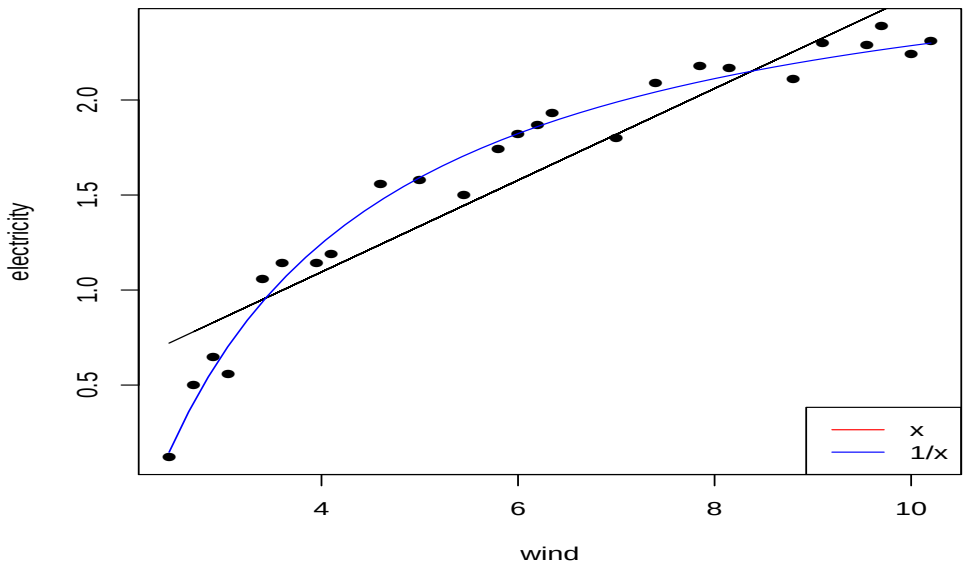
Transformation on the regressors

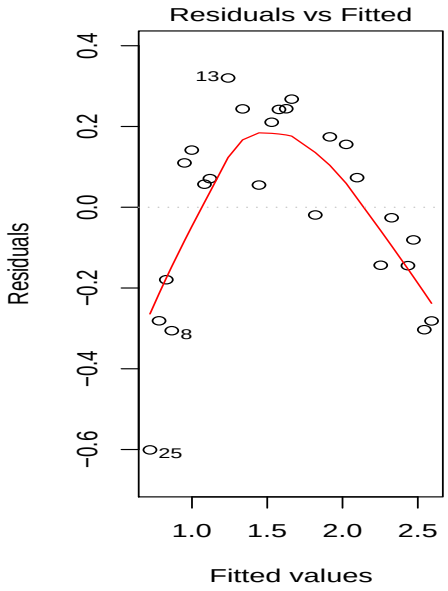
box-tidwell transformation

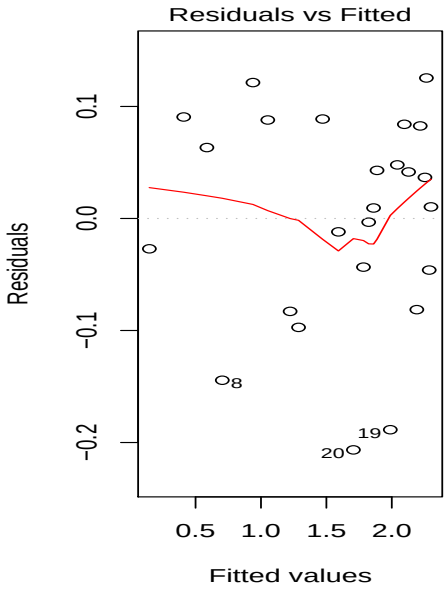
```
> windmill=read.table("ex-5-2.csv", header=T, sep=",")
> str(windmill)

'data.frame':        25 obs. of  3 variables:
 $ Observation.Number..i  : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Wind.Velocity..xi..mph.: num  5 6 3.4 2.7 10 9.7 9.55 3.05 8.15 6.2 ...
 $ DC.Output..yi          : num  1.58 1.82 1.06 0.5 2.24 2.39 2.29 0.56 2.17 1.87 ...

> x=windmill[,2]
> y=windmill[,3]
> f=lm(y~x)
> summary(f)

Call:
lm(formula = y ~ x)

Residuals:
     Min       1Q   Median       3Q      Max
-0.60084 -0.14463  0.05709  0.17433  0.32019

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.12946    0.12619   1.026    0.316
x            0.24138    0.01908  12.651 7.64e-12

Residual standard error: 0.2364 on 23 degrees of freedom
Multiple R-squared: 0.8744,        Adjusted R-squared: 0.8689
F-statistic: 160.1 on 1 and 23 DF,  p-value: 7.641e-12

> f.inv=lm(y~I(1/x))
> summary(f)

Call:
lm(formula = y ~ x)

Residuals:
     Min       1Q   Median       3Q      Max
-0.60084 -0.14463  0.05709  0.17433  0.32019

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.12946    0.12619   1.026    0.316
x            0.24138    0.01908  12.651 7.64e-12

Residual standard error: 0.2364 on 23 degrees of freedom
Multiple R-squared: 0.8744,        Adjusted R-squared: 0.8689
F-statistic: 160.1 on 1 and 23 DF,  p-value: 7.641e-12
```

```
> plot(x,y,xlab="wind",ylab="electricity", pch=16)
> lines(x, fitted(f))
> x.new=seq(min(x), max(x), length=40)
> yfit=predict(f.inv, data.frame(x=x.new))
> lines(x.new, yfit, col=4)
> legend("bottomright", legend=c("x","1/x"), lty=1, col=c(2,4))
```




```
> c(summary(f)$adj.r, summary(f.inv)$adj.r)

[1] 0.8688935 0.9789429

> c(summary(f)$sigma, summary(f.inv)$sigma)

[1] 0.23642480 0.09475018

> par(mfrow=c(1,2))
> plot(f,1)
> plot(f.inv,1)
```





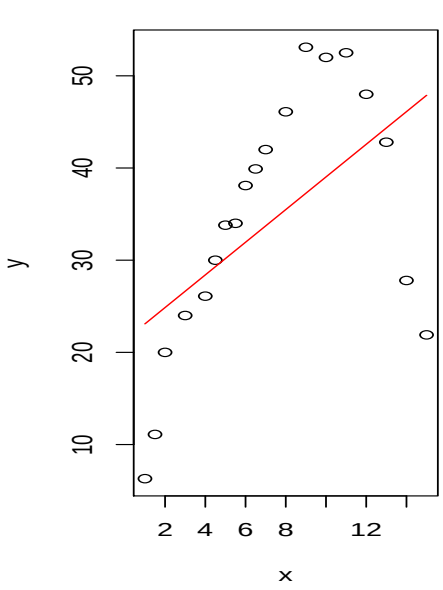

```
> library(car)
> box.tidwell(y, x)

                        1
Initial Power   -0.91800
Score Statistic -9.09851
p-value          0.00000
MLE of Power    -0.83301

iterations =  3
```

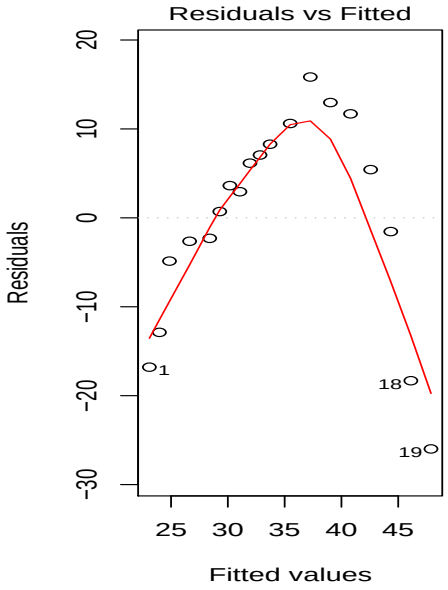
Polynomial regression

```
> par(mfrow=c(1,2))
> hardwood=read.table("ex-7-1.txt", header=T)
> x=hardwood[,1]
> y=hardwood[,2]
> n=length(x)  #no of obs
> plot(y~x)
> g1=lm(y~x)
> summary(g1)

Call:
lm(formula = y ~ x)

Residuals:
    Min      1Q  Median      3Q     Max
-25.986  -3.749   2.938   7.675  15.840
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  21.3213     5.4302   3.926  0.00109
x             1.7710     0.6478   2.734  0.01414

Residual standard error: 11.82 on 17 degrees of freedom
Multiple R-squared: 0.3054,        Adjusted R-squared: 0.2645
F-statistic: 7.474 on 1 and 17 DF,  p-value: 0.01414
```

```
> lines(x, fitted(g1), col=2)
> plot(g1, which=1)
```



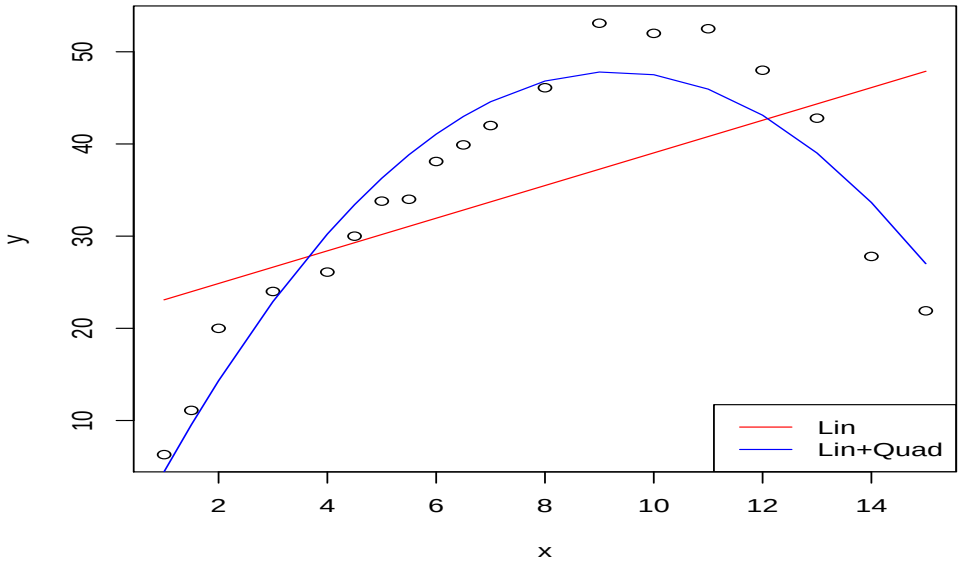



```
> summary(g1)$adj.r

[1] 0.2645135

> summary(g1)$sigma

[1] 11.81589

> summary(g1)$r.squared

[1] 0.3053739

> summary(g1)$f.statistic

NULL

> g2=lm(y~x+I(x^2)) #lines(x, fitted(g2))
> print(g2)
```

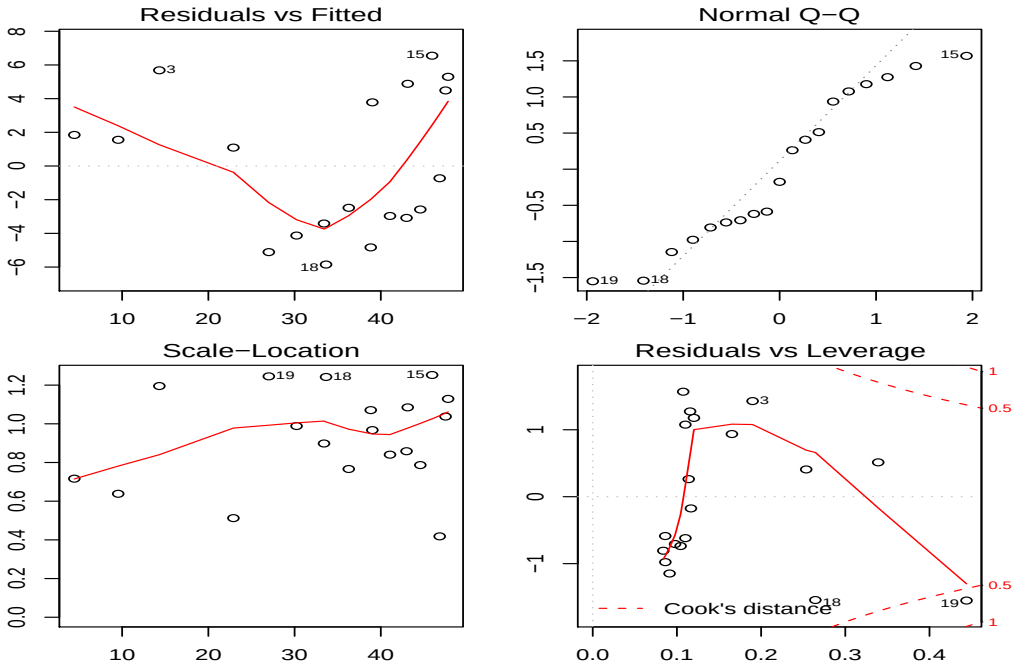
```
Call:
lm(formula = y ~ x + I(x^2))

Coefficients:
(Intercept)            x       I(x^2)
    -6.6742      11.7640      -0.6345
```

```
> summary(g2)

Call:
lm(formula = y ~ x + I(x^2))

Residuals:
    Min      1Q  Median      3Q     Max
-5.8503 -3.2482 -0.7267  4.1350  6.5506

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -6.67419    3.39971  -1.963   0.0673
x           11.76401    1.00278  11.731 2.85e-09
I(x^2)      -0.63455    0.06179 -10.270 1.89e-08

Residual standard error: 4.42 on 16 degrees of freedom
Multiple R-squared: 0.9085,        Adjusted R-squared: 0.8971
F-statistic: 79.43 on 2 and 16 DF,  p-value: 4.912e-09
```

```
> plot(x,y)
> lines(x, fitted(g1), col=2)
> lines(x, fitted(g2), col=4)
> legend("bottomright", legend=c("Lin","Lin+Quad"), lty=1, col=c(2,4))
> summary(g1)$adj.r

[1] 0.2645135

> summary(g2)$adj.r

[1] 0.8970647
```




```
> #centered x and how to get all the information we need from summary(g)
>
> center.x=scale(x, center=T, scale=F) #center.x=x-mean(x)
> g=lm(y~center.x+I(center.x^2))
> summary(g)

Call:
lm(formula = y ~ center.x + I(center.x^2))

Residuals:
    Min      1Q  Median      3Q     Max
-5.8503 -3.2482 -0.7267  4.1350  6.5506

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   45.29497    1.48287   30.55 1.29e-15
center.x       2.54634    0.25384   10.03 2.63e-08
I(center.x^2) -0.63455    0.06179  -10.27 1.89e-08

Residual standard error: 4.42 on 16 degrees of freedom
Multiple R-squared: 0.9085,        Adjusted R-squared: 0.8971
F-statistic: 79.43 on 2 and 16 DF,  p-value: 4.912e-09

> coef(g) #regression coefficients

  (Intercept)      center.x I(center.x^2)
   45.2949731     2.5463440    -0.6345492

> par(mfrow=c(2,2), mar=c(2,2,2,2))
> plot(g, ask=F)
```



orthogonal polynomial regression: orthogonal polynomials

```
> g3=lm(y~poly(x,2))
> summary(g3)

Call:
lm(formula = y ~ poly(x, 2))

Residuals:
    Min      1Q  Median      3Q     Max
-5.8503 -3.2482 -0.7267  4.1350  6.5506

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   34.184      1.014  33.709 2.73e-16
poly(x, 2)1   32.302      4.420   7.308 1.76e-06
poly(x, 2)2  -45.396      4.420 -10.270 1.89e-08

Residual standard error: 4.42 on 16 degrees of freedom
Multiple R-squared: 0.9085,        Adjusted R-squared: 0.8971
F-statistic: 79.43 on 2 and 16 DF,  p-value: 4.912e-09

> anova(g3)

Analysis of Variance Table

Response: y
           Df  Sum Sq Mean Sq F value    Pr(>F)
poly(x, 2)  2 3104.25 1552.12  79.434 4.912e-09
Residuals  16  312.64   19.54
```

```
> (X=model.matrix(g3))

   (Intercept) poly(x, 2)1 poly(x, 2)2
1            1 -0.34338179  0.41054014
2            1 -0.31596896  0.31794828
3            1 -0.28855613  0.23234542
4            1 -0.23373047  0.08210670
5            1 -0.17890480 -0.04017599
6            1 -0.15149197 -0.09083384
7            1 -0.12407914 -0.13450267
8            1 -0.09666630 -0.17118251
9            1 -0.06925347 -0.20087334
10           1 -0.04184064 -0.22357516
11           1 -0.01442781 -0.23928798
12           1  0.04039786 -0.24974661
13           1  0.09522352 -0.23224922
14           1  0.15004919 -0.18679582
15           1  0.20487485 -0.11338639
16           1  0.25970052 -0.01202095
17           1  0.31452618  0.11730050
18           1  0.36935185  0.27457797
19           1  0.42417751  0.45981146
attr(,"assign")
[1] 0 1 1

> t(X) %*% X

             (Intercept)   poly(x, 2)1   poly(x, 2)2
(Intercept)  1.900000e+01  0.000000e+00 -3.885781e-16
poly(x, 2)1  0.000000e+00  1.000000e+00 -1.215662e-17
poly(x, 2)2 -3.885781e-16 -1.215662e-17  1.000000e+00
```

Piecewise regression

when a polynomial degree 3 is not sufficiently flexible to fit the data adequately.

broken stick regression

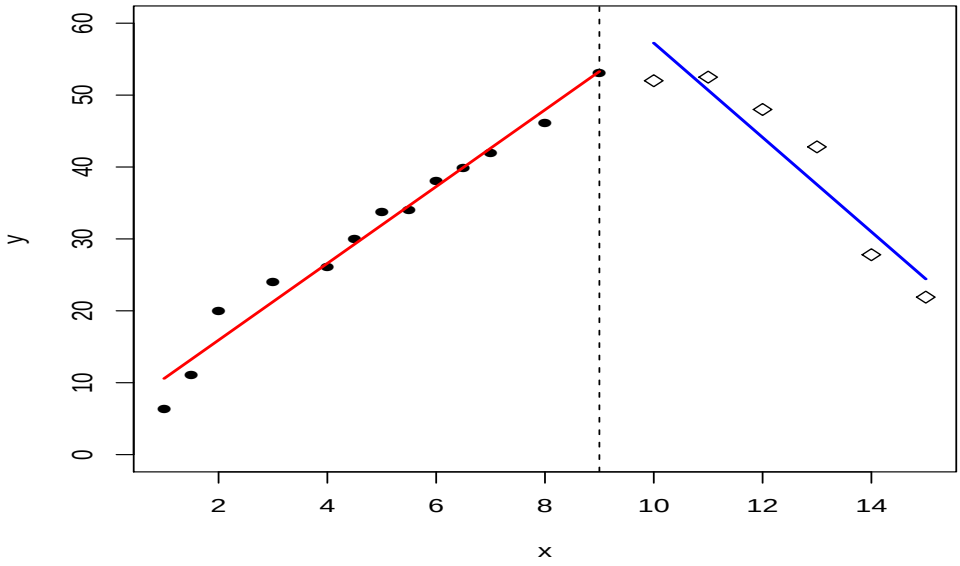
1. lm using subset

```
> t0=9
> gb1=lm(y~x, subset=(x<=t0))
> (b1=coef(gb1))

(Intercept)           x
   5.245108    5.338311

> gb2=lm(y~x, subset=(x>t0))
> (b2=coef(gb2))
```

```
(Intercept)           x
 122.904762   -6.565714
```

2. define two subsets.

```
> plot(x,y,type="n", ylim=c(0,60))
> abline(v=t0, lty=2)
> x.s=x[x<=t0];y.s=y[x<=t0]
> fb1=lm(y.s~x.s)
> print(fb1)

Call:
lm(formula = y.s ~ x.s)

Coefficients:
(Intercept)          x.s
      5.245        5.338

> points(x.s, y.s, pch=16)
> library(car)
> reg.line(fb1)
> x.n=x[x>t0];y.n=y[x>t0]
> fb2=lm(y.n~x.n)
> print(fb2)

Call:
lm(formula = y.n ~ x.n)

Coefficients:
(Intercept)          x.n
    122.905       -6.566

> points(x.n,y.n, pch=5)
> reg.line(fb2, col=4)
```

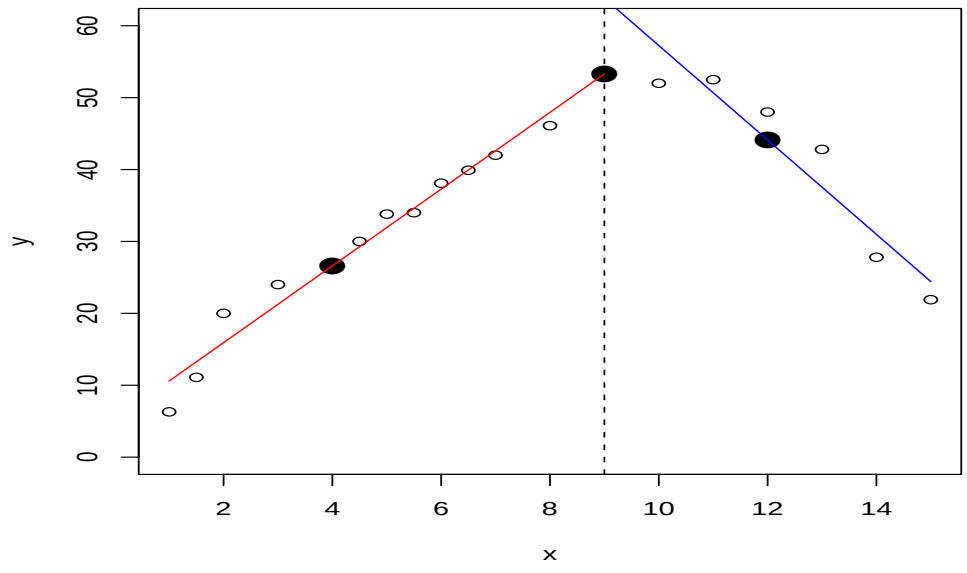


response surface model: second-order polynomial regression
(polynomial regression in several variables)
g=lm(y polym(x1, x2, degree=2))
3. piecewise regression in one regression model

```
> br.zero=function(x) ifelse(x<=t0, 0, 1)
> br.one=function(x) ifelse(x>t0, x-t0, 0)
> gb=lm(y~x+br.zero(x)+br.one(x))
> summary(gb)

Call:
lm(formula = y ~ x + br.zero(x) + br.one(x))

Residuals:
    Min      1Q  Median      3Q     Max
-5.2476 -2.0021 -0.1899  1.8407  5.2495

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   5.2451     1.9533   2.685  0.01695
x             5.3383     0.3613  14.776  2.4e-10
br.zero(x)   10.5234     3.3840   3.110  0.00717
br.one(x)   -11.9040     0.8292 -14.355  3.6e-10

Residual standard error: 3.122 on 15 degrees of freedom
Multiple R-squared: 0.9572,        Adjusted R-squared: 0.9486
F-statistic: 111.8 on 3 and 15 DF,  p-value: 1.731e-10

> model.matrix(gb)
```

```
   (Intercept)    x br.zero(x) br.one(x)
1            1  1.0          0         0
2            1  1.5          0         0
3            1  2.0          0         0
4            1  3.0          0         0
5            1  4.0          0         0
6            1  4.5          0         0
7            1  5.0          0         0
8            1  5.5          0         0
9            1  6.0          0         0
10           1  6.5          0         0
11           1  7.0          0         0
12           1  8.0          0         0
13           1  9.0          0         0
14           1 10.0          1         1
15           1 11.0          1         2
16           1 12.0          1         3
17           1 13.0          1         4
18           1 14.0          1         5
19           1 15.0          1         6
attr(,"assign")
[1] 0 1 2 3

> (b=coef(gb))

(Intercept)           x  br.zero(x)   br.one(x)
   5.245108    5.338311   10.523426  -11.904025

> b[1]

(Intercept)
   5.245108

> b[2]

       x
5.338311

> b[1]+b[3]-b[4]*t0

(Intercept)
   122.9048

> b[2]+b[4]

        x
-6.565714
```

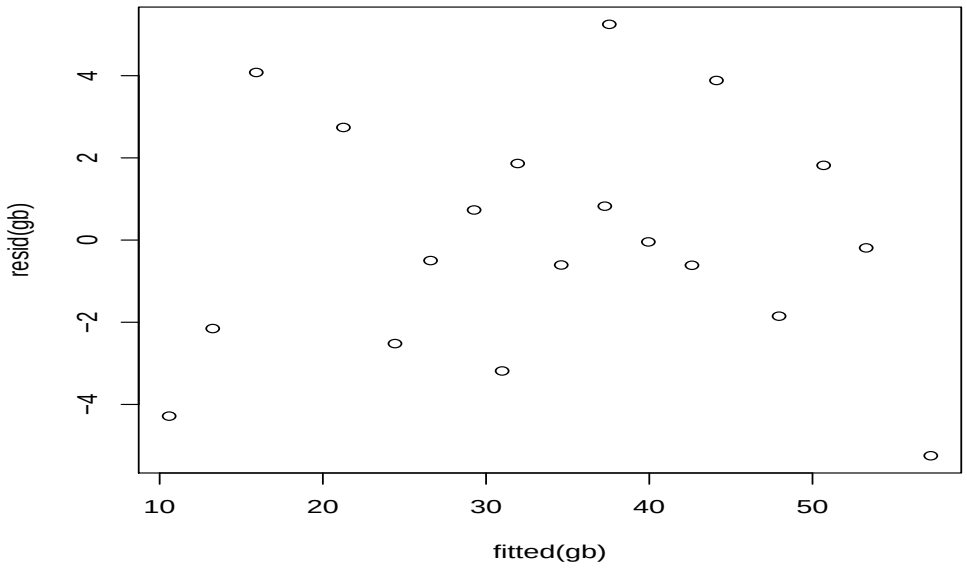
```
> plot(x,y,type="n", ylim=c(0,60))
> abline(v=t0, lty=2)
> points(x, y)
> x.new=c(4,9, 12)
> yfit=b[1]+b[2]*x.new+b[3]*br.zero(x.new)+b[4]*br.one(x.new)
> points(x.new, yfit, pch=16, cex=2)
> aa=min(x)
> bb=max(x)
> xx=c(aa, t0, t0+.05, bb)
> yy=c(b[1]+b[2]*xx[1]+b[3]*br.zero(xx[1])+b[4]*br.one(xx[1]),
+ b[1]+b[2]*xx[2]+b[3]*br.zero(xx[2])+b[4]*br.one(xx[2]),
+ b[1]+b[2]*xx[3]+b[3]*br.zero(xx[3])+b[4]*br.one(xx[3]),
+ b[1]+b[2]*xx[4]+b[3]*br.zero(xx[4])+b[4]*br.one(xx[4]))
> xx

[1]  1.00  9.00  9.05 15.00

> yy

(Intercept) (Intercept) (Intercept) (Intercept)
   10.58342    53.28991    63.48505    24.41905

> segments(xx[1],yy[1], xx[2], yy[2], col=2)
> segments(xx[3],yy[3], xx[4], yy[4], col=4)
```




```
> summary(gb)

Call:
lm(formula = y ~ x + br.zero(x) + br.one(x))
```

```
Residuals:
    Min      1Q  Median      3Q     Max
-5.2476 -2.0021 -0.1899  1.8407  5.2495

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   5.2451     1.9533   2.685  0.01695
x             5.3383     0.3613  14.776  2.4e-10
br.zero(x)   10.5234     3.3840   3.110  0.00717
br.one(x)   -11.9040     0.8292 -14.355  3.6e-10

Residual standard error: 3.122 on 15 degrees of freedom
Multiple R-squared: 0.9572,        Adjusted R-squared: 0.9486
F-statistic: 111.8 on 3 and 15 DF,  p-value: 1.731e-10
```

```
> par(mfrow=c(1,1))
> plot(resid(gb)~fitted(gb))
```

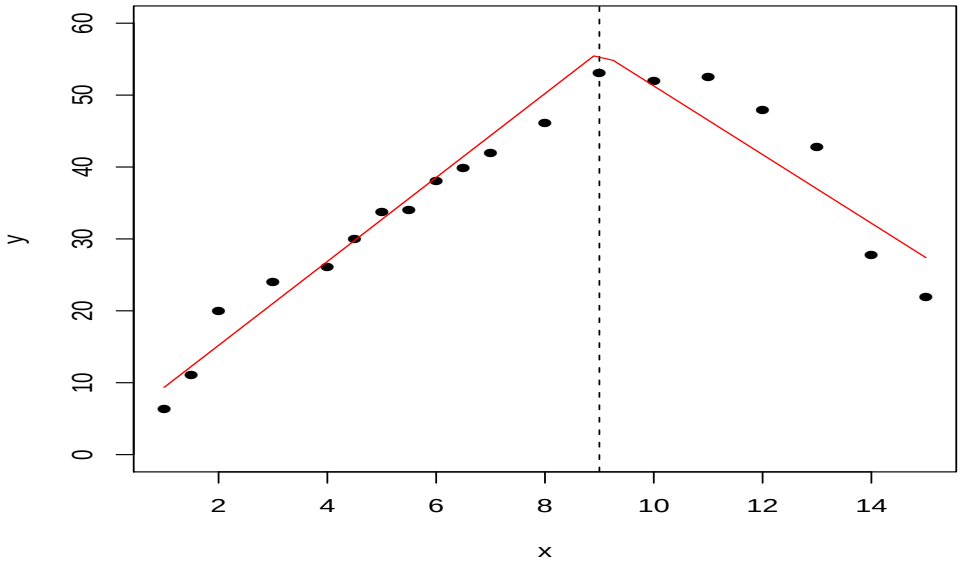


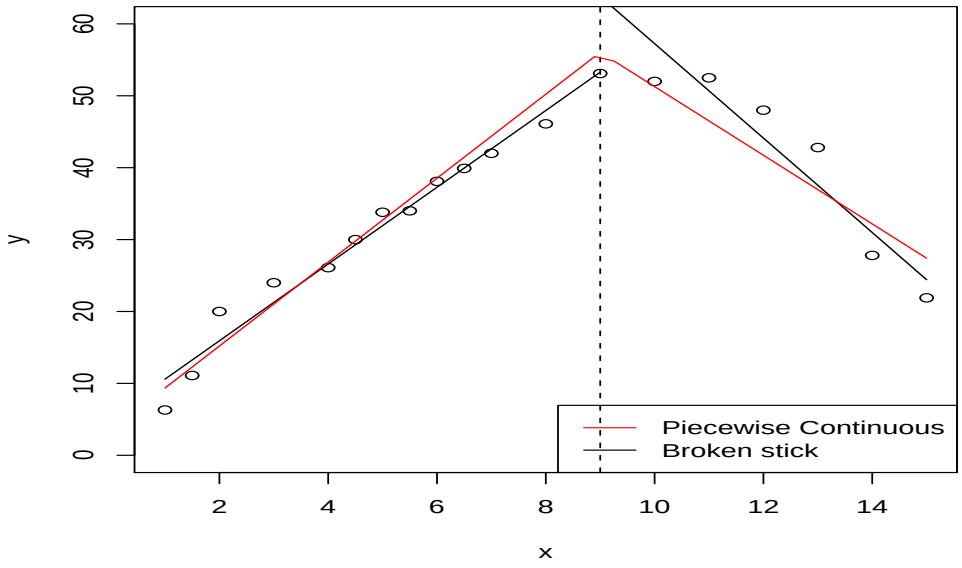
continuous piecewise regression

```
> gc=lm(y~x+br.one(x))
> model.matrix(gc)
```

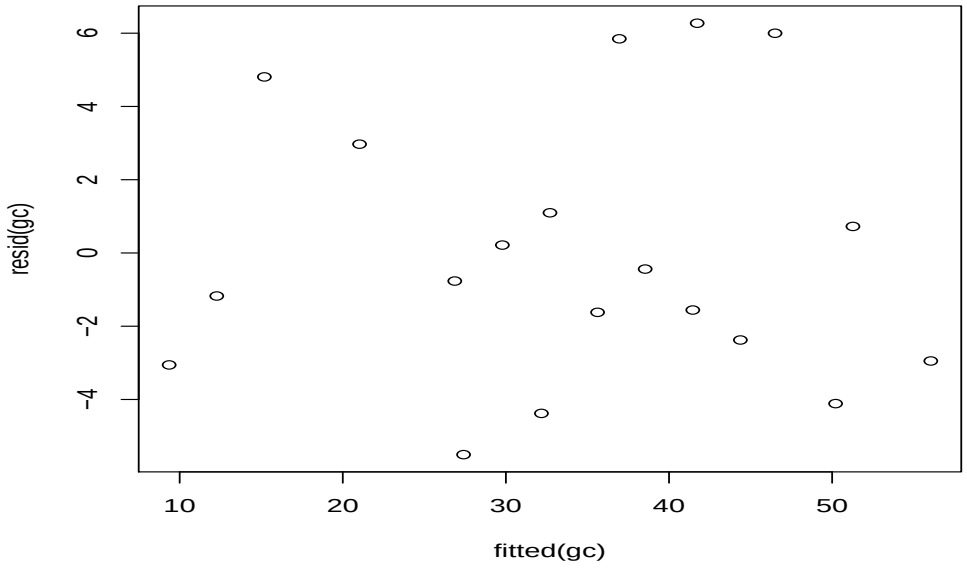
```
   (Intercept)    x br.one(x)
1            1  1.0         0
2            1  1.5         0
3            1  2.0         0
4            1  3.0         0
5            1  4.0         0
```

```
6            1  4.5         0
7            1  5.0         0
8            1  5.5         0
9            1  6.0         0
10           1  6.5         0
11           1  7.0         0
12           1  8.0         0
13           1  9.0         0
14           1 10.0         1
15           1 11.0         2
16           1 12.0         3
17           1 13.0         4
18           1 14.0         5
19           1 15.0         6
attr(,"assign")
[1] 0 1 2

> (b=coef(gc))

(Intercept)           x   br.one(x)
   3.519704    5.836556  -10.610434

> summary(gc)

Call:
lm(formula = y ~ x + br.one(x))

Residuals:
   Min     1Q Median     3Q    Max
-5.505 -2.662 -0.766  2.034  6.273

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   3.5197     2.3256   1.513    0.150
x             5.8366     0.4021  14.515 1.25e-10
br.one(x)   -10.6104     0.8908 -11.912 2.29e-09

Residual standard error: 3.877 on 16 degrees of freedom
Multiple R-squared: 0.9296,        Adjusted R-squared: 0.9208
F-statistic: 105.6 on 2 and 16 DF,  p-value: 6.028e-10

> plot(x,y, type="n", ylim=c(0,60))
> abline(v=t0, lty=2)
> points(x, y, pch=16)
> x.new=seq(min(x),max(x), length=40)
> yfit=predict(gc, data.frame(x=x.new)) #b[1]+b[2]*x.new+b[3]*br.zero(x.new)
> lines(x.new, yfit, col=2)
```






```
> par(mfrow=c(1,1))
> plot(resid(gc)~fitted(gc))
```




```
> c(summary(gb)$adj.r, summary(gc)$adj.r)

[1] 0.9486407 0.9208088

> c(summary(gb)$sigma, summary(gc)$sigma)

[1] 3.122401 3.877194
```

Splines

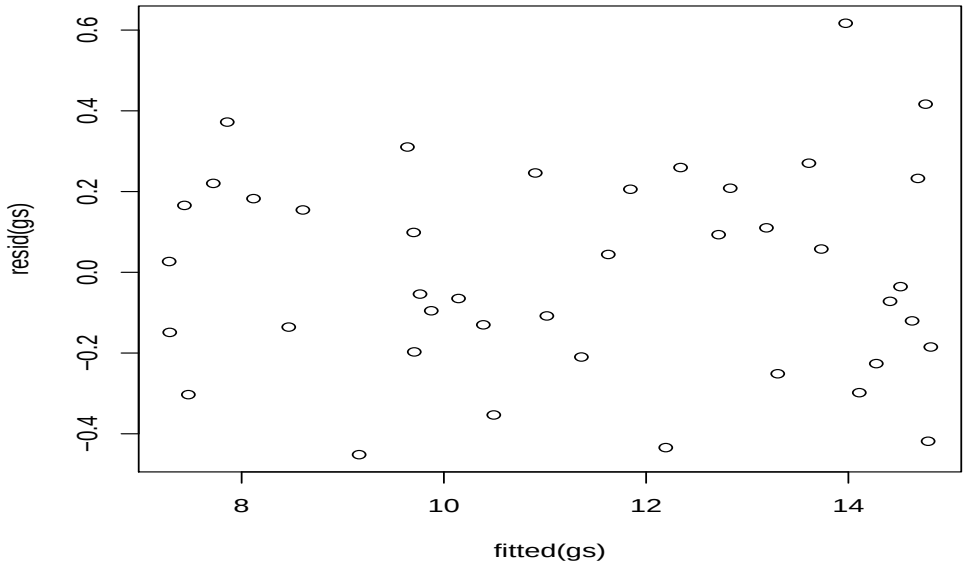
A spline curve is formed by joining together two or more polynomial curves, typically cubics, in such a way that there is a smooth transition from one polynomial to the next.

Knots: join points

```
> vol=read.table("ex-7-2.txt", header=T)
> x=vol[,1]
> y=vol[,2]
> bx=bs(x, knots=c(6.5, 13), degree=3)
> gs=lm(y~bx)
> summary(gs)

Call:
lm(formula = y ~ bx)

Residuals:
     Min       1Q   Median       3Q      Max
-0.45168 -0.18499 -0.03547  0.20577  0.61694
```

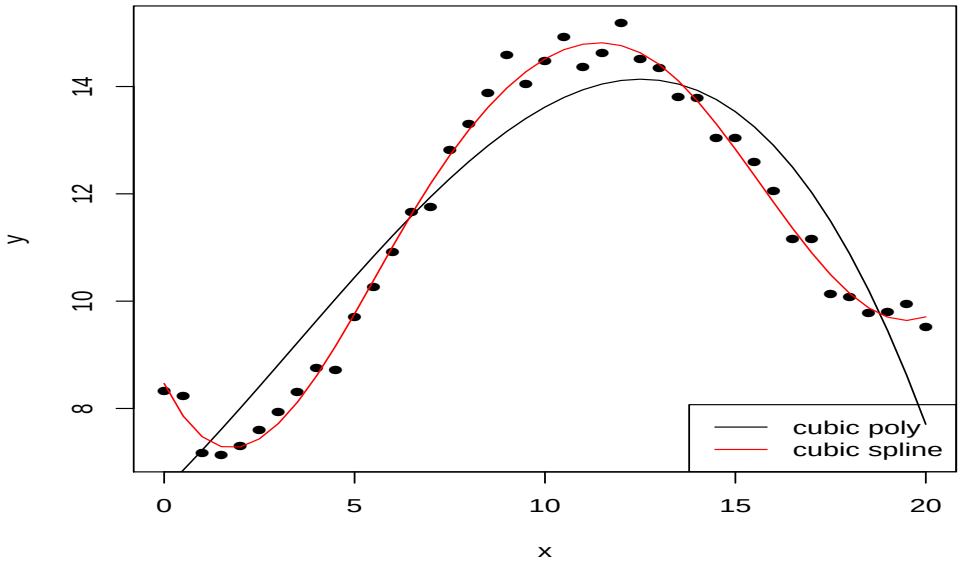
```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   8.4657     0.2005  42.219  < 2e-16
bx1          -3.1484     0.3934  -8.002 2.04e-09
bx2           4.3532     0.2843  15.312  < 2e-16
bx3           8.5518     0.3691  23.169  < 2e-16
bx4           0.5990     0.3059   1.958 0.058192
bx5           1.2414     0.2871   4.324 0.000121

Residual standard error: 0.2678 on 35 degrees of freedom
Multiple R-squared: 0.9904,        Adjusted R-squared: 0.9891
F-statistic: 725.5 on 5 and 35 DF,  p-value: < 2.2e-16
```

```
> plot(resid(gs)~fitted(gs))
```



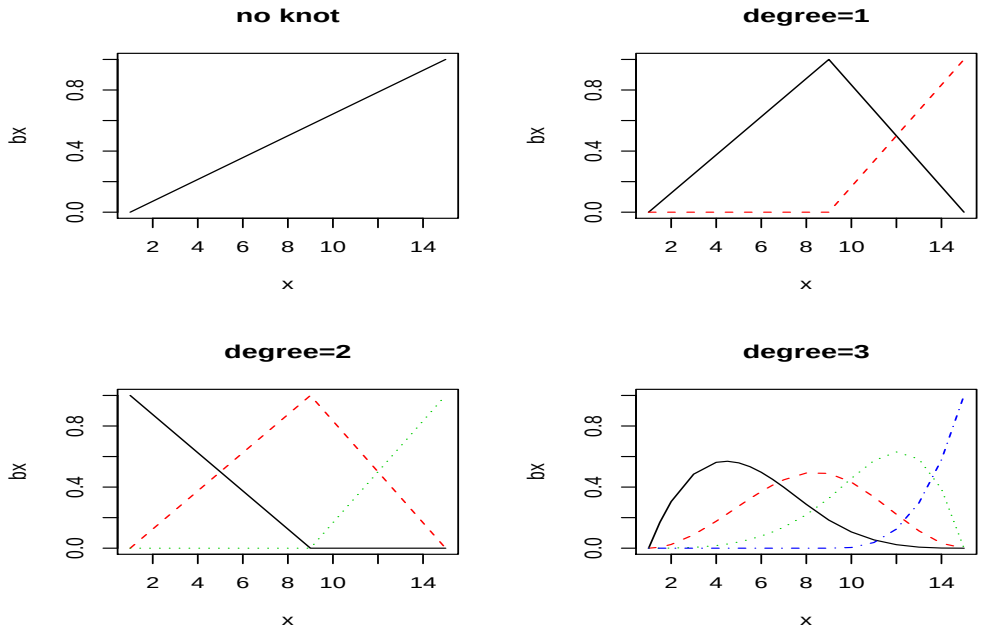

```
> br1=function(x) ifelse(x<6.5, 0, x-6.5)
> br2=function(x) ifelse(x<13, 0, x-13)
> g1=lm(y~x+I(x^2)+I(x^3))
> summary(g1)

Call:
lm(formula = y ~ x + I(x^2) + I(x^3))

Residuals:
    Min      1Q  Median      3Q     Max
-1.3503 -0.7340 -0.1859  0.6440  1.8390

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.4910163  0.5336473  12.163 1.71e-14
```

```
x            0.7031952  0.2339552   3.006 0.004738
I(x^2)       0.0340179  0.0273762   1.243 0.221829
I(x^3)      -0.0033072  0.0008992  -3.678 0.000743

Residual standard error: 0.9335 on 37 degrees of freedom
Multiple R-squared: 0.8773,        Adjusted R-squared: 0.8673
F-statistic: 88.14 on 3 and 37 DF,  p-value: < 2.2e-16

> g2=lm(y~x+I(x^2)+I(x^3)+I(br1(x)^3)+I(br2(x)^3))
> summary(g2)

Call:
lm(formula = y ~ x + I(x^2) + I(x^3) + I(br1(x)^3) + I(br2(x)^3))

Residuals:
     Min       1Q   Median       3Q      Max
-0.45168 -0.18499 -0.03547  0.20577  0.61694

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  8.465678   0.200520  42.219  < 2e-16
x           -1.453124   0.181586  -8.002 2.04e-09
I(x^2)       0.489889   0.043018  11.388 2.54e-13
I(x^3)      -0.029467   0.002848 -10.347 3.44e-12
I(br1(x)^3)  0.024706   0.004039   6.116 5.43e-07
I(br2(x)^3)  0.027112   0.003578   7.577 6.98e-09

Residual standard error: 0.2678 on 35 degrees of freedom
Multiple R-squared: 0.9904,        Adjusted R-squared: 0.9891
F-statistic: 725.5 on 5 and 35 DF,  p-value: < 2.2e-16

> anova(g1, g2)

Analysis of Variance Table

Model 1: y ~ x + I(x^2) + I(x^3)
Model 2: y ~ x + I(x^2) + I(x^3) + I(br1(x)^3) + I(br2(x)^3)
  Res.Df    RSS Df Sum of Sq      F    Pr(>F)
1     37 32.244
2     35  2.510  2    29.734 207.29 < 2.2e-16

> plot(x,y, pch=16)
> lines(x, fitted(g1))
> lines(x, fitted(g2), col=2)
> legend("bottomright", legend=c("cubic poly","cubic spline"), lty=1, col=c(1,2))
```

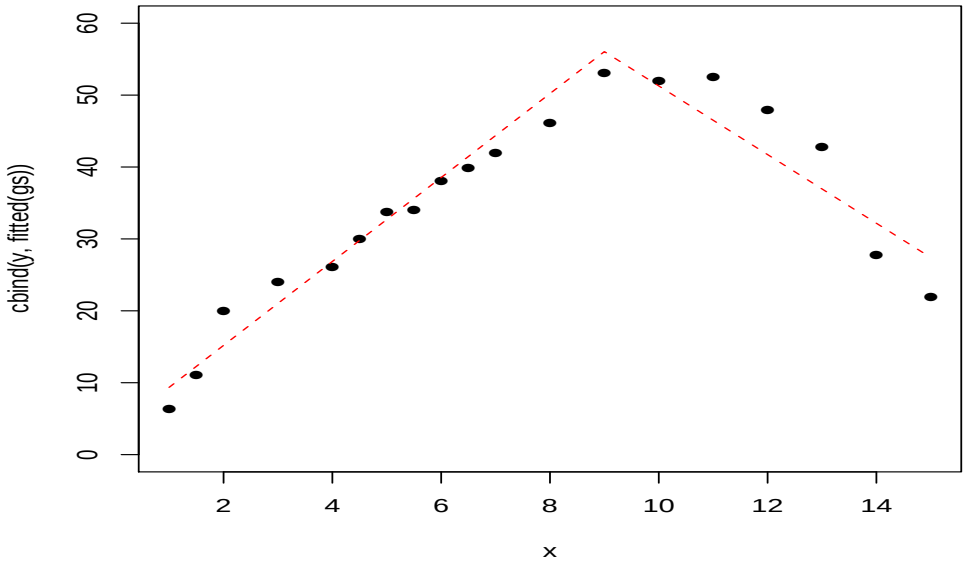


Generate the B-spline basis matrix for a polynomial spline.

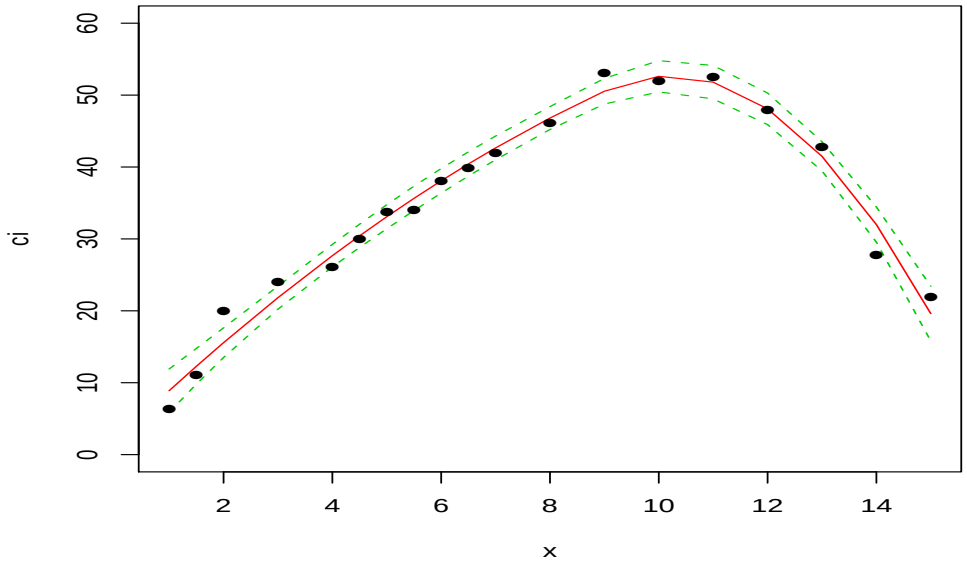
```
> x=hardwood[,1]
> y=hardwood[,2]
> t0=9
> library(splines)
> par(mfrow=c(2,2))
> bx=bs(x, degree=1)
> matplot(x, bx, type="l", main="no knot")
> bx=bs(x, knots=t0, degree=1)
> matplot(x, bx, type="l", main="degree=1")
> bx=bs(x, knots=t0, degree=1, intercept=T)
> matplot(x, bx, type="l", main="degree=2")
> bx=bs(x, knots=t0, degree=3)
> matplot(x, bx, type="l",  main="degree=3")
```



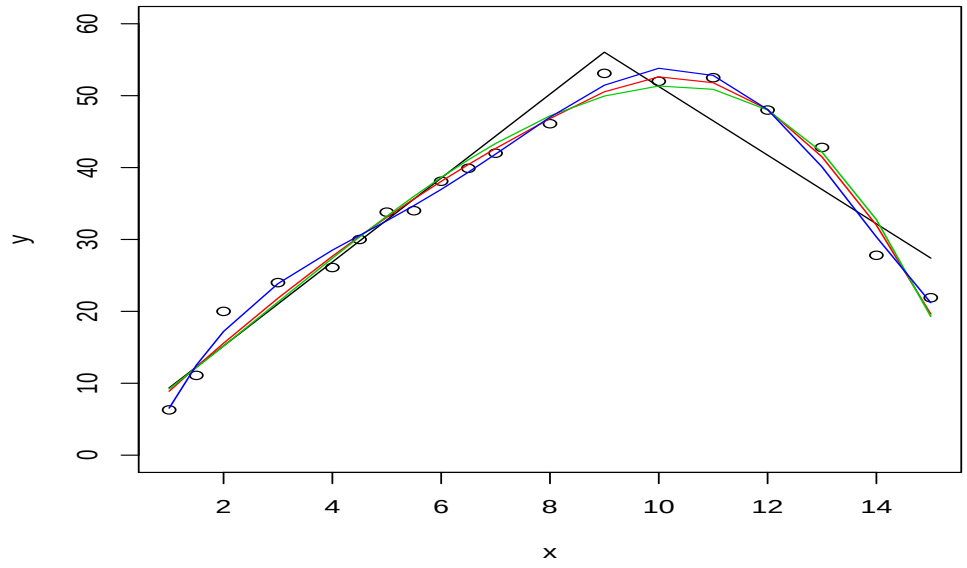

```
> par(mfrow=c(1,1))
> bx=bs(x, knots=t0, degree=1)
> gs=lm(y~bx)
> summary(gs)

Call:
lm(formula = y ~ bx)

Residuals:
   Min     1Q Median     3Q    Max
-5.505 -2.662 -0.766  2.034  6.273

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    9.356      1.973   4.742 0.000221
bx1           46.692      3.217  14.515 1.25e-10
bx2           18.049      3.029   5.958 2.01e-05

Residual standard error: 3.877 on 16 degrees of freedom
Multiple R-squared: 0.9296,        Adjusted R-squared: 0.9208
F-statistic: 105.6 on 2 and 16 DF,  p-value: 6.028e-10

> matplot(x, cbind(y,fitted(gs)), type="pl",ylim=c(0,60), pch=16)
```



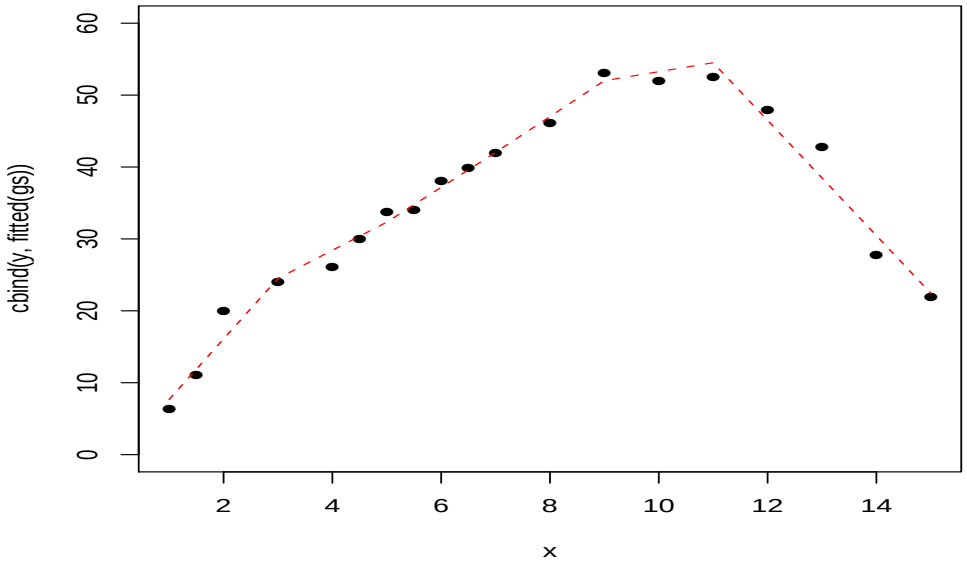

```
> bx=bs(x, knots=t0, degree=2)
> gs2=lm(y~bx)
> summary(gs2)

Call:
lm(formula = y ~ bx)

Residuals:
    Min      1Q  Median      3Q     Max
-4.1946 -0.9460 -0.4294  1.0129  4.4297

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    8.890      1.416   6.280 1.48e-05
bx1           27.565      2.674  10.310 3.34e-08
bx2           52.217      2.093  24.949 1.25e-13
bx3           10.718      2.364   4.534 0.000395

Residual standard error: 2.183 on 15 degrees of freedom
Multiple R-squared: 0.9791,        Adjusted R-squared: 0.9749
F-statistic: 234.1 on 3 and 15 DF,  p-value: 8.131e-13

> ci=predict(gs2, interval="confidence")
> matplot(x, ci, type="l", ylim=c(0,60), col=c(2,3,3),lty=c(1,2,2))
> points(x,y, pch=16)
```



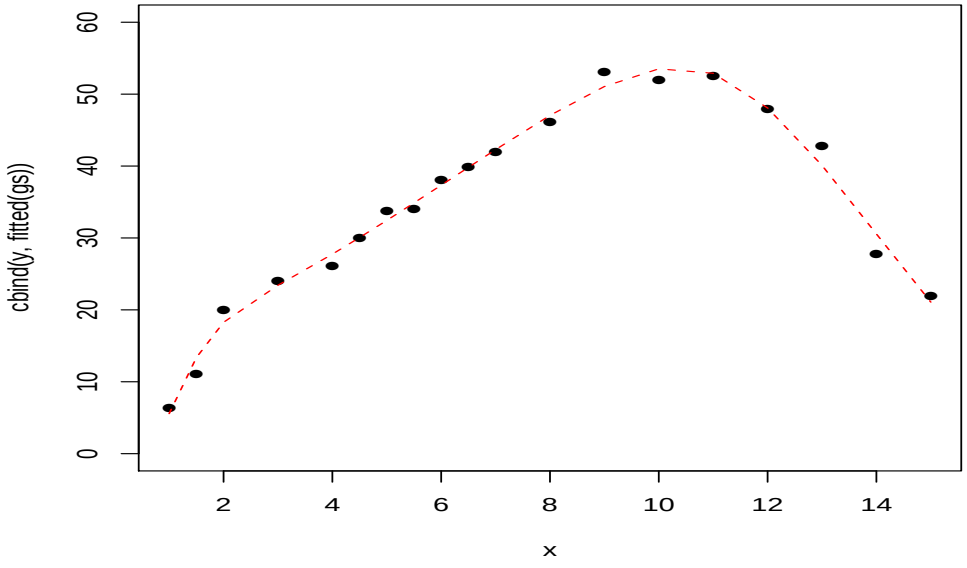

```
> plot(x,y, ylim=c(0,60))
> lines(x,fitted(lm(y~bs(x, knots=9, degree=1))), col=1)
> lines(x,fitted(lm(y~bs(x, knots=9, degree=2))), col=2)
> lines(x,fitted(lm(y~bs(x, knots=9, degree=3))), col=3)
> lines(x,fitted(lm(y~bs(x, knots=9, degree=4))), col=4)
```



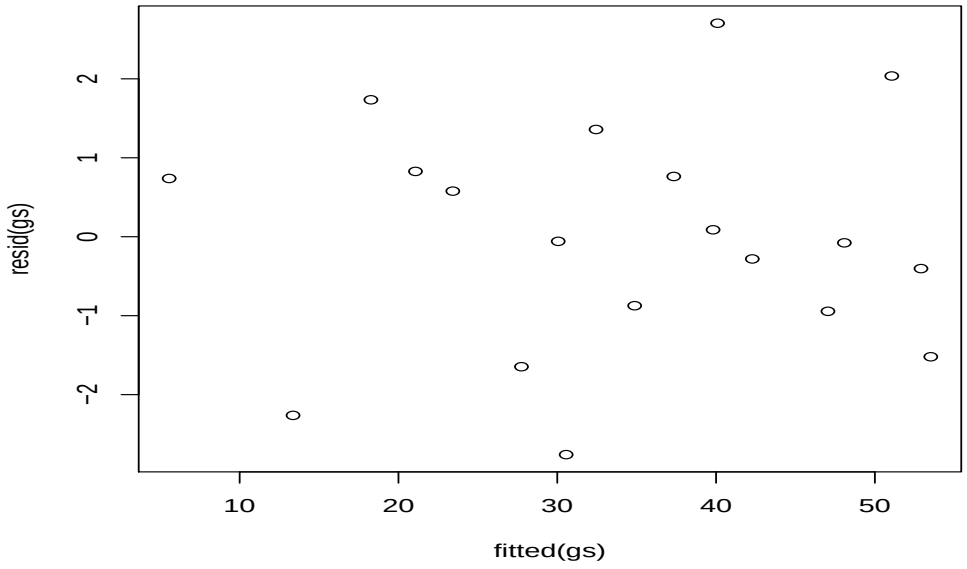

```
> bx=bs(x, knots=c(3,5,7,9,11), degree=1)
> gs=lm(y~bx)
> matplot(x, cbind(y,fitted(gs)), type="pl",ylim=c(0,60), pch=16)
```




```
> bx=bs(x, knots=c(3,5,7,9,11), degree=3)
> gs=lm(y~bx)
> matplot(x, cbind(y,fitted(gs)), type="pl",ylim=c(0,60), pch=16)
```




```
> par(mfrow=c(1,1))
> plot(resid(gs)~fitted(gs))
```




```
> c(summary(gb)$adj.r, summary(gc)$adj.r, summary(gs)$adj.r)

[1] 0.9486407 0.9208088 0.9799102

> c(summary(gb)$sigma, summary(gc)$sigma, summary(gs)$sigma)

[1] 3.122401 3.877194 1.952840
```

Kernel regression

A smooth function K such that p215

Some commonly used kernels:

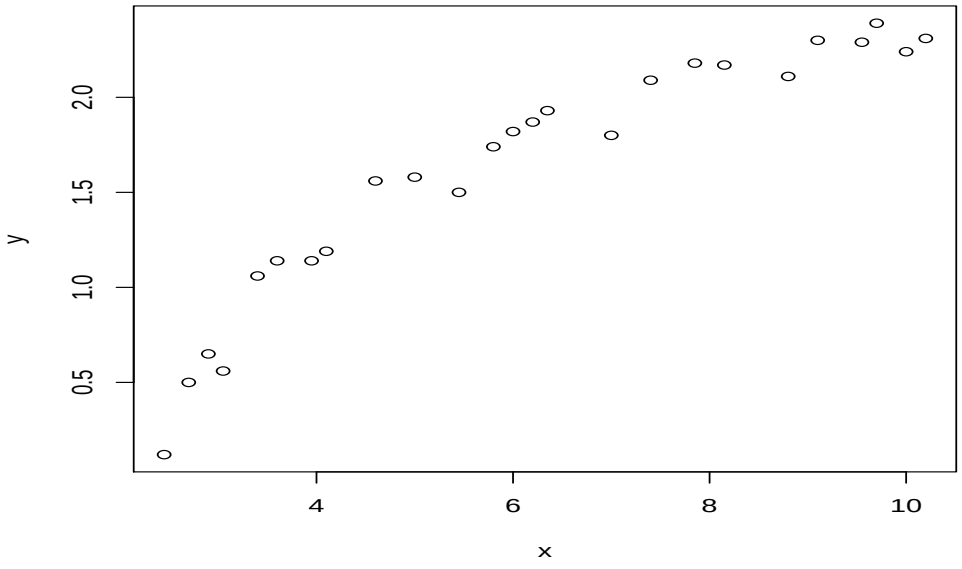
box, triangle, parzen, Epanechinikov, tricube, Gaussian

Let $b > 0$ be a positive number, call the bandwidth. and the weights for the knernel smoother are

given by

$$w_{ij} = \frac{K(\frac{x_i - x_j}{b})}{\sum_{k=1}^{n} K(\frac{x_i - x_k}{b})}$$

```
> windmill=read.table("ex-5-2.csv", header=T, sep=",")
> x=windmill[,2]
> y=windmill[,3]
> ord=order(x)
> x=x[ord]
> y=y[ord]
> #library(locfit)
> #out=locfit(y~x)
> #print(out)
> #summary(out)
> plot(x,y)
```



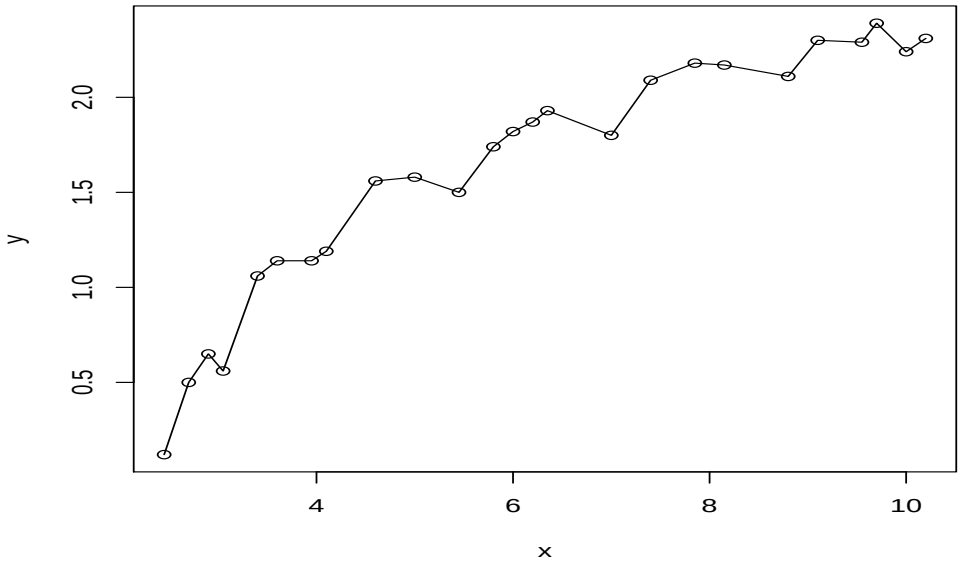
locfit.raw(x, y, weights=1, cens=NULL, base=0, scale=F, alpha=0.7, deg=2, kern="tcub", kt="sph", acri="none", basis=list(NULL), deriv=numeric(0), dc=F, family, link="default", xlim, renorm=F, ev="tree", flim, mg=10, cut=0.8, maxk=100, itype="default", mint=20, maxit=20, debug=0, geth=F, sty=rep(1,d))

kern: Weight function, default = "tcub". Other choices are "rect", "trwt", "tria", "epan", "bisq" and "gauss". Choices may be restricted when derivatives are required; e.g. for confidence bands and some bandwidth selectors.

Lowess (loess)

locally weighted regression

```
> plot(x,y)
> lines(loess(y~x))
```


y
0.5
1.0
1.5
2.0
4
6
8
10
x