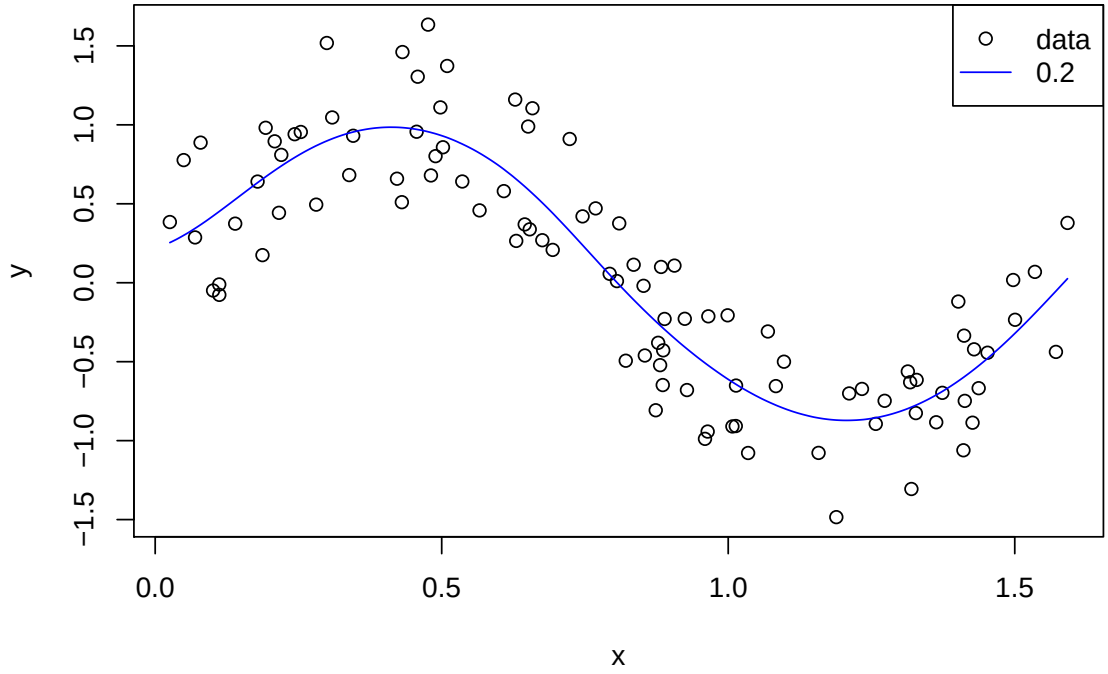

Figure 20: chosen model with degree 2 and bandwidth 0.2