Lab 4. Flexible Modelling

Smoothing

"Smoothing" describes a broad umbrella of techniques which fit a **nonparametric** model to covariate-outcome data. **Parametric** models are models with explicit functional forms for means and variances. They require assumptions and in return allow you to summarize a data relationship in terms of parameters (for example, β_j in a linear regression summarizes the relationship between covariate x_j and the outcome). **Nonparametric** models require fewer (if any) assumptions and are much more flexible, but may be hard to summarize. They are useful for both **visualization** and **prediction**.

Note:
You are not required to know how to generate the figures in {Section 1. Scatterplot smoothing} in any language (except LOWESS curve). However, the code can help you understand what those smoothing methods are doing. You can play with it if you are interested in more details, otherwise knowing the general idea behind the method is enough for this course!

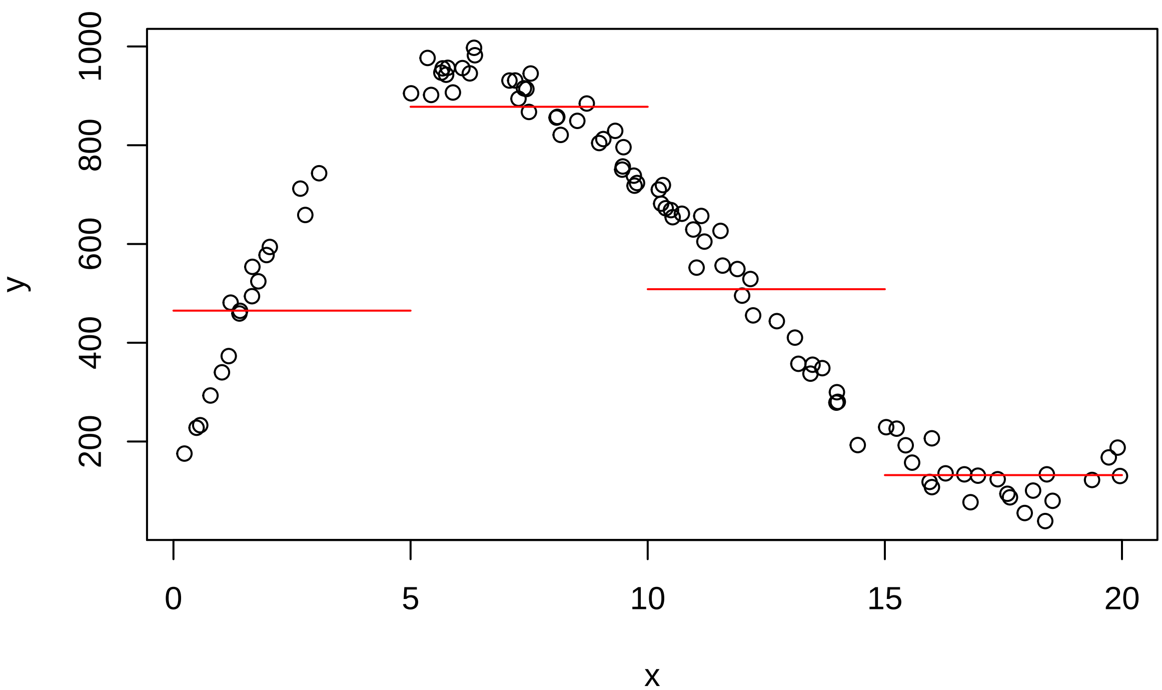
1.Scatterplot Smoothing

We will generate the data for this lab in R. The same data is saved in "lab4-1.csv" if you want to follow along in Stata or SAS.

1.1 Bin Smoother

Create categories for x variables and average Y over each category (piecewise constant)

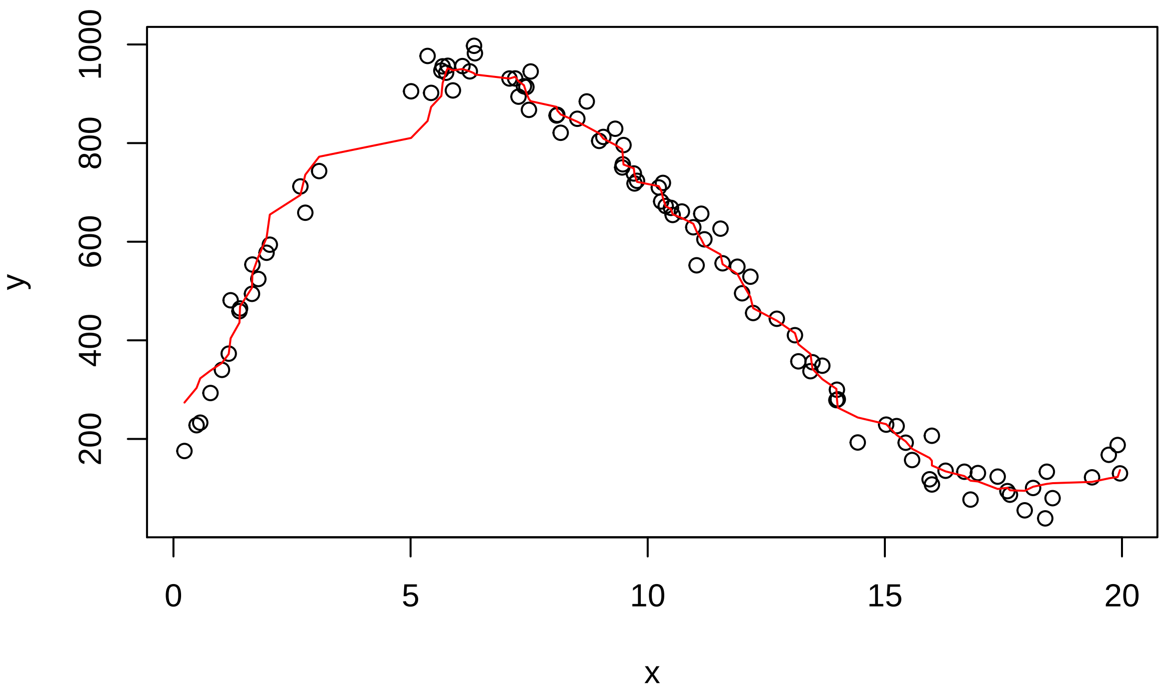
```
set.seed(210)
x=c(runif(25,0,6),runif(50,6,14),runif(25,14,20))
x=sort(x)
y=-(x-3)^3+(x-6)^3+(x-9)^3+1000+rnorm(100,0,30)
bin1=mean(y[x>0 & x<=5])
bin2=mean(y[x>5 & x<=10])
bin3=mean(y[x>10 & x<=15])
bin4=mean(y[x>15 & x<=20])
plot(x,y)
lines(c(0,5),c(bin1,bin1),col="red")
lines(c(5,10),c(bin2,bin2),col="red")
lines(c(10,15),c(bin3,bin3),col="red")
lines(c(15,20),c(bin4,bin4),col="red")
```



1.2 Running mean Smoother

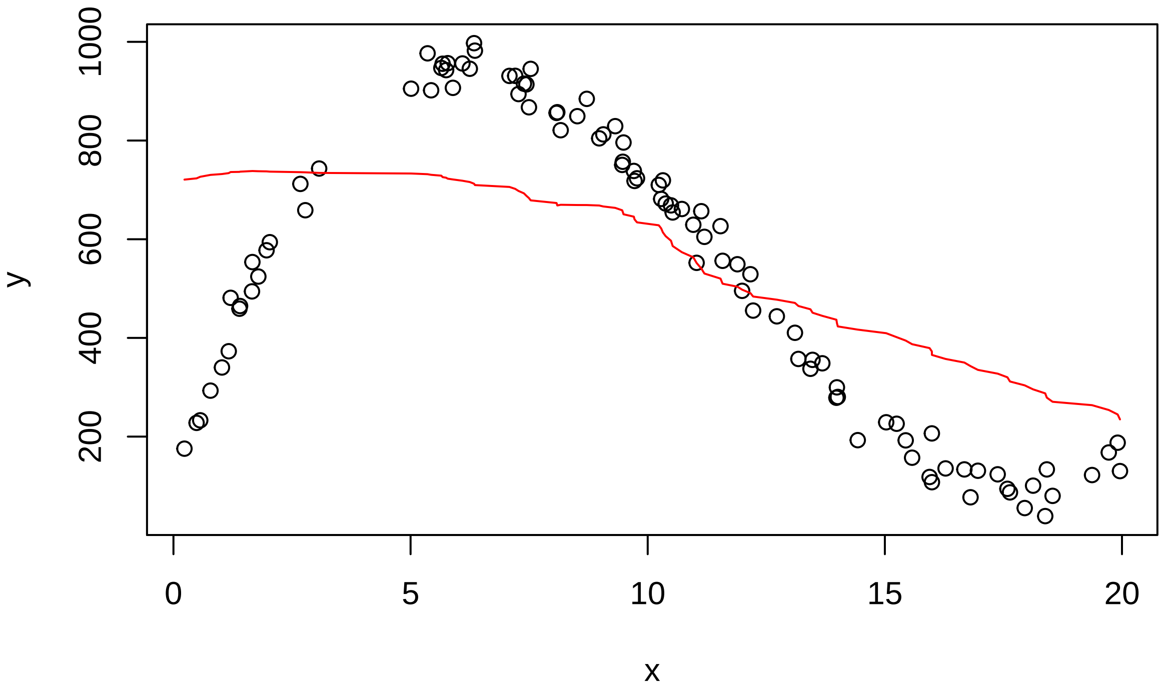
Mean of Y over a moving neighborhood of x, with a relatively small bandwidth (which is 11 in this case)

```
running_mean=c()
for(i in 1:100){
  neighbor=max(1,i-5):min(i+5,100)
  running_mean=c(running_mean,mean(y[neighbor]))
}
plot(x,y)
lines(x,running_mean,col="red")
```



If we choose a large bandwidth, say, 75, we cannot capture the shape of the data

```
running_mean=c()
for(i in 1:100){
  neighbor=max(1,i-37):min(i+37,100)
  running_mean=c(running_mean,mean(y[neighbor]))
}
plot(x,y)
lines(x,running_mean,col="red")
```

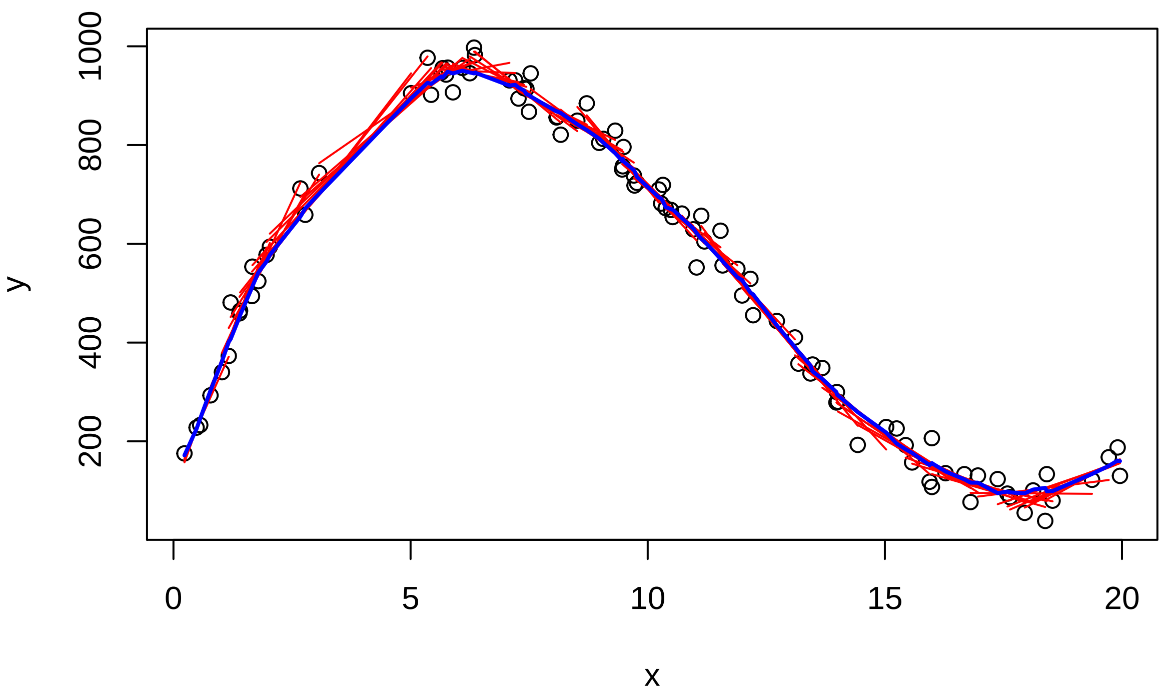


1000
800
600
400
200
y
0
5
10
15
20
x

1.3 Running line Smoother

Linear fit of Y over a moving neighborhood of x, with a relatively small bandwidth (which is 11 in this case)

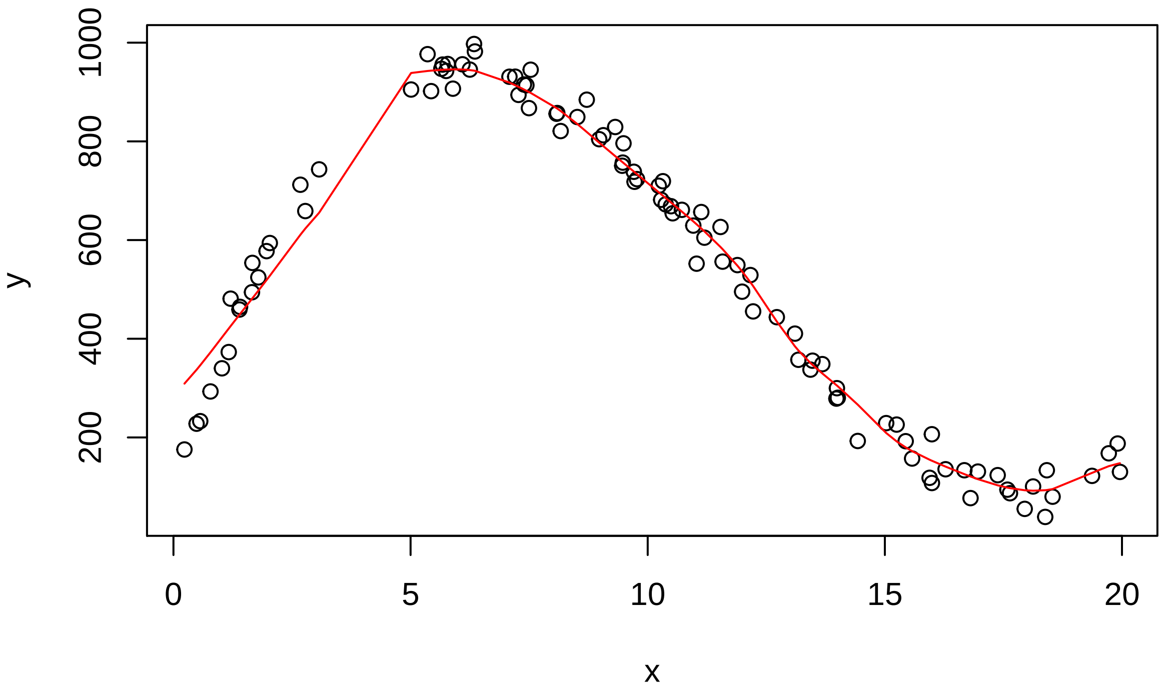
```
plot(x,y)
midpoint=c()
for(i in 1:100){
  neighbor=max(1,i-5):min(i+5,100)
  mod=lm(y[neighbor]~x[neighbor])
  midpoint[i]=sum(coef(mod)*c(1,x[i]))
  lines(x[neighbor],fitted(mod),col="red")
}
lines(x,midpoint,col="blue",lw=2)
```



1.4 Kernel Smoother

Locally **weighted** running mean smoother of Y over a moving neighborhood of x (the kernel has higher weights the closer you are to the middle of the neighborhood, while bin smoother calculates **equally** weighted mean), with a relatively large bandwidth (which is 75% of the data in this case)

```
kernel_smooth=c()
for(i in 1:100){
  neighbor=max(1,i-37):min(i+37,100)
  weight=exp(-(x[neighbor]-x[i])^2)/sum(exp(-(x[neighbor]-x[i])^2)) ## Gaussian Kernel
  kernel_smooth=c(kernel_smooth,sum(weight*y[neighbor]))
}
plot(x,y)
lines(x,kernel_smooth,col="red")
```

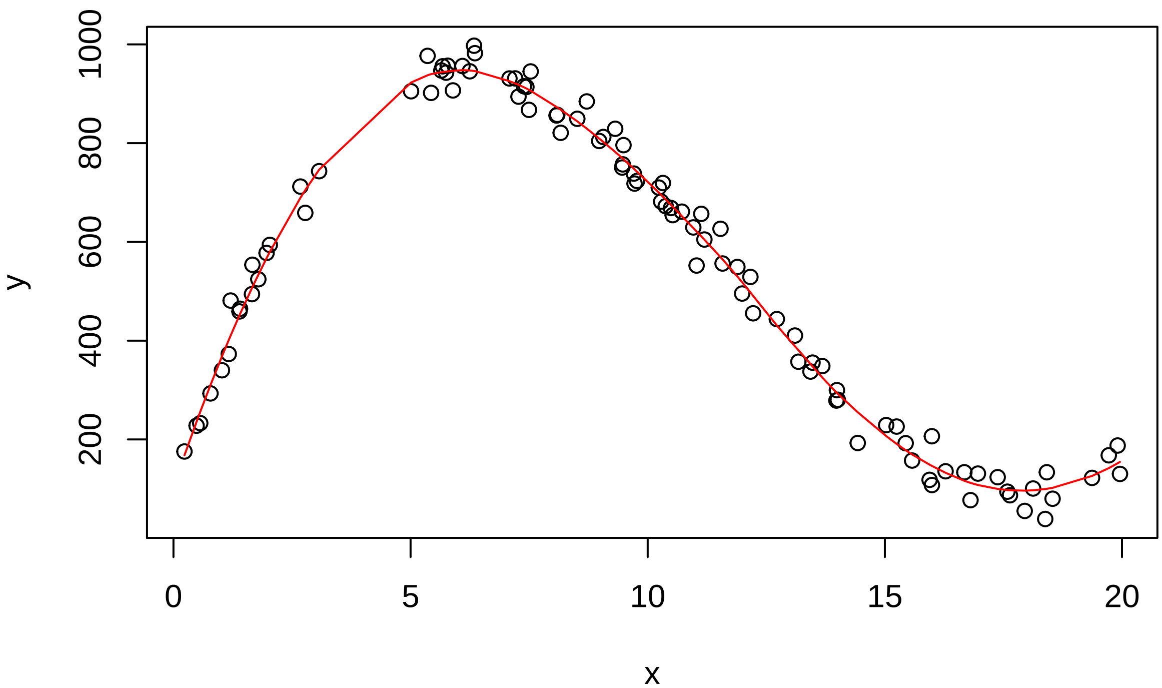


1.5 LOWESS

Locally weighted running line smoother of Y over a moving neighborhood of x (the kernel has higher weights the closer you are to the middle of the neighborhood), with a relatively large bandwidth (e.g. 40% of the data).

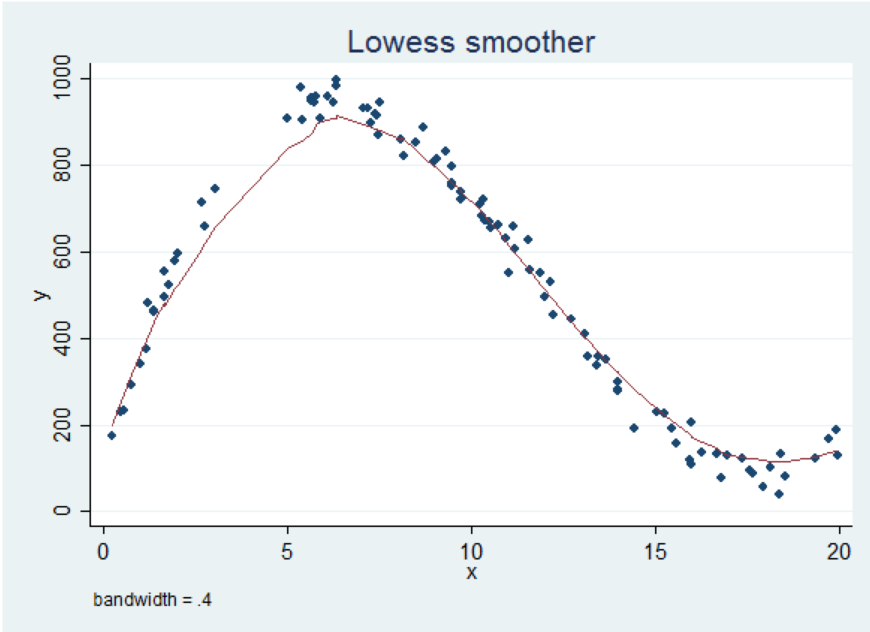
The smoother span ('span' option in the following code) is the proportion of points used in smoothing at each value. Larger values give more smoothness. It is same as 'bwidth' in Stata.

```
plot(x,y)
fit=loess(y~x,span=0.4)
lines(x,fitted(fit),col="red")
```



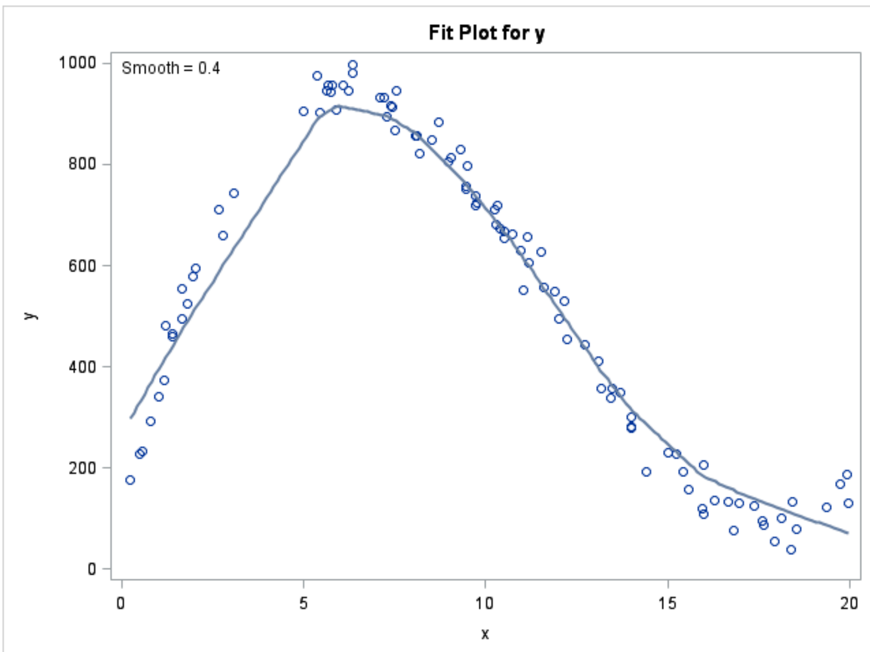
In Stata, if you want to use 40% of data for smoothing at each point, you can use:

```
lowess y x, bwidth(0.4)  ## default is 0.8
```



In SAS,

```
proc loess data = dat;
model y = x/smooth=0.4; ## default is chosen by AICC
run;
```

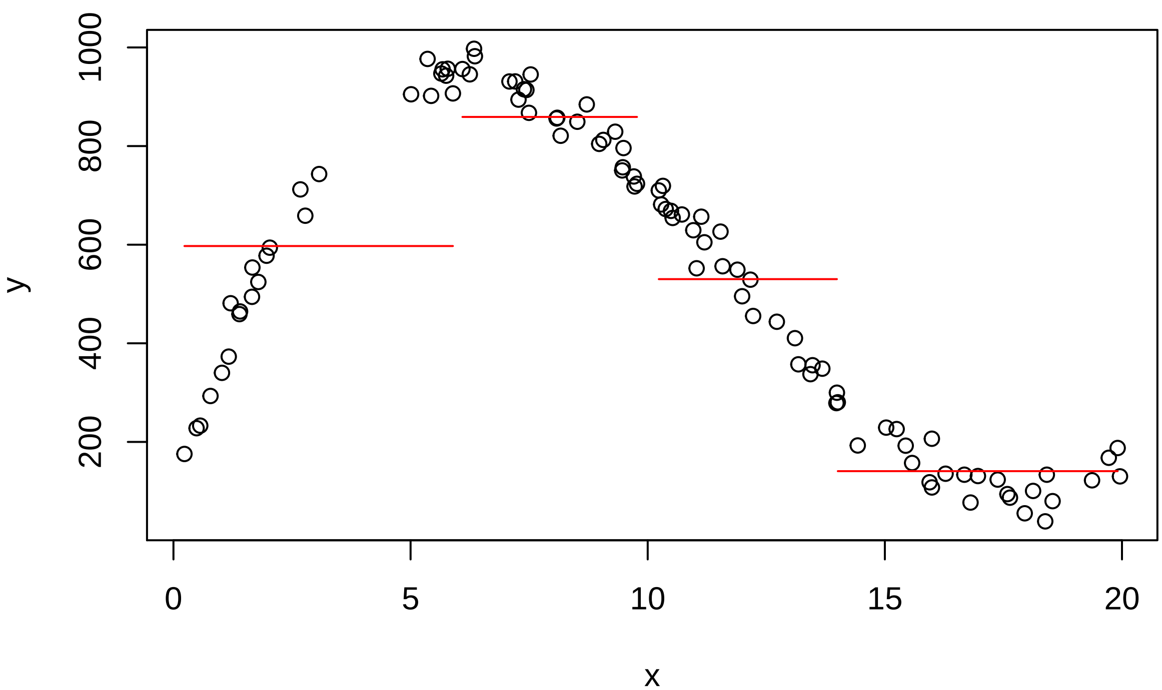


2.Splines

2.1 Piecewise Constant Splines

Same as a bin smoother, discontinuous at knots. We'll specify the knots by ourselves.

```
library(splines2)
mod3=lm(y~bSpline(x,knots=quantile(x,c(0.25,0.5,0.75)),degree=0))
plot(x,y)
lines(x[1:25],fitted(mod3)[1:25],type='l',col="red")
lines(x[26:50],fitted(mod3)[26:50],type='l',col="red")
lines(x[51:75],fitted(mod3)[51:75],type='l',col="red")
lines(x[76:99],fitted(mod3)[76:99],type='l',col="red")
```

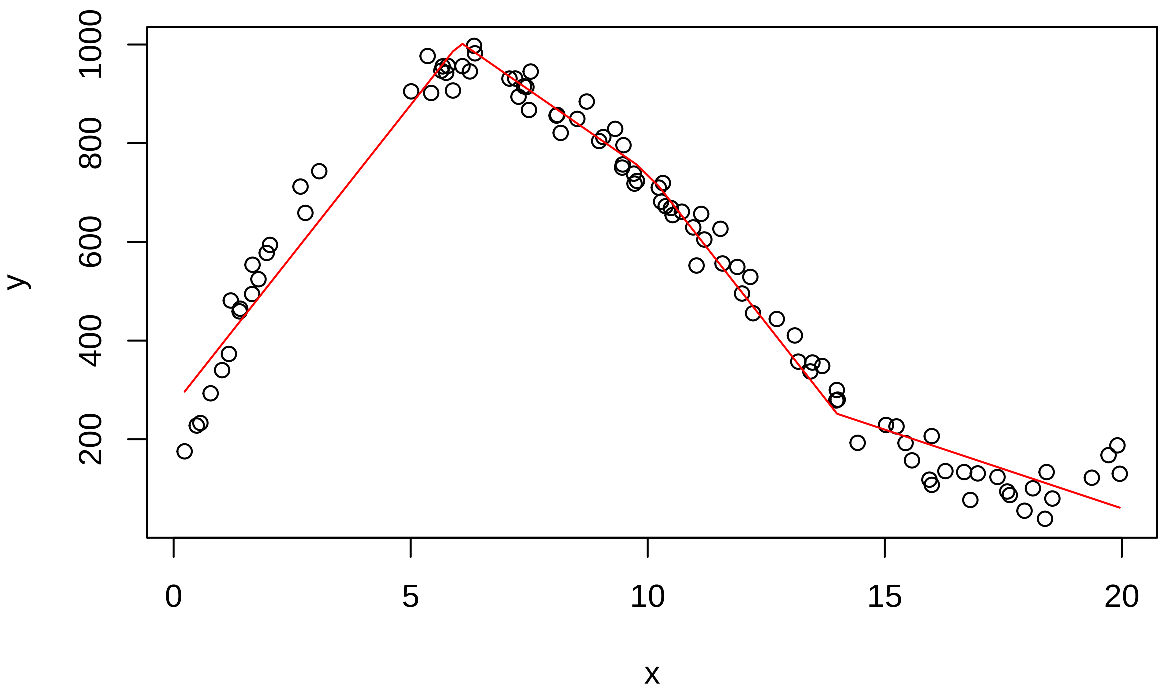


Alternatively to specifying the knots ourselves, we can specify only the number of knots with the `df` argument and allow `bSpline` to pick the knot locations.

2.2 Piecewise Linear Splines

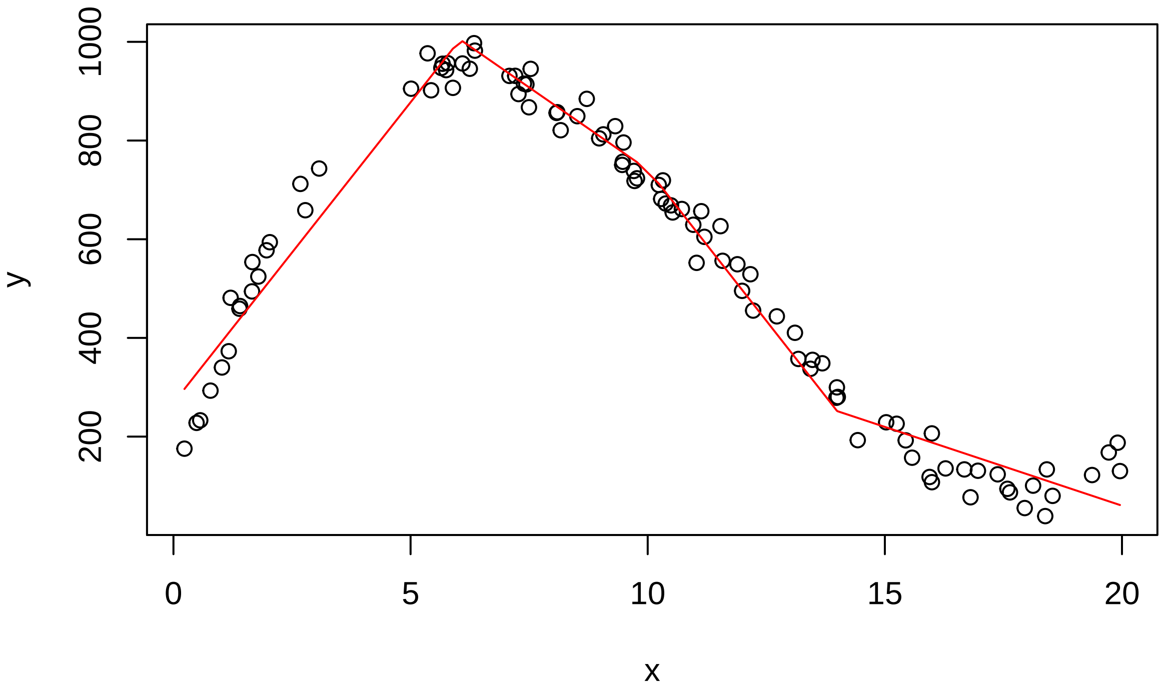
can be continuous at knots

```
mod4=lm(y~bSpline(x,df=4,degree=1))
plot(x,y)
lines(x,fitted(mod4),col="red")
```



Similar to constant spline, you can specify the knots you want

```
mod5=lm(y~bSpline(x,knots=quantile(x,c(0.25,0.5,0.75)),degree=1))
plot(x,y)
lines(x,fitted(mod5),col="red")
```

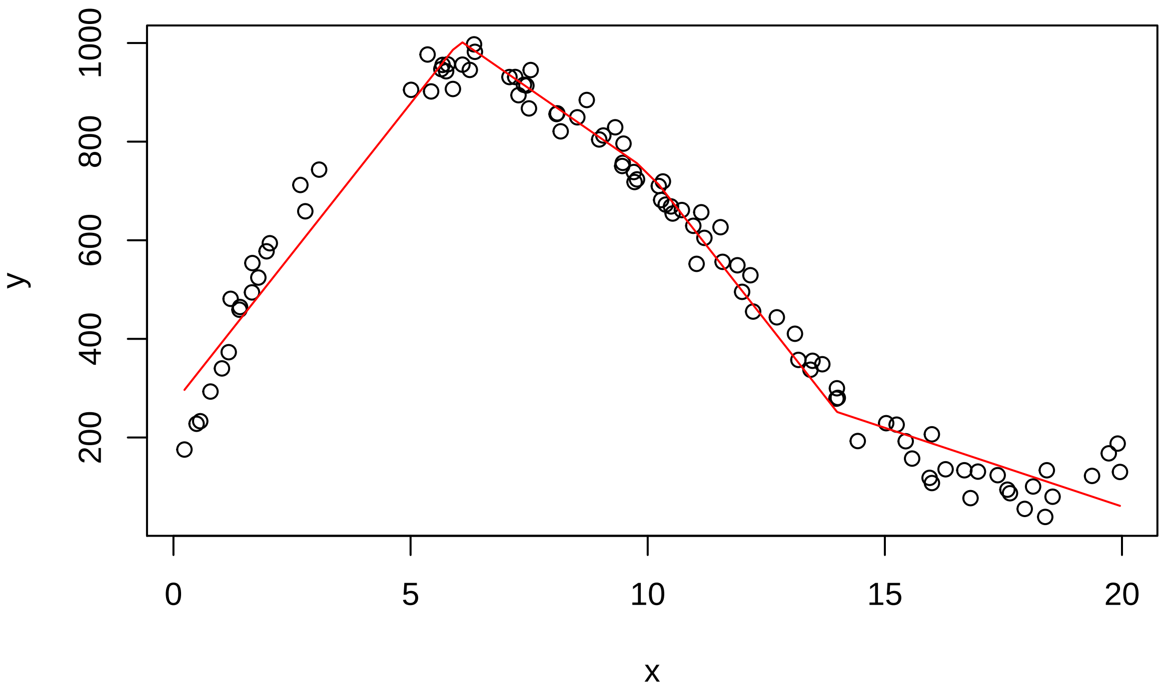



```
temp=summary(mod5)
rownames(temp$coefficients)=c("intercept","Spline1",'Spline2','Spline3','Spline4')
temp
```

```
##
## Call:
## lm(formula = y ~ bSpline(x, knots = quantile(x, c(0.25, 0.5,
##     0.75)), degree = 1))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -120.973  -29.505   -1.807   30.811  124.741
##
## Coefficients:
##           Estimate Std. Error t value Pr(>|t|)
## intercept   296.54      15.80  18.774  < 2e-16 ***
## Spline1     708.10      23.14  30.605  < 2e-16 ***
## Spline2     444.37      19.36  22.956  < 2e-16 ***
## Spline3     -44.80      21.29  -2.104    0.038 *
## Spline4    -235.20      24.05  -9.778 4.93e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 49.9 on 95 degrees of freedom
## Multiple R-squared:  0.974,  Adjusted R-squared:  0.9729
## F-statistic: 889.2 on 4 and 95 DF,  p-value: < 2.2e-16
```

It is equivalent to fit a linear model with intercept, x, $(x - Q_{0.25})_+$, $(x - Q_{0.5})_+$, $(x - Q_{0.75})_+$, where $(x - a)_+$ = max(x-a,0). Let's check it!

```r
x1=x-quantile(x,0.25); x1[x1<0]=0
x2=x-quantile(x,0.50); x2[x2<0]=0
x3=x-quantile(x,0.75); x3[x3<0]=0
mod6=lm(y~x+x1+x2+x3)
plot(x,y)
lines(x,fitted(mod6),col="red")
```




```r
summary(mod6)
```

```
##
## Call:
## lm(formula = y ~ x + x1 + x2 + x3)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -120.973  -29.505   -1.807   30.811  124.741
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  268.317     16.568   16.19  < 2e-16 ***
## x            121.840      3.981   30.61  < 2e-16 ***
## x1          -188.431      8.161  -23.09  < 2e-16 ***
## x2           -56.013      9.152   -6.12 2.08e-08 ***
## x3            90.681      8.786   10.32  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 49.9 on 95 degrees of freedom
## Multiple R-squared:  0.974,  Adjusted R-squared:  0.9729
## F-statistic: 889.2 on 4 and 95 DF,  p-value: < 2.2e-16
```

Question: What are the slopes for four pieces?
Answer: The slopes for four pieces are 121, 121-188, 121-188-56 and 121-188-56+90 respectively.

In Stata,

```
mkspline sx1 6 sx2 10 sx3 14 sx4 = x
regress y sx*
OR
mkspline sx 3 = x, pctile displayknots
regress y sx*
OR
gen x1=cond(x > 6, x - 6, 0)
gen x2=cond(x > 10, x - 10, 0)
gen x3=cond(x > 14, x - 14, 0)
regress y x x1 x2 x3
predict fitted_spline
twoway scatter y x || line fitted_spline x
```

In SAS,

```
proc transreg data=dat ss2 short;
   model identity(y) = pspline(x / nknots=3 degree = 1);
run;
```

The TRANSREG Procedure Hypothesis Tests for Identity(y)

Univariate ANOVA Table Based on the Usual Degrees of Freedom					
Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	4	8854623	2213656	888.41	<.0001
Error	95	236712	2492		
Corrected Total	99	9091335			

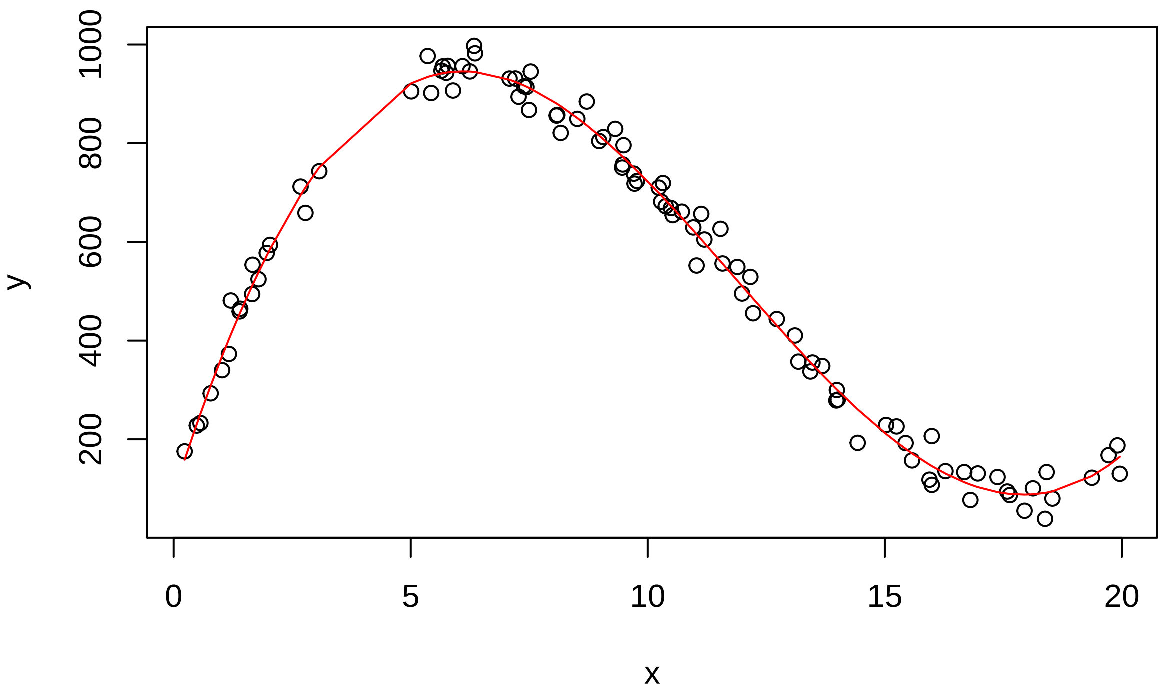
Root MSE	49.91697	R-Square	0.9740
Dependent Mean	536.37060	Adj R-Sq	0.9729
Coeff Var	9.30643		

Univariate Regression Table Based on the Usual Degrees of Freedom							
Variable	DF	Coefficient	Type II Sum of Squares	Mean Square	F Value	Pr > F	Label
Intercept	1	269.66383	664191	664191	266.56	<.0001	Intercept
Pspline.x_1	1	121.03622	2383539	2383539	956.59	<.0001	x 1
Pspline.x_2	1	-190.96987	1475069	1475069	591.99	<.0001	x 2
Pspline.x_3	1	-54.31165	88078	88078	35.35	<.0001	x 3
Pspline.x_4	1	92.96705	261424	261424	104.92	<.0001	x 4

2.3 Piecewise Cubic Splines

match 1st and 2nd derivatives at knots

```
mod7=lm(y~bSpline(x,knots=quantile(x,c(0.25,0.5,0.75)),degree=3))
plot(x,y)
lines(x,fitted(mod7),col="red")
```



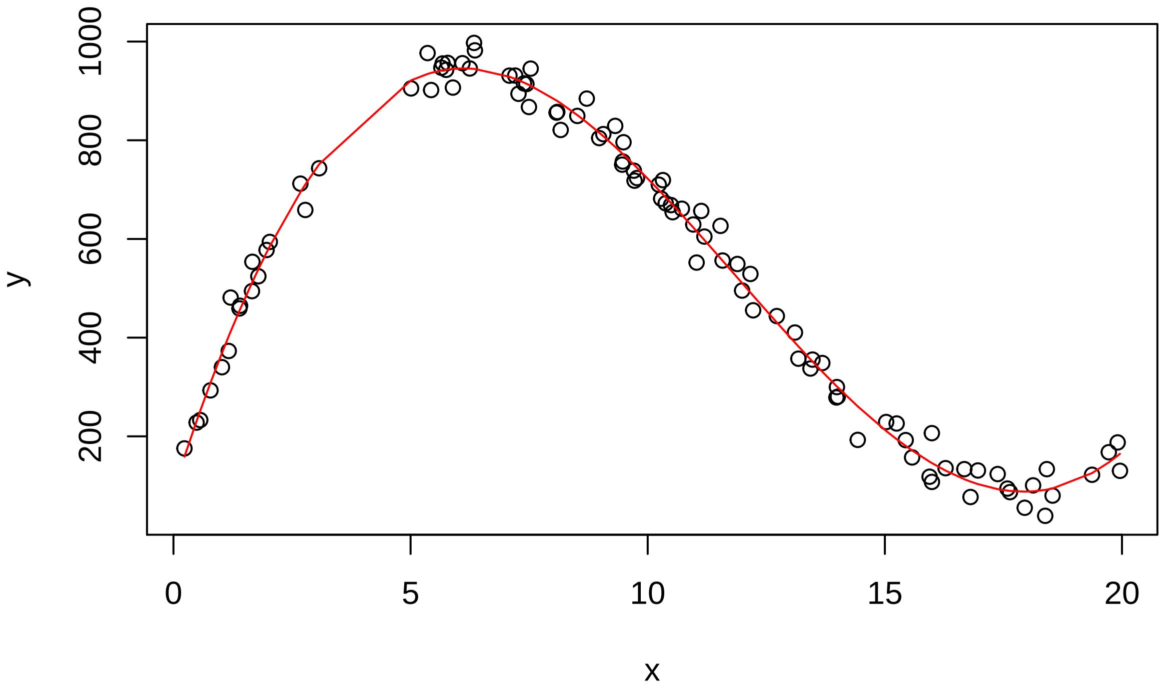

```
temp=summary(mod7)
rownames(temp$coefficients)=c("intercept","Spline1",'Spline2','Spline3','Spline4','Spline5','Spline6')
temp
```

```
##
## Call:
## lm(formula = y ~ bSpline(x, knots = quantile(x, c(0.25, 0.5,
##     0.75)), degree = 3))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -67.870 -18.718  -0.011  17.513  69.002
##
## Coefficients:
##           Estimate Std. Error t value Pr(>|t|)
## intercept  158.678     17.082   9.289 6.52e-15 ***
## Spline1    559.668     38.592  14.502  < 2e-16 ***
## Spline2    932.721     20.517  45.461  < 2e-16 ***
## Spline3    595.629     23.527  25.317  < 2e-16 ***
## Spline4     22.769     25.048   0.909    0.366
## Spline5   -151.407     28.007  -5.406 4.96e-07 ***
## Spline6      5.939     23.437   0.253    0.801
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 29.33 on 93 degrees of freedom
## Multiple R-squared:  0.9912, Adjusted R-squared:  0.9906
## F-statistic:  1746 on 6 and 93 DF,  p-value: < 2.2e-16
```

Similarly, it is equivalent to fit a linear model with an intercept, x, x^2, x^3, $(x - Q_{0.25})^3_+$, $(x - Q_{0.5})^3_+$, $(x - Q_{0.75})^3_+$.

```r
x1_3=x1^3
x2_3=x2^3
x3_3=x3^3
x_sq=x^2
x_cb=x^3
mod8=lm(y~x+x_sq+x_cb+x1_3+x2_3+x3_3)
plot(x,y)
lines(x,fitted(mod8),col="red")
```




```r
summary(mod8)
```

```
##
## Call:
## lm(formula = y ~ x + x_sq + x_cb + x1_3 + x2_3 + x3_3)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -67.870 -18.718  -0.011  17.513  69.002
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   90.13359   21.50000   4.192 6.29e-05 ***
## x            302.89791   22.39548  13.525  < 2e-16 ***
## x_sq         -30.43245    5.57569  -5.458 3.97e-07 ***
```

```
## x_cb          0.61739    0.39667   1.556    0.123
## x1_3          0.47225    0.63347   0.745    0.458
## x2_3          0.03212    0.51415   0.062    0.950
## x3_3         -0.28691    0.58260  -0.492    0.624
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 29.33 on 93 degrees of freedom
## Multiple R-squared:  0.9912, Adjusted R-squared:  0.9906
## F-statistic:  1746 on 6 and 93 DF,  p-value: < 2.2e-16
```

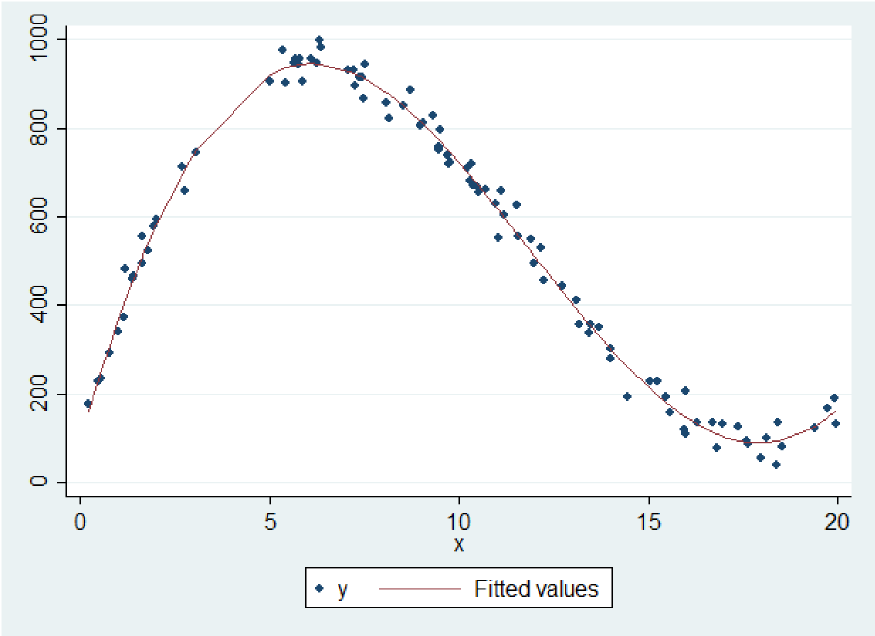
In Stata,

```
gen x_sq=x^2
gen x_cub=x^3
gen x1=cond(x > 6, (x - 6)^3, 0)
gen x2=cond(x > 10, (x - 10)^3, 0)
gen x3=cond(x > 14, (x - 14)^3, 0)
regress y x x_sq x_cub x1 x2 x3
predict fitted_spline
twoway scatter y x || line fitted_spline x
```

```
. regress y x x_sq x_cub x1 x2 x3

      Source |       SS           df       MS      Number of obs   =       100
-------------+----------------------------------   F(6, 93)        =   1781.32
       Model |  9012910.13         6  1502151.69   Prob > F        =    0.0000
    Residual |  78425.1391        93  843.281066   R-squared       =    0.9914
-------------+----------------------------------   Adj R-squared   =    0.9908
       Total |  9091335.27        99  91831.6694   Root MSE        =    29.039

------------------------------------------------------------------------------
           y |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
           x |   305.3281   15.22431    20.06   0.000     275.0956    335.5606
        x_sq |  -32.41541   2.300889   -14.09   0.000    -36.98452    -27.8463
       x_cub |   .8335522   .1436818     5.80   0.000     .5482287    1.118876
          x1 |   .0872974   .1181901     0.74   0.462    -.1474048    .3219995
          x2 |   .0017223   .0491719     0.04   0.972    -.0959233    .0993679
          x3 |  -.0347964   .0481239    -0.72   0.471    -.1303608     .060768
       _cons |   90.26577   18.35739     4.92   0.000     53.81163    126.7199
------------------------------------------------------------------------------
```



In SAS,

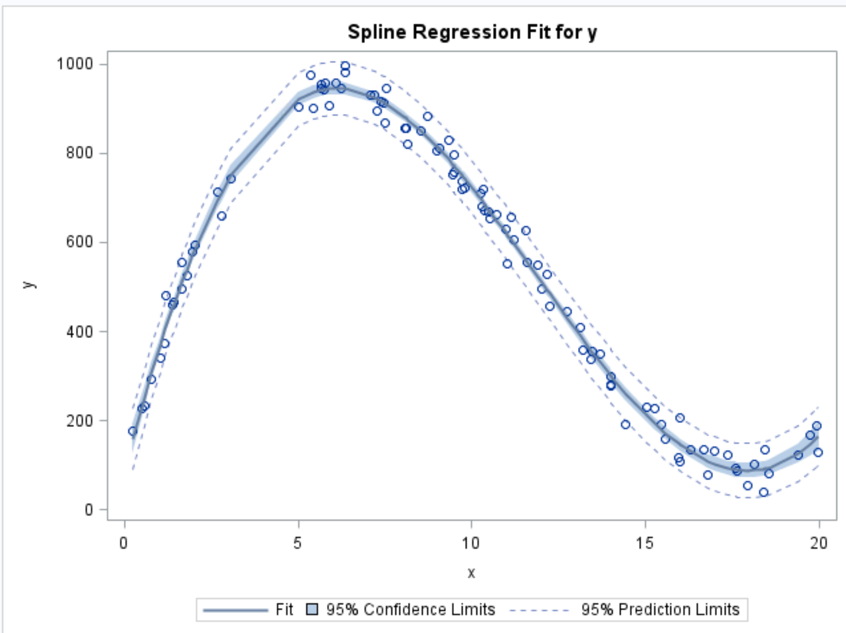
```
proc transreg data=dat ss2 short;
   model identity(y) = spline(x / knots=6 10 14);
run;
```

The TRANSREG Procedure Hypothesis Tests for Identity(y)

Univariate ANOVA Table Based on the Usual Degrees of Freedom					
Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	6	9011337	1501890	1745.99	<.0001
Error	93	79998	860		
Corrected Total	99	9091335			

Root MSE	29.32907	R-Square	0.9912
Dependent Mean	536.37060	Adj R-Sq	0.9906
Coeff Var	5.46806		

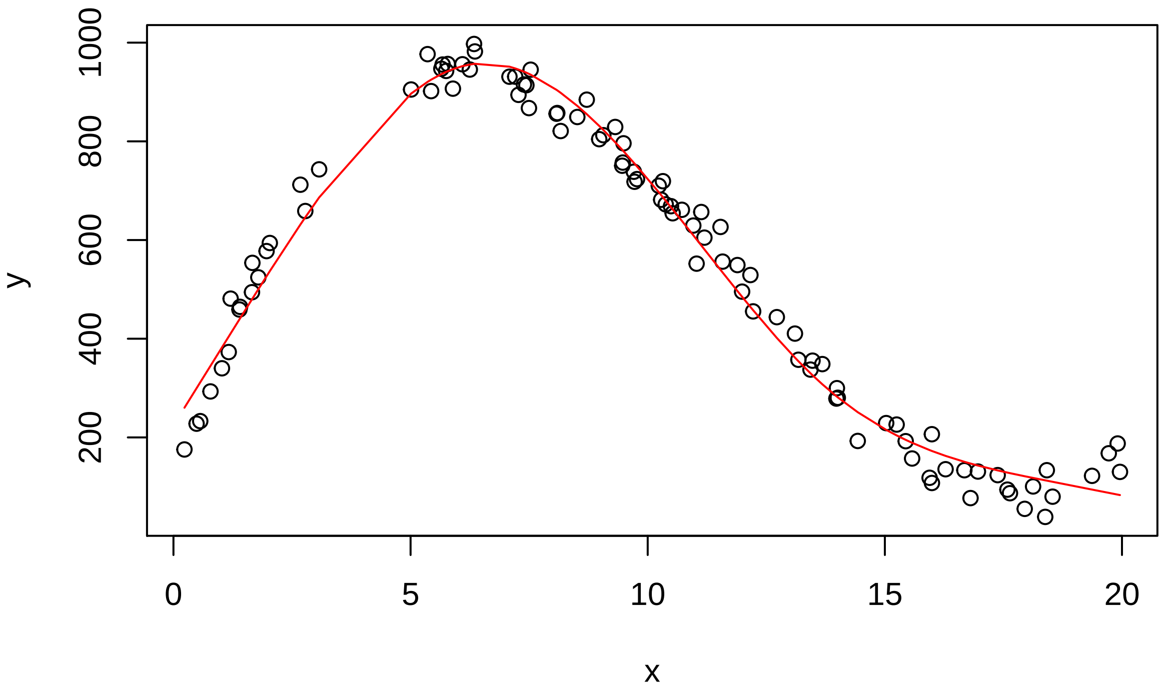
Univariate Regression Table Based on the Usual Degrees of Freedom						
Variable	DF	Coefficient	Type II Sum of Squares	Mean Square	F Value	Pr > F
Intercept	1	1085.94235	2.71E7	2.71E7	31503.8	<.0001
Spline(x)	6	-55.12164	9011337	1501890	1745.99	<.0001



2.4 Restricted/Natural Cubic Splines

Linear on edges!

```
library(Hmisc)
mod9=lm(y~rcspline.eval(x,nk=5,inclx=T))
plot(x,y)
lines(x,fitted(mod9),col="red")
```



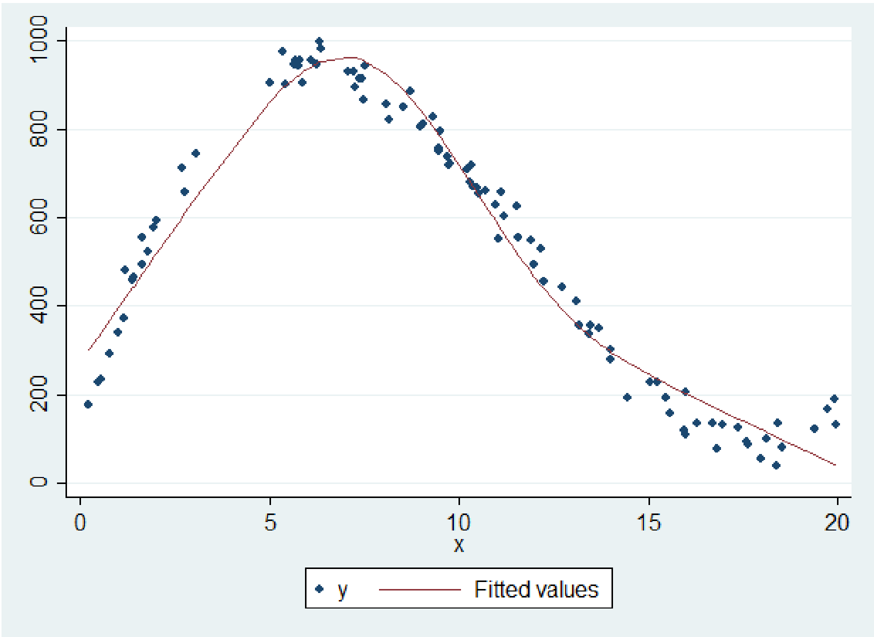

```
temp=summary(mod9)
rownames(temp$coefficients)=c("intercept","Spline1",'Spline2','Spline3','Spline4')
temp
```

```
##
## Call:
## lm(formula = y ~ rcspline.eval(x, nk = 5, inclx = T))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -84.599 -27.803   2.039  24.330 103.586
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## intercept    224.404     15.401  14.571   <2e-16 ***
## Spline1      154.378      4.852  31.819   <2e-16 ***
## Spline2     -527.398     24.600 -21.439   <2e-16 ***
## Spline3     1088.711    105.172  10.352   <2e-16 ***
## Spline4       15.315    161.574   0.095    0.925
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 42.77 on 95 degrees of freedom
## Multiple R-squared:  0.9809, Adjusted R-squared:  0.9801
```

```
## F-statistic:  1219 on 4 and 95 DF,  p-value: < 2.2e-16
```

In Stata,

```
keep x y
mkspline x_spline = x, cubic nknots(5) displayknots
regress y x_spline*
OR
mkspline x_spline = x, cubic knots(1 6 10 14 19)
quietly regress y x_spline*
predict fitted_spline
twoway scatter y x || line fitted_spline x
```

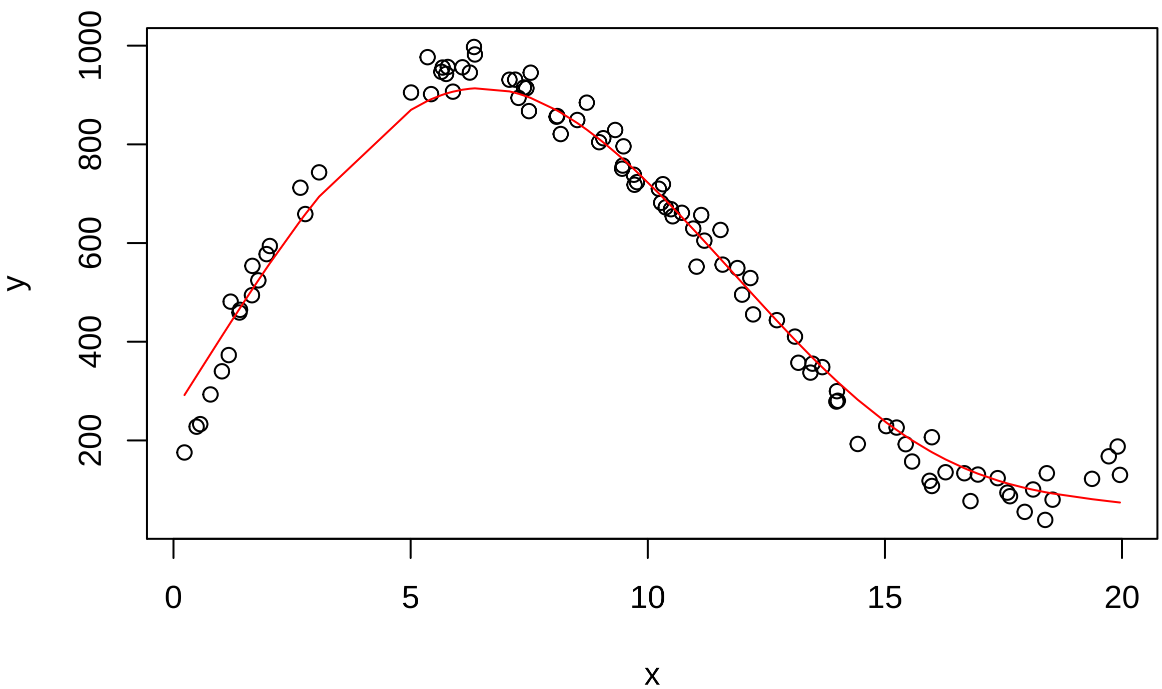


In SAS,
There is no function to do it **directly**.

3.Generalized Additive Model (GAM)

A generalized additive model (GAM) is a generalized linear model in which the linear predictor depends linearly on unknown smooth functions of some predictor variables. It is like a cubic spline but with every x value as a knot and a penalty on the second derivative (curvature). (We don't want a discontinuous or a zigzag curve)
GAM is very flexible. If now there are three covariates x_1, x_2, x_3 and an outcome y, and you believe that y and x_3 have a linear relationship, you can specify a GAM: $E(Y) = f_1(x_1) + f_2(x_2) + \beta_3 x_3$. If you are not sure at all, you can just use a very general model: $E(Y) = f_1(x_1) + f_2(x_2) + f_3(x_3)$, where f_1, f_2, f_3 are some nonparametric functions (You cannot write out the explicit form of those functions), and GAM can find the 'best' fit for you. In GAM, we still assume a normal distribution for the error terms (which is parametric), and that is why we say GAM is a **semiparametric** method.

```r
library(gam)
mod10=gam(y~s(x,4))
plot(x,y)
lines(x,fitted(mod10),col='red')
```




```r
summary(mod10)
```

```
##
## Call: gam(formula = y ~ s(x, 4))
## Deviance Residuals:
##       Min        1Q    Median        3Q       Max
## -116.2892  -21.6298   -0.0602   30.9159  112.9267
##
## (Dispersion Parameter for gaussian family taken to be 2039.414)
##
##     Null Deviance: 9091335 on 99 degrees of freedom
## Residual Deviance: 193744.7 on 95.0002 degrees of freedom
## AIC: 1052.7
```

```
##
## Number of Local Scoring Iterations: 2
##
## Anova for Parametric Effects
##            Df  Sum Sq Mean Sq F value    Pr(>F)
## s(x, 4)     1 3055335 3055335  1498.1 < 2.2e-16 ***
## Residuals 95  193745    2039
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##             Npar Df Npar F     Pr(F)
## (Intercept)
## s(x, 4)           3 954.95 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

In Stata, there is no GAM function.
In SAS,

```
proc gam data=dat;
   model y = spline(x);
run;
```

Regression Model Analysis
Parameter Estimates

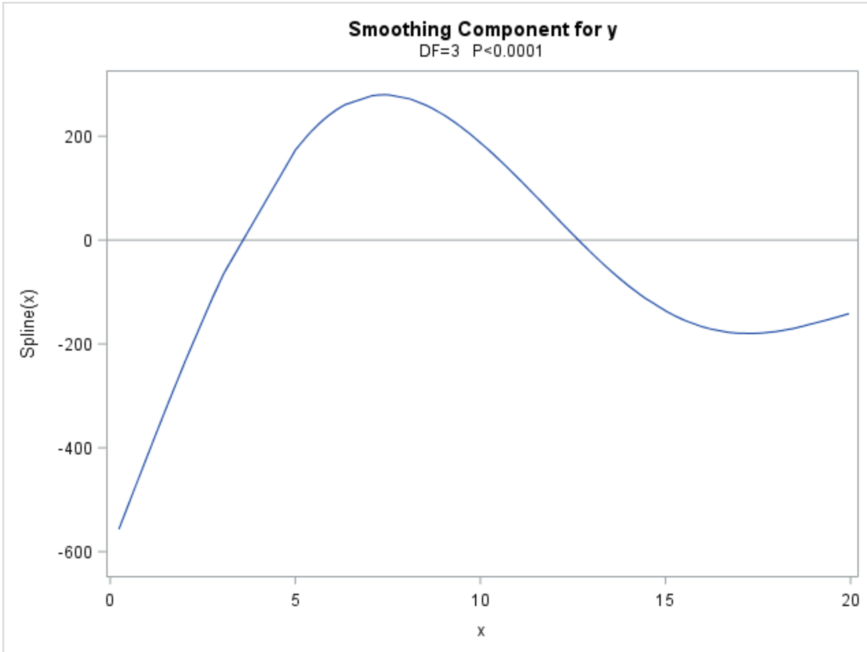
Parameter	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	856.37727	9.42097	90.90	<.0001
Linear(x)	-32.09643	0.82927	-38.70	<.0001

Smoothing Model Analysis
Fit Summary for Smoothing Components

Component	Smoothing Parameter	DF	GCV	Num Unique Obs
Spline(x)	0.999913	3.000000	2146.903500	100

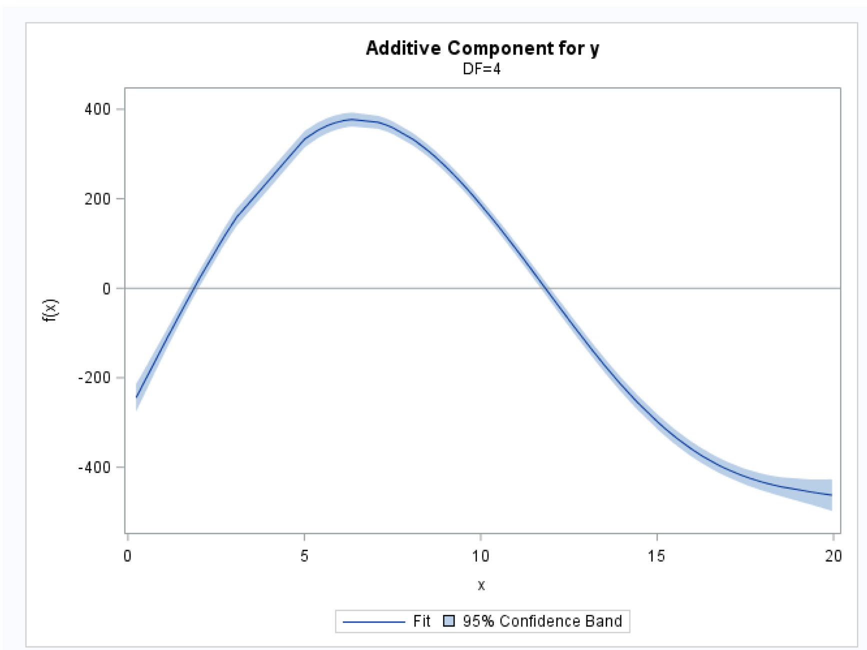
Smoothing Model Analysis
Analysis of Deviance

Source	DF	Sum of Squares	Chi-Square	Pr > ChiSq
Spline(x)	3.00000	5842245	2864.4656	<.0001



The figure above shows the “departure” from linear model. If we combine both the linear effect and nonlinear effect, we should get the same plot as in R. Let’s try!

```
proc gam data=dat plots=COMPONENTS(additive clm);
   model y = spline(x);
run;
```



4.Assess nonlinearity with Splines/GAM

Given the dataset "lab4-1.csv", we are interested in whether x and y have a linear relationship. How can we do a formal statistical test?
Model 9: $E[Y] = \beta_0 + \beta_1 x$
Model 10: $E[Y] = \beta_0 + \beta_1 x + Spline(x)/GAM(x)$
We use F test to compare the variance explained by the linear term and the variance explained by (linear term + spline term). If the former is not far smaller than the latter, then we say, a single linear term is enough! Remember, F test can only compare nested models!

```
mod11=lm(y~x)
mod12=lm(y~x+bSpline(x, df=6, degree = 3))
## or use GAM mod10=gam(y~x+s(x,4))
anova(mod11,mod12)
```

```
## Analysis of Variance Table
##
## Model 1: y ~ x
## Model 2: y ~ x + bSpline(x, df = 6, degree = 3)
##   Res.Df     RSS Df Sum of Sq      F    Pr(>F)
## 1     98 6036003
## 2     93   80004  5   5955999 1384.7 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Q: How to interpret this result?

We showed in section 2, a cubic spline with 3 knots is equivalent to doing a regression on x, x^2, x^3, $(x-a)^3_+$, $(x-b)^3_+$, $(x-c)^3_+$. Hence, we can rewrite Model 10 as:
$E[Y] = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x-a)^3_+ + \beta_5 (x-b)^3_+ + \beta_6 (x-c)^3_+$.
Then testing whether x and y are linear is equivalent to jointly testing:
$H_0 : \beta_2 = \beta_3 = \beta_4 = \beta_5 = \beta_6 = 0$
H_A: At least one is not equal to zero

Q: Why the F statistic has df(5,93)?

In stata,

```
keep x y
gen x_sq=x^2
gen x_cub=x^3
gen x1=cond(x > 6, (x - 6)^3, 0)
gen x2=cond(x > 10, (x - 10)^3, 0)
gen x3=cond(x > 14, (x - 14)^3, 0)
regress y x x_sq x_cub x1 x2 x3
test (x_sq=0)(x_cub=0)(x1=0)(x2=0)(x3=0)
```

```
. regress y x x_sq x_cub x1 x2 x3

      Source |       SS           df       MS      Number of obs   =       100
-------------+----------------------------------   F(6, 93)        =   1781.32
       Model |  9012910.13         6  1502151.69   Prob > F        =    0.0000
    Residual |  78425.1391        93  843.281066   R-squared       =    0.9914
-------------+----------------------------------   Adj R-squared   =    0.9908
       Total |  9091335.27        99  91831.6694   Root MSE        =    29.039

------------------------------------------------------------------------------
           y |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
           x |   305.3281   15.22431    20.06   0.000     275.0956    335.5606
        x_sq |  -32.41541   2.300889   -14.09   0.000    -36.98452    -27.8463
       x_cub |   .8335522   .1436818     5.80   0.000     .5482287    1.118876
          x1 |   .0872974   .1181901     0.74   0.462    -.1474048    .3219995
          x2 |   .0017223   .0491719     0.04   0.972    -.0959233    .0993679
          x3 |  -.0347964   .0481239    -0.72   0.471    -.1303608     .060768
       _cons |   90.26577   18.35739     4.92   0.000     53.81163    126.7199
------------------------------------------------------------------------------

. test (x_sq=0)(x_cub=0)(x1=0)(x2=0)(x3=0)

 ( 1)  x_sq = 0
 ( 2)  x_cub = 0
 ( 3)  x1 = 0
 ( 4)  x2 = 0
 ( 5)  x3 = 0

       F(  5,    93) = 1412.95
            Prob > F =    0.0000
```

The difference in F statistic is due to different quantile calculation methods in R and Stata.

Or you can use a restricted cubic spline to test it in Stata (It is easier in the sense that there is a built-in function and the number of parameters is smaller given the same number of knots).

```
keep x y
mkspline x_spline = x, cubic knots(1 6 10 14 19)
regress y x x_spline*
test (x_spline2=0)(x_spline3=0)(x_spline4=0)
```

y	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
x	160.2311	4.798569	33.39	0.000	150.7047	169.7574
x_spline1	0	(omitted)				
x_spline2	-586.2153	26.03742	-22.51	0.000	-637.9062	-534.5245
x_spline3	1135.601	96.7791	11.73	0.000	943.4703	1327.732
x_spline4	13.62831	139.644	0.10	0.922	-263.6	290.8566
_cons	216.2682	14.68676	14.73	0.000	187.1113	245.4251

```
.
. test (x_spline2=0)(x_spline3=0)(x_spline4=0)

 ( 1)  x_spline2 = 0
 ( 2)  x_spline3 = 0
 ( 3)  x_spline4 = 0

       F(  3,    95) = 1211.67
            Prob > F =    0.0000
```

Q: How to test if $x + x^2$ is sufficient?

Q: Are more complex models always better?

Q: How to write code to deal with more than one covariate?
For example, there are three covariates, and you are only interested in whether x_3 and y are linear. Given that x_1, x_2 are potential confounders, you may want to add them to your model.