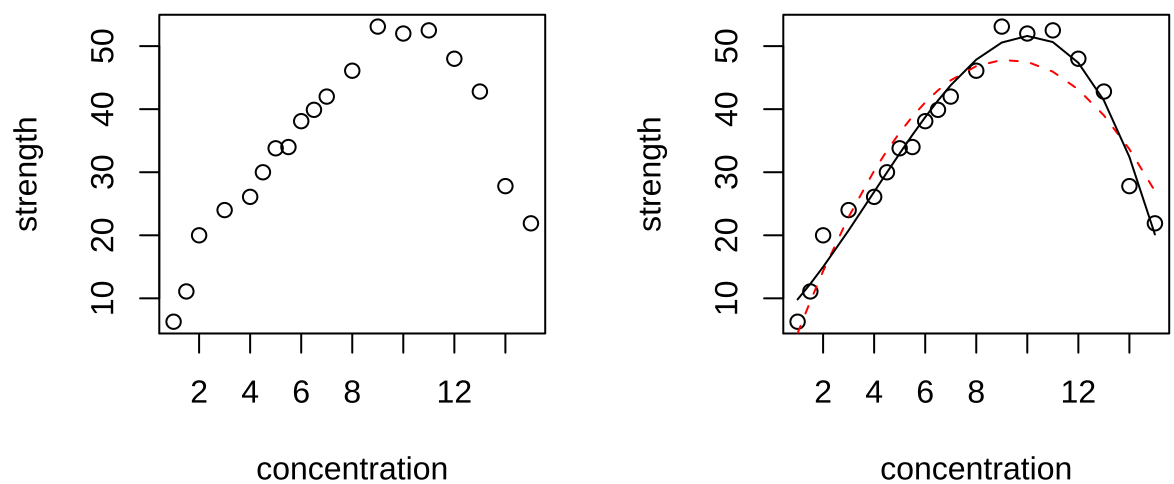
1 Polynomial Regression: The Hardwood Data (MPV 7.1)

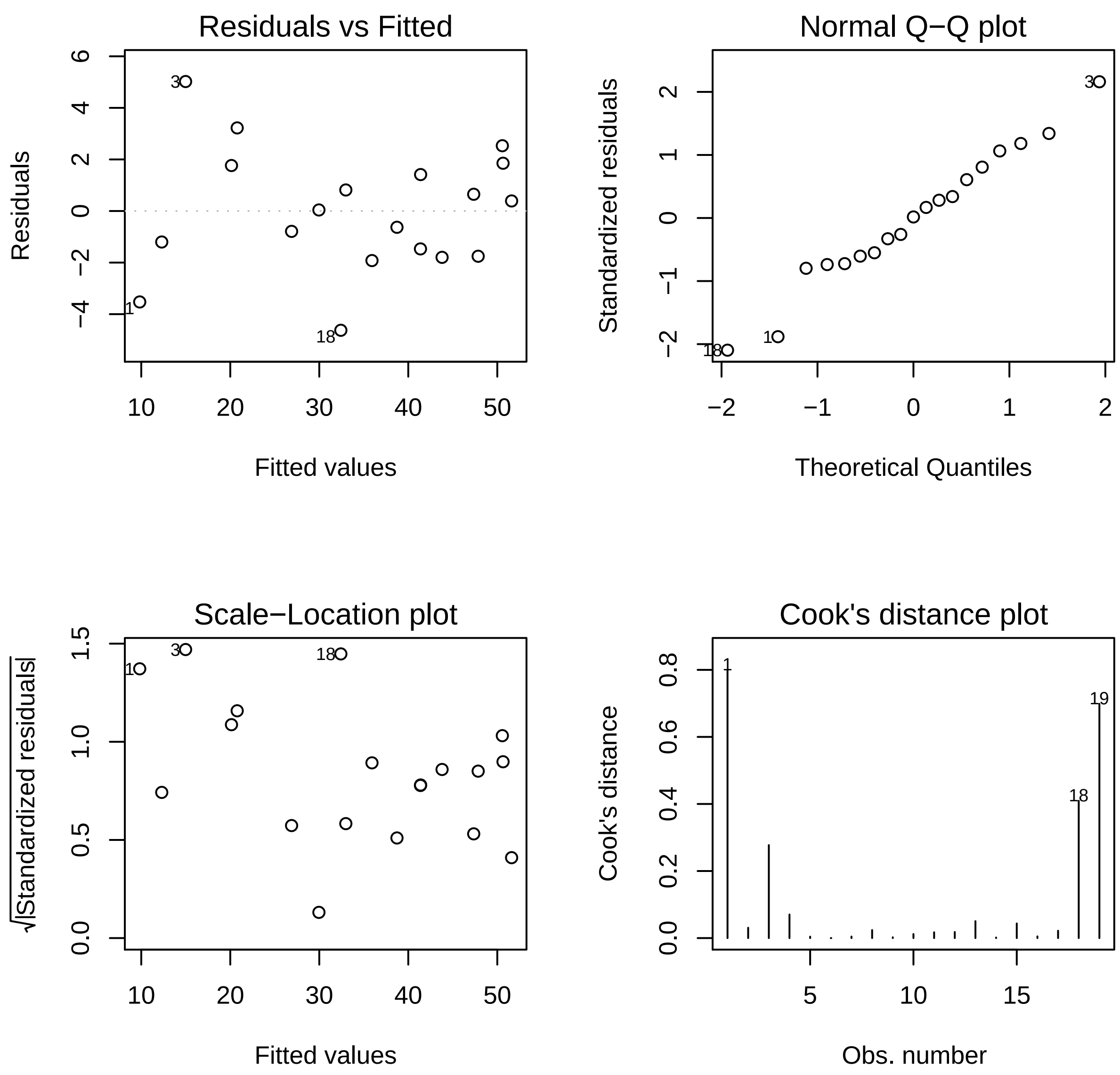
- Data concerning the strength of kraft paper and the percentage of hardwood in the batch of pulp from which the paper was produced.




```
> dat=read.table("hardwood.dat", h=T)
> attach(dat)
> pdf("hardwood.pdf", w=6.5, h=3.5)  # save graph to a pdf file
> par(mfrow=c(1,2))  # two plots in one row
> plot(strength~concentration)
>
> ## centering data, reduces vif
> mean(concentration)
[1] 7.263158
> conc0 = concentration - mean(concentration)
> g2=lm(strength~conc0+I(conc0^2))
> summary(g2)
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 45.29497    1.48287   30.55 1.29e-15 ***
conc0        2.54634    0.25384   10.03 2.63e-08 ***
I(conc0^2)  -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

> g3=lm(strength~conc0+I(conc0^2)+I(conc0^3))
> summary(g3)
             Estimate Std. Error t value Pr(>|t|)
```

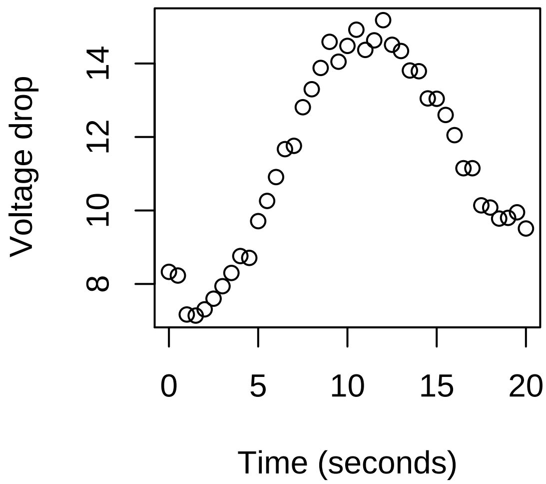
```
(Intercept) 44.975562   0.869032  51.754  < 2e-16 ***
conc0        4.339394   0.350978  12.364 2.87e-09 ***
I(conc0^2)  -0.548873   0.039199 -14.002 5.11e-10 ***
I(conc0^3)  -0.055188   0.009789  -5.638 4.72e-05 ***

Residual standard error: 2.585 on 15 degrees of freedom
Multiple R-Squared: 0.9707,Adjusted R-squared: 0.9648
F-statistic: 165.4 on 3 and 15 DF,  p-value: 1.025e-11

> g4=lm(strength~conc0+I(conc0^2)+I(conc0^3)+I(conc0^4))
> summary(g4)
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 44.165124   1.072816  41.167 5.21e-16 ***
conc0        4.115267   0.388740  10.586 4.59e-08 ***
I(conc0^2)  -0.394166   0.129932  -3.034  0.00894 **
I(conc0^3)  -0.045268   0.012479  -3.628  0.00274 **
I(conc0^4)  -0.003505   0.002812  -1.247  0.23298

Residual standard error: 2.539 on 14 degrees of freedom
Multiple R-Squared: 0.9736,Adjusted R-squared: 0.9661
F-statistic: 129.1 on 4 and 14 DF,  p-value: 6.994e-11

> library(car)
> vif(g2)
     conc0 I(conc0^2)
  1.097050   1.097050
> vif(g3)
     conc0 I(conc0^2) I(conc0^3)
  6.132746   1.291093   6.817759
> vif(g4)
     conc0 I(conc0^2) I(conc0^3) I(conc0^4)
  7.801336  14.709081  11.488967  19.582862

> # compare the vif without centering
> h3=lm(strength~concentration+I(concentration^2)+I(concentration^3))
> summary(h3)
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)         5.648395   2.954663   1.912   0.0752 .
concentration       3.578489   1.565854   2.285   0.0373 *
I(concentration^2)  0.653635   0.231330   2.826   0.0128 *
I(concentration^3) -0.055188   0.009789  -5.638 4.72e-05 ***

Residual standard error: 2.585 on 15 degrees of freedom
Multiple R-Squared: 0.9707,Adjusted R-squared: 0.9648
F-statistic: 165.4 on 3 and 15 DF,  p-value: 1.025e-11

> vif(h3)
     concentration I(concentration^2) I(concentration^3)
          122.0670           701.7123           270.3496

> ## polynomial of degree=3 is the most appropriate
> plot(strength~concentration)
> i=order(concentration)
> lines(concentration[i], fitted(g2)[i], col=2, lty=2)
```

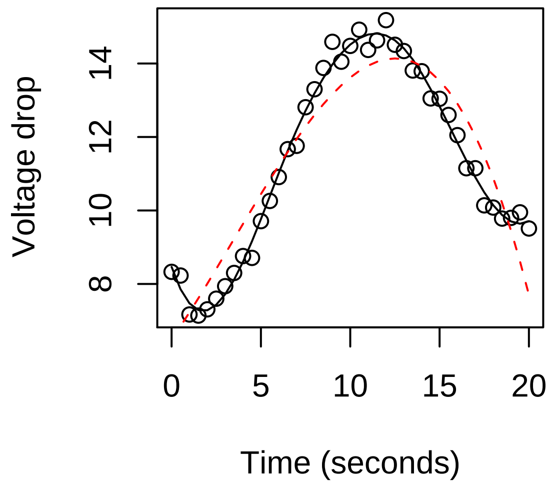
```
> lines(concentration[i], fitted(g3)[i], col=1, lty=1)
> dev.off()  # close file

> # Check residual plots for the cubic model
> pdf("hardwood2.pdf", w=6.5, h=6.5)
> par(mfrow=c(2,2))   # two plots in two rows
> plot(g3, 1:4)
> dev.off()
```



2 Splines: The Voltage Drop Data (MPV 7.2)

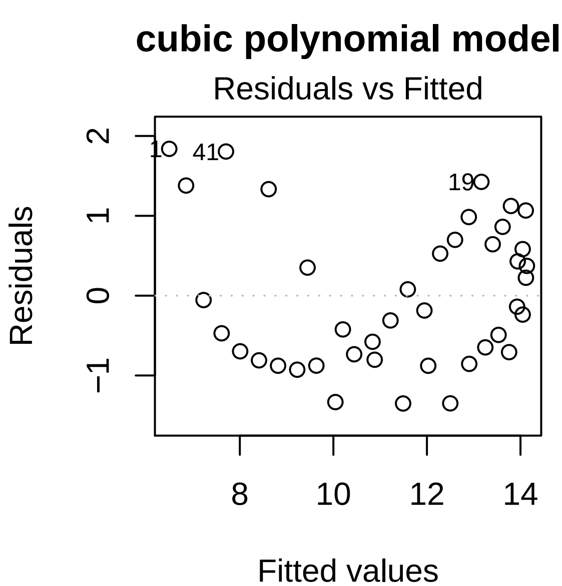
- The battery voltage drop (y) in a guided missile motor observed over the time of missile flight.

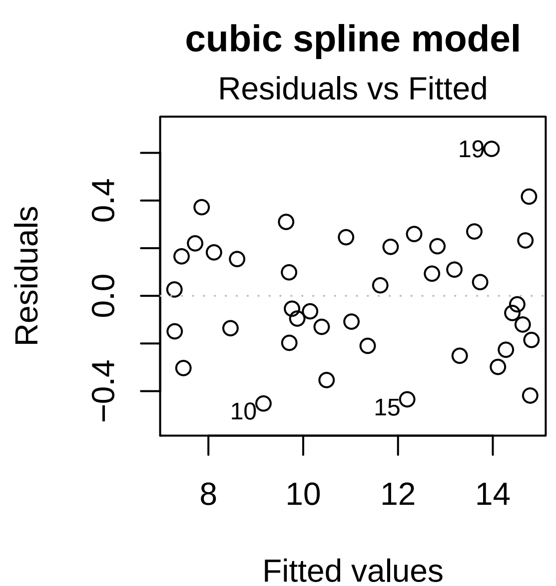





```
> dat=read.table("voltage.dat", h=T)
> attach(dat)
> par(mfrow=c(1,2))
> plot(dat, ylab="Voltage drop", xlab="Time (seconds)")

> ## first try cubic polynomials
> g3=lm(drop~time+I(time^2)+I(time^3))
> summary(g3)
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.4910163  0.5336473  12.163 1.71e-14 ***
time         0.7031952  0.2339552   3.006 0.004738 **
I(time^2)    0.0340179  0.0273762   1.243 0.221829
I(time^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

> # now try cubic spline fitting
> xplus = function(x) ifelse(x>=0, x, 0)
> time6.5=xplus(time-6.5)
> time13=xplus(time-13)
> gs=lm(drop~time+I(time^2)+I(time^3)+I(time6.5^3)+I(time13^3))
> summary(gs)
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   8.465678   0.200520  42.219  < 2e-16 ***
time         -1.453124   0.181586  -8.002 2.04e-09 ***
I(time^2)     0.489889   0.043018  11.388 2.54e-13 ***
```

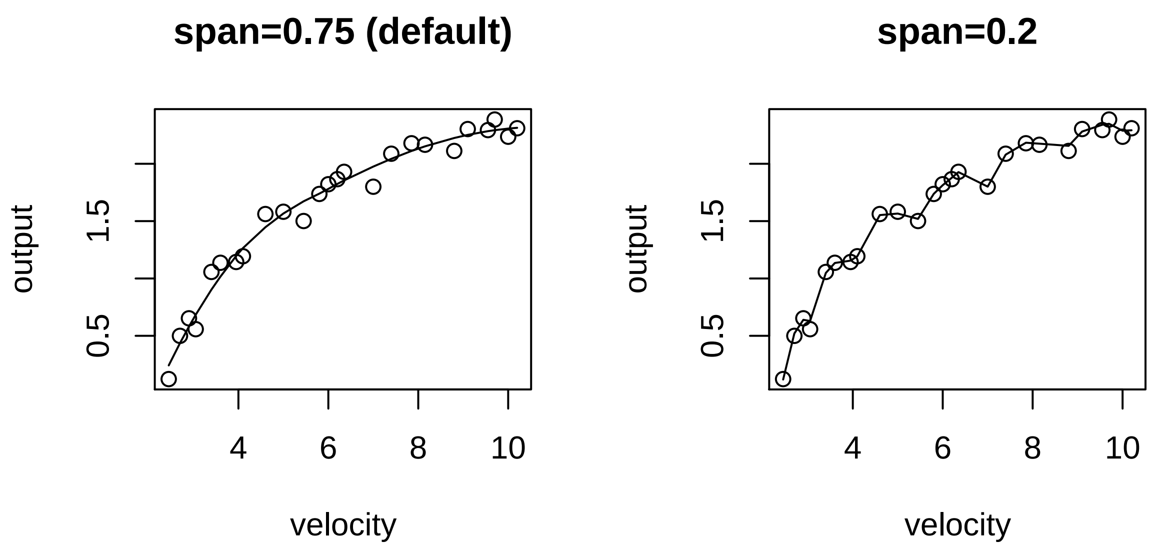
```
I(time^3)    -0.029467   0.002848 -10.347 3.44e-12 ***
I(time6.5^3)  0.024706   0.004039   6.116 5.43e-07 ***
I(time13^3)   0.027112   0.003578   7.577 6.98e-09 ***

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

> # fitted curves
> plot(dat, ylab="Voltage drop", xlab="Time (seconds)")
> i=order(time)
> lines(time[i], fitted(gs)[i], col=1, lty=1)
> lines(time[i], fitted(g3)[i], col=2, lty=2)

> # residual plots
> plot(g3, 1, main="cubic polynomial model")
> plot(gs, 1, main="cubic spline model")

> # how about polynomial of degree 4?
> g4=lm(drop~time+I(time^2)+I(time^3)+I(time^4), dat)
> summary(g4)
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  8.496e+00  1.750e-01   48.55  < 2e-16 ***
time        -1.553e+00  1.244e-01  -12.48 1.23e-14 ***
I(time^2)    5.563e-01  2.576e-02   21.59  < 2e-16 ***
I(time^3)   -4.426e-02  1.947e-03  -22.73  < 2e-16 ***
I(time^4)    1.024e-03  4.827e-05   21.21  < 2e-16 ***

Residual standard error: 0.2576 on 36 degrees of freedom
Multiple R-Squared: 0.9909,Adjusted R-squared: 0.9899
F-statistic: 980.3 on 4 and 36 DF,  p-value: < 2.2e-16
```



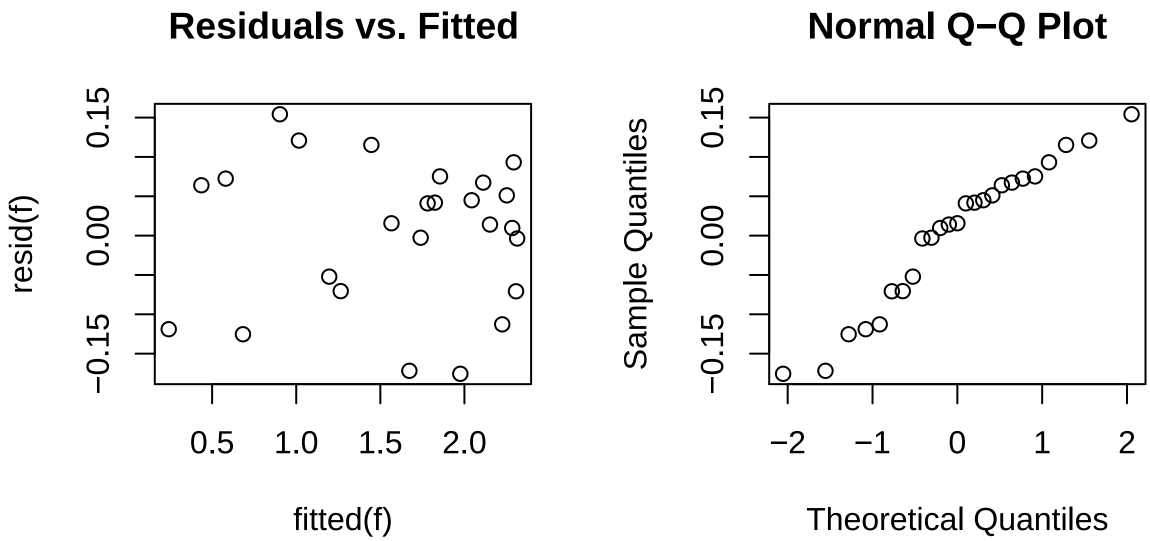


3 Loess Regression: The Windmill Data (MPV 7.3)

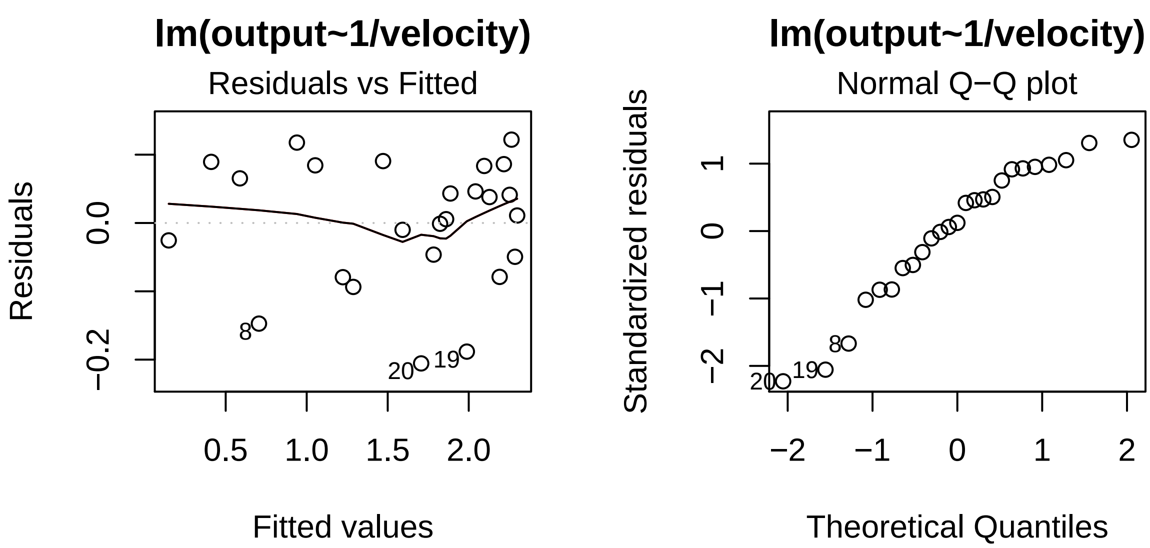
- Data collected by an engineer to investigate the relationship of wind velocity and the DC electrical output for a windmill.




```
> dat=read.table("windmill.dat", h=T)
> attach(dat)
> par(mfrow=c(1,2))
> plot(output~velocity, main="span=0.75 (default)")
> f=loess(output~velocity)  # default span=0.75
> i=order(f$x)
> lines(f$x[i], f$fitted[i])
> plot(output~velocity, main="span=0.2")
> f=loess(output~velocity, span=.2)
> lines(f$x[i], f$fitted[i])

> f=loess(output~velocity)  # default span=0.75
> summary(f)
Number of Observations: 25
Equivalent Number of Parameters: 4.41
Residual Standard Error: 0.1017
Trace of smoother matrix: 4.84

Control settings:
  normalize:  TRUE
  span     :  0.75
  degree   :  2
  family   :  gaussian
  surface  :  interpolate   cell = 0.2

> # check residual plots
> plot(resid(f)~fitted(f), main="Residuals vs. Fitted")
> qqnorm(resid(f))
```

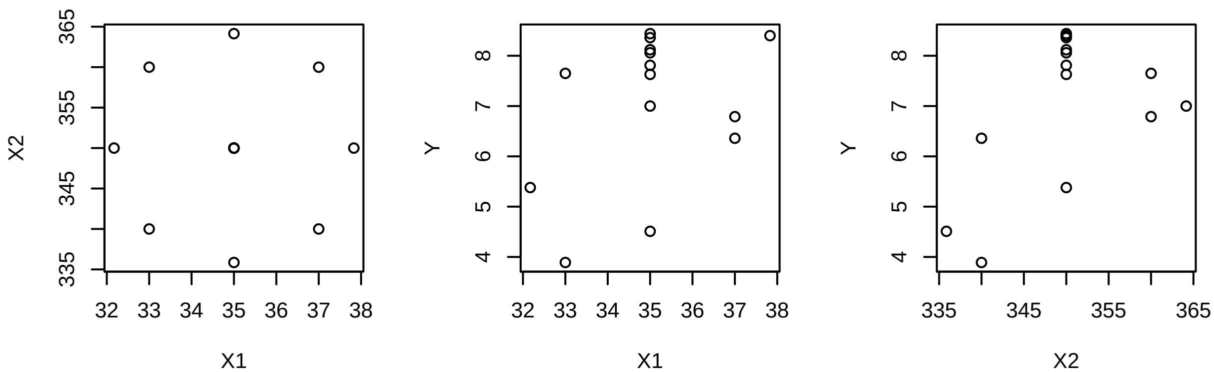



```
> # compare with transforming the predictor
> g=lm(output~I(1/velocity)); summary(g)
               Estimate Std. Error t value Pr(>|t|)
(Intercept)     2.9789     0.0449   66.34   <2e-16 ***
I(1/velocity)  -6.9345     0.2064  -33.59   <2e-16 ***

Residual standard error: 0.09417 on 23 degrees of freedom
Multiple R-Squared:  0.98,Adjusted R-squared: 0.9792
F-statistic:  1128 on 1 and 23 DF,  p-value: < 2.2e-16

> plot(g,1, panel=panel.smooth, main="lm(output~1/velocity)")
> lines(lowess(fitted(g), resid(g), f=2/3))  # the default smoother in the residual plot
> plot(g,2,  main="lm(output~1/velocity)")
```



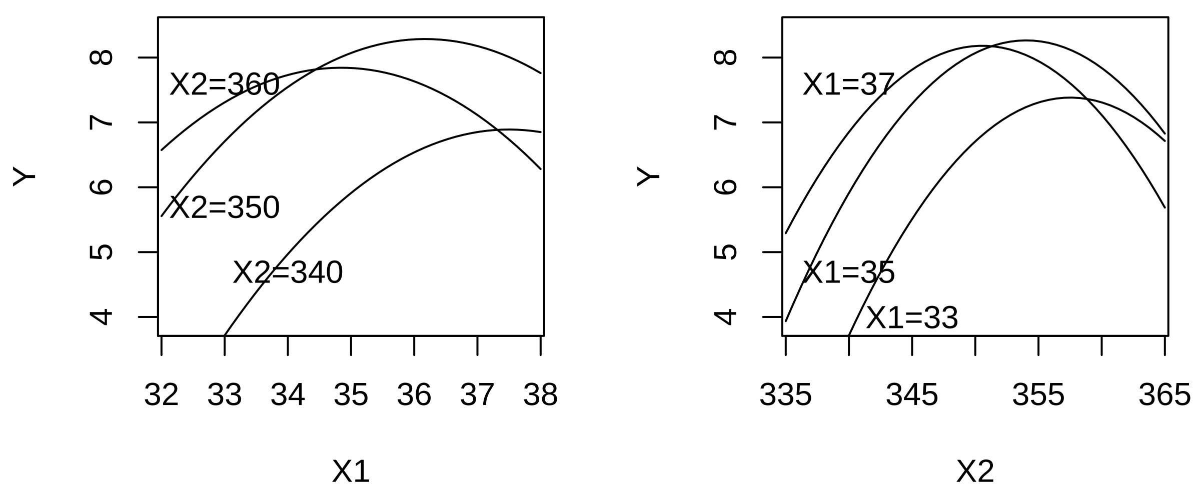
4 Polynomial models with two predictors: cakes example

- Weisberg 6.1.1: data from a small experiment on baking packaged cake mies
- two factors: X1=baking time in minutes and X2=baking temperature in degrees F
- response Y=average palatability score of four cakes baked at a given combination of (X1, X2), with higher values desirable.
- there are 6 observations taken at the center point (35, 350)

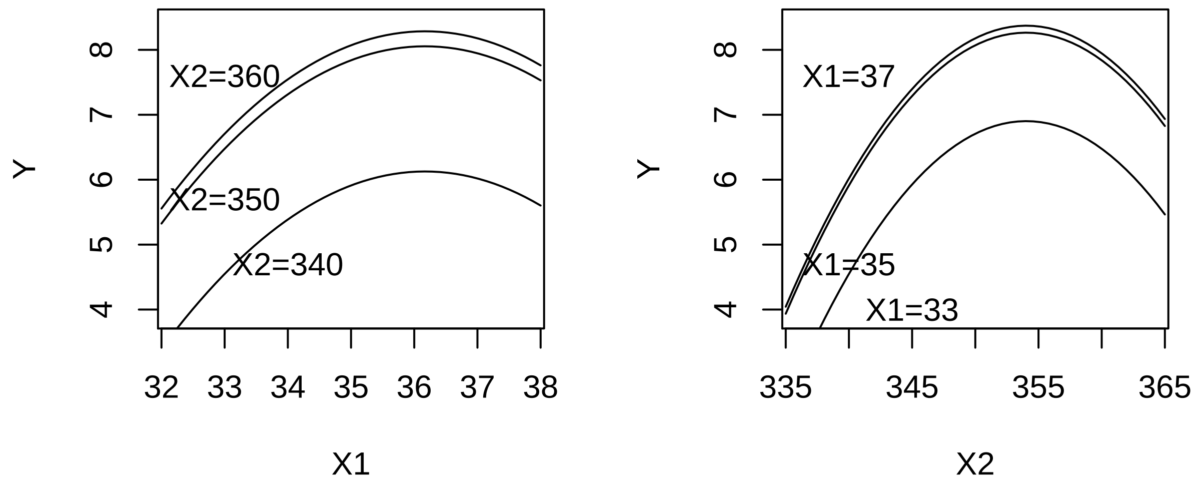



```
> library(alr3)
> data(cakes);  attach(cakes)
> plot(X1, X2);  plot(X1, Y);  plot(X2, Y)

> m1 <- lm(Y ~ X1 + X2 + I(X1^2) + I(X2^2) + X1:X2, data=cakes)
> summary(m1)
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -2.204e+03  2.416e+02  -9.125 1.67e-05 ***
X1           2.592e+01  4.659e+00   5.563 0.000533 ***
X2           9.918e+00  1.167e+00   8.502 2.81e-05 ***
I(X1^2)     -1.569e-01  3.945e-02  -3.977 0.004079 **
I(X2^2)     -1.195e-02  1.578e-03  -7.574 6.46e-05 ***
X1:X2       -4.163e-02  1.072e-02  -3.883 0.004654 **

Residual standard error: 0.4288 on 8 degrees of freedom
Multiple R-Squared: 0.9487,Adjusted R-squared: 0.9167
F-statistic:  29.6 on 5 and 8 DF,  p-value: 5.864e-05

> # create a function to draw the fitted curves
> plot.fittedcurves=function(m1)
+ {
+     par(mfrow=c(1,2)) # two plots in one row
+     plot(X1,Y,type="n")
+     X1new <- seq(32,38,len=50)
+     lines(X1new,predict(m1,newdata=data.frame(X1=X1new,X2=rep(340,50))))
+     lines(X1new,predict(m1,newdata=data.frame(X1=X1new,X2=rep(350,50))))
+     lines(X1new,predict(m1,newdata=data.frame(X1=X1new,X2=rep(360,50))))
+     text(34,4.7,"X2=340")
```

```
+     text(33,5.7,"X2=350")
+     text(33,7.6,"X2=360")
+
+     plot(X2,Y,type="n")
+     X2new <- seq(335,365,len=50)
+     lines(X2new,predict(m1,newdata=data.frame(X1=rep(33,50),X2=X2new)))
+     lines(X2new,predict(m1,newdata=data.frame(X1=rep(35,50),X2=X2new)))
+     lines(X2new,predict(m1,newdata=data.frame(X1=rep(37,50),X2=X2new)))
+     text(345,4.0,"X1=33")
+     text(340,4.7,"X1=35")
+     text(340,7.6,"X1=37")
+ }
> plot.fittedcurves(m1)  # with interaction
```




```
> # compare with second-order model without interaction
> m2 = lm(Y ~ X1 + X2 + I(X1^2) + I(X2^2) , data=cakes)
> summary(m2)
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.695e+03  3.247e+02  -5.219 0.000550 ***
X1           1.135e+01  4.423e+00   2.566 0.030404 *
X2           8.461e+00  1.769e+00   4.784 0.000996 ***
I(X1^2)     -1.569e-01  6.317e-02  -2.483 0.034791 *
I(X2^2)     -1.195e-02  2.527e-03  -4.730 0.001075 **

Residual standard error: 0.6866 on 9 degrees of freedom
Multiple R-Squared: 0.852,Adjusted R-squared: 0.7863
F-statistic: 12.96 on 4 and 9 DF,  p-value: 0.0008913

> plot.fittedcurves(m2)  # without interaction
```




```
> ## estimated contour plots
> x1=seq(32, 38, len=50);  x2=seq(335, 365, len=50)
> f=function(x1, x2) -2204.485+25.917558*x1+9.918267*x2-0.156875*x1^2-0.011950*x2^2-0.041625*x1*x2
> z=outer(x1, x2, f)
> contour(x1, x2, z, xlab="X1", ylab="X2", main="contour plot w. interaction")
> f=function(x1, x2)   -1694.579+11.348808*x1 +8.461392*x2 -0.156875*x1^2 -0.011950*x2^2
> z=outer(x1, x2, f)
> contour(x1, x2, z, xlab="X1", ylab="X2",  main="contour plot w/o interaction")
```