Notes for Thursday Jan 14, 1997

1 Simple regression and transformations

The purpose of this set of examples is to review and extend some of the techniques of linear regression that you learned in 36-401.

```
402 > help(air)

New York Ozone Concentration

SUMMARY:
       A data frame with 111 observations (rows), and 4 variables
       (columns), taken from an environmental study that measured
       the four variables ozone,  solar  radiation,  temperature,
       and wind speed for 111 consecutive days.

DATA DESCRIPTION:
ozone:    surface concentration of ozone in New  York,  in  parts
       per million.
radiation:     solar radiation
temperature:   observed temperature, in degrees Fahrenheit.
wind:     wind speed, in miles per hour.

SOURCE:
       John M. Chambers and Trevor J. Hastie, (eds.)  Statistical
       Models in S, Wadsworth and Brooks, Pacific Grove, CA 1992,
       pg. 348.

EXAMPLES:
       pairs(air)
```

> *Roughly speaking, the question is: which of the other three variables are good predictors of ozone levels?*

A little spelunking with search() and objects(where=6), etc. shows that this data exists in two built-in SPLUS data sets:

- `ozone.data` is the raw data
- `air` is the same data set but with `ozone` replaced with `ozone`$^{(1/3)}$.

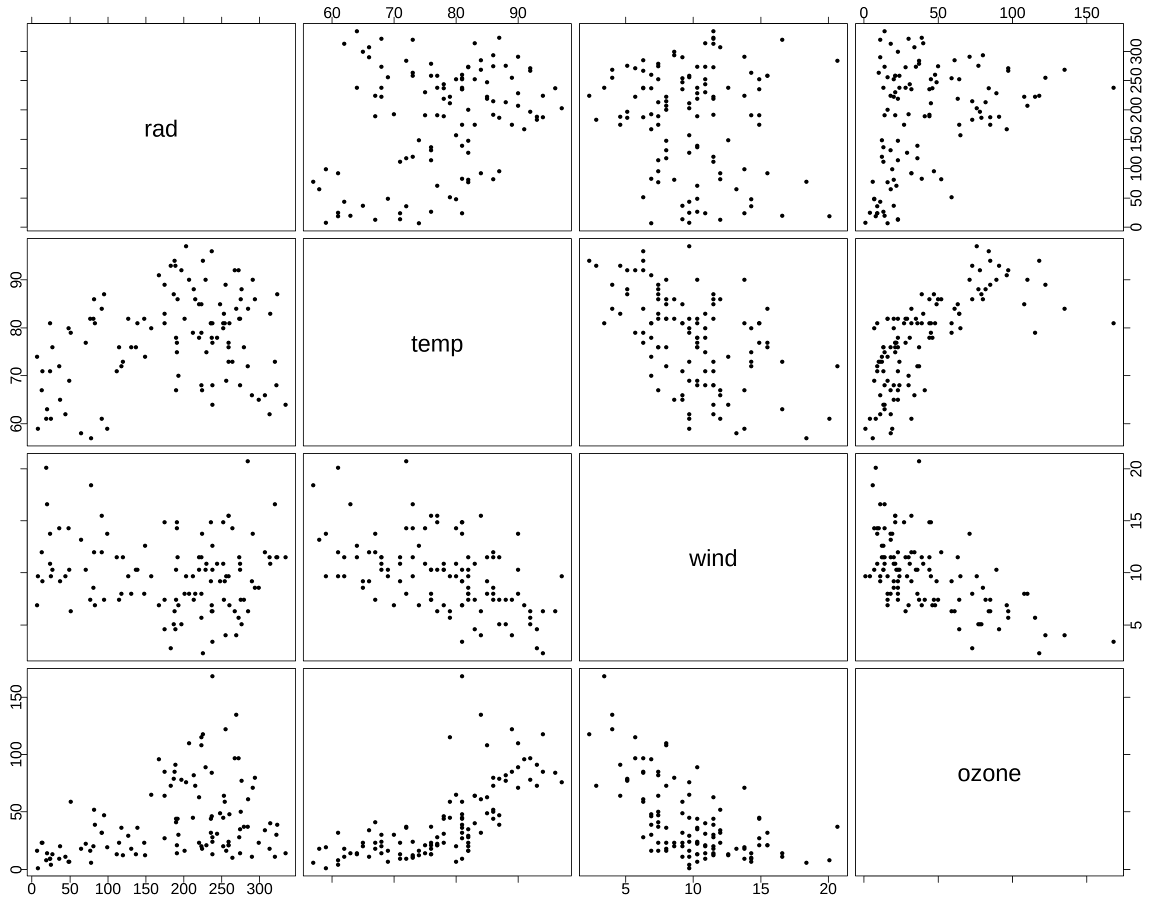
Both have $n = 111$ observations, as expected. The names in `ozone.data` are shortened to "rad" "temp" "wind" "ozone".

```
402 > attach(ozone.data)
402 > ozone.data[1:4,]

  rad temp wind ozone
1 190   67  7.4    41
2 118   72  8.0    36
3 149   74 12.6    12
4 313   62 11.5    18
[etc]
```

(This is actually a famous data set for examples in linear regression. Sometimes when this data is analyzed, the ozone values 41, 36, 12, 18, ..., are replaced with their cube roots, 3.448217, 3.301927, 2.289428, 2.620741, ...; the reason for doing this will become clearer later.)

Let's begin by looking at scatter plots of all the variable pairs.

```
pairs(ozone.data)
```

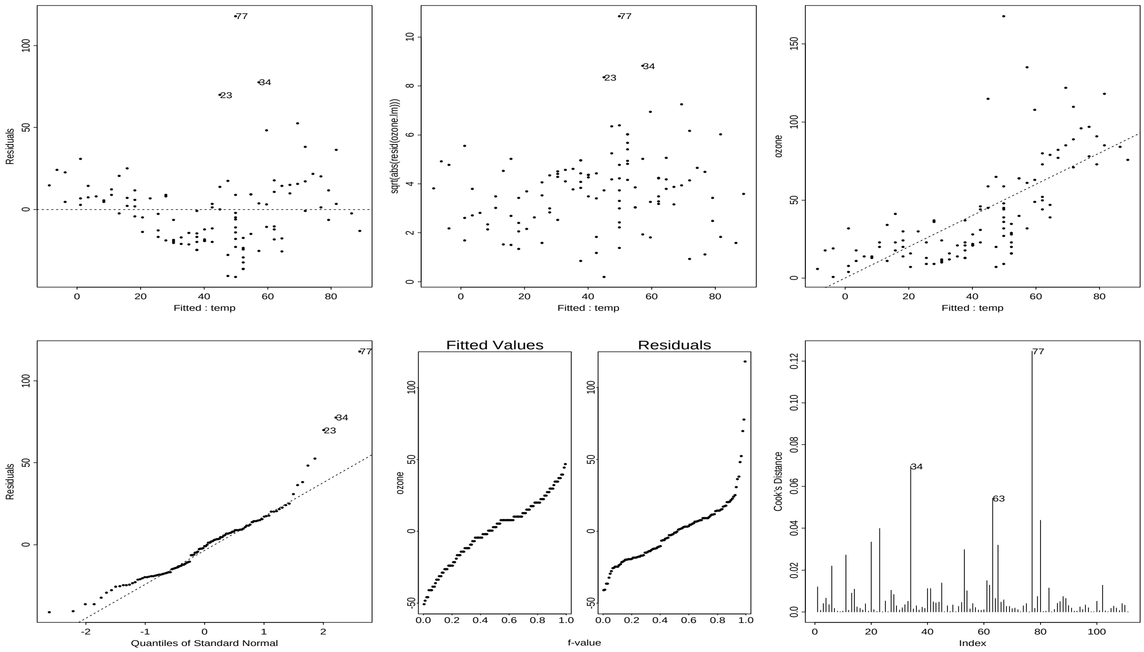


Looking at the bottom row of these plots, it appears that the most linear relationship is with temperature. So we can try that regression first

```
402 > ozone.lm _ lm(ozone ~ temp, data = ozone.data)
402 > summary(ozone.lm)
Call: lm(formula = ozone ~ temp)
Residuals:
    Min     1Q  Median    3Q   Max
 -40.92 -17.46 -0.8738 10.44 118.1

Coefficients:
                Value Std. Error   t value  Pr(>|t|)
(Intercept) -147.6461   18.7553    -7.8723    0.0000
       temp    2.4391    0.2393    10.1919    0.0000

Residual standard error: 23.92 on 109 degrees of freedom
Multiple R-Squared: 0.488
F-statistic: 103.9 on 1 and 109 degrees of freedom, the p-value is 0

Correlation of Coefficients:
     (Intercept)
temp -0.9926
```

Some diagnostic plots are available with the `plot` method for `lm`:

```
402 > par(mfrow=c(2,3))
402 > plot(ozone.lm)
```



One can also request these plots interactively, which is helpful when you are trying to examine each one carefully for evidence of fit or misfit.

```
402 > par(mfrow=c(1,1))
402 >  plot(ozone.lm,id.n=5,ask=T)

Make a plot selection (or 0 to exit):

1: plot: All
2: plot: Residuals vs Fitted Values
3: plot: Sqrt of abs(Residuals) vs Fitted Values
4: plot: Response vs Fitted Values
5: plot: Normal QQplot of Residuals
6: plot: r-f spread plot
7: plot: Cook's Distances
```

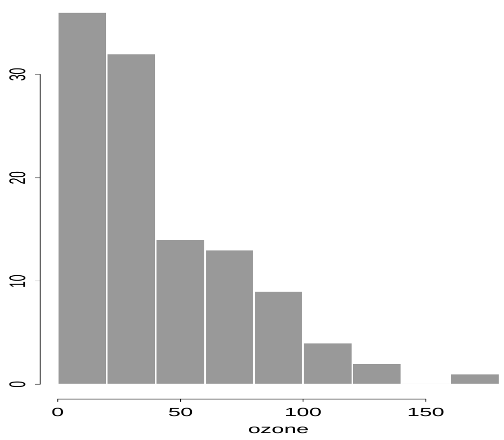
The `identify()` function can also be used to identify values in a scatter plot (give `identify()` the same data arguments as you did the `plot` command, and then click with the mouse near data points that you want to be identified in the scatter plot). This can be useful when you are comparing the Cook's distance plot to a scatter plot of fitted vs response, for example, to idenfity the high-influence or high-leverage points.

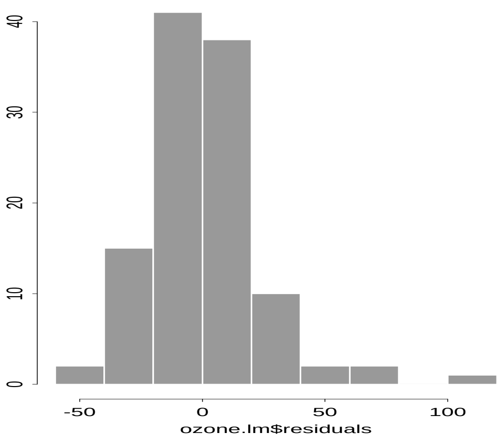
We see three potential problems with this model already:

- Outliers and Curvilinearity show up in the residual plots (and elsewhere);
- Violations of normality in the residuals;
- Residual spread is much greater than the fitted spread; this is a visual reflection of the relatively low R^2, 48%.

The five worst outliers are observations 77, 34, 23, 63, 53, so we could re-run the regression without some or all of them. Here are the commands, but I do not reproduce the output since the improvement is minor.

To address curvilinearity and non-normality, we might try a transformation of the data. First, we plot histograms of the response variable and the residuals in `ozone.lm`, to see what might work:



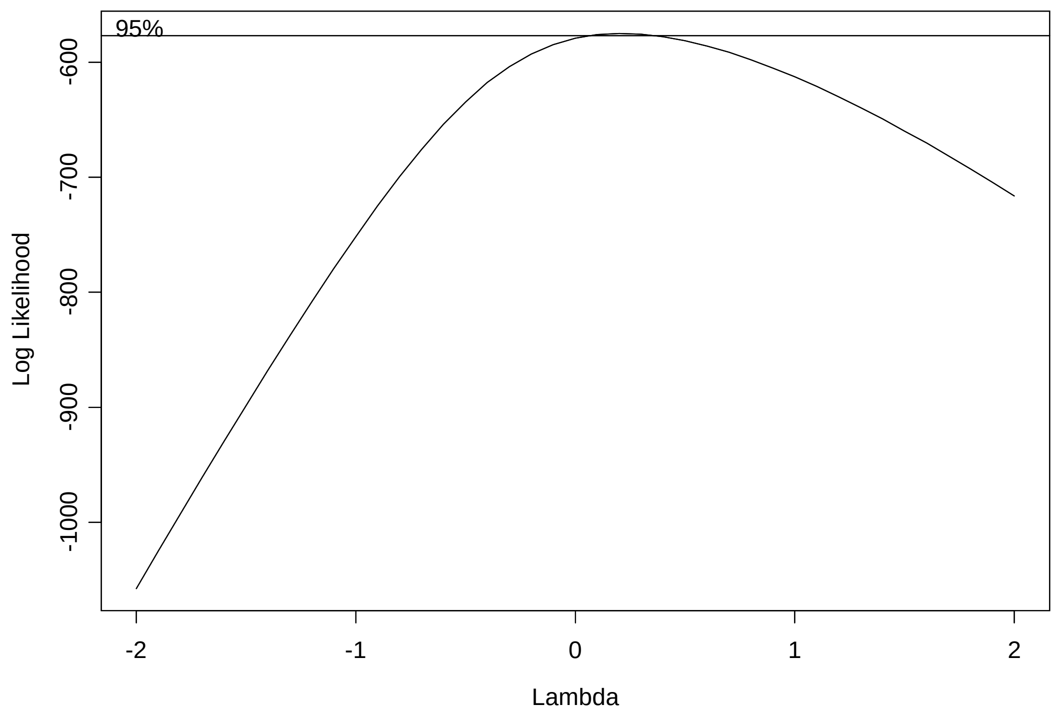


We see that `ozone` is strongly skewed right, and the residuals are mildly skewed right; this is also clear from the qq plot of residuals in the diagnostic plots above. With right-skewing we should try a fractional power or a logarithm. One can estimate the optimal power in the Box-Cox transformation,

$$y^* = (y^\lambda - 1)/\lambda \quad (\textit{or just } y^\lambda)$$

Venables and Ripley have a special library, MASS, that provides a function `boxcox`, that estimates λ directly.

```
402 > library(MASS)
402 > bozo _ boxcox(ozone ~ temp,plotit=T,data=ozone.data)
402 > rbind(bozo$x,bozo$y)

 [...]
        [,22]     [,23]     [,24]     [,25]    [,26]     [,27]     [,28]     [,29]
[1,]    0.100    0.2000    0.3000    0.4000    0.500    0.6000    0.7000    0.8000
[2,] -576.038 -574.9743 -575.6652 -577.8538 -581.325 -585.9027 -591.4433 -597.8292

 [...]
```



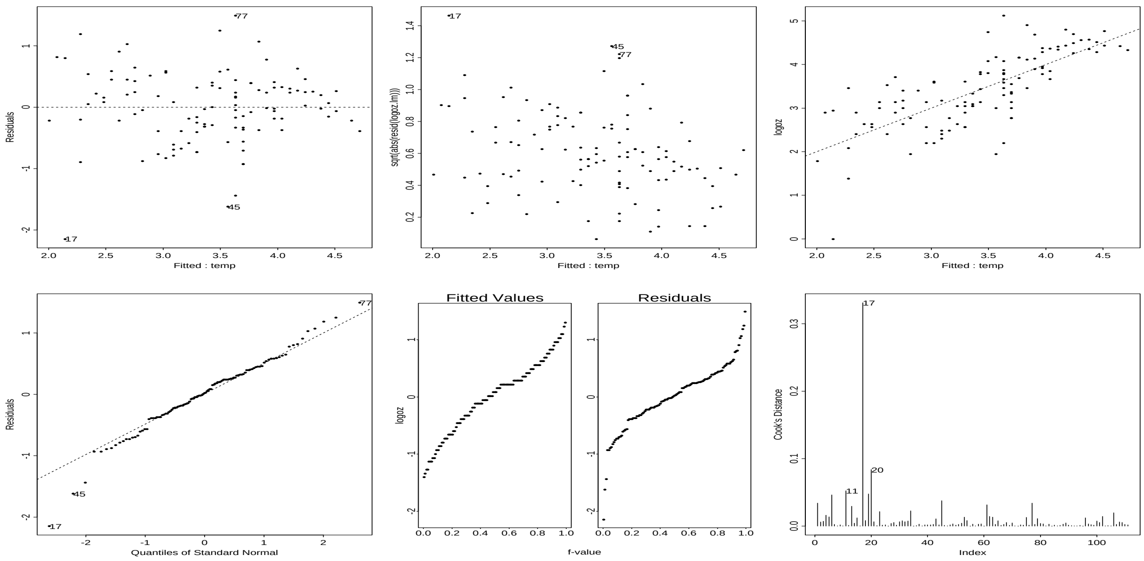
or one can guess, using nice numbers like $\lambda = 1/2, 1/3, \ldots$. Guessing is often adequate. After some guesswork, $\lambda = 1/3$ looks nice, but I also don't see much wrong and going all the way to $\lambda = 0$, i.e.

a log transformation, so that's what I will do:

$$y^* = \log(y)$$

Most aspects of the model are greatly improved:

```
402 > logoz _ log(ozone)
402 > logoz.data _ cbind(ozone.data,logoz=logoz)[1:3,5]
402 > detach()
402 > attach(logoz.data)
402 > logoz.lm _ lm(logoz ~ temp, data = logoz.data)
402 > par(mfrow=c(2,3))
402 > plot(logoz.lm)
402 > summary(logoz.lm)
Call: lm(formula = logoz ~ temp)
Residuals:
    Min      1Q  Median     3Q   Max
 -2.144 -0.3255 0.02066 0.3423 1.491

Coefficients:
               Value Std. Error  t value Pr(>|t|)
(Intercept)  -1.8485   0.4551    -4.0620   0.0001
       temp   0.0677   0.0058    11.6539   0.0000

Residual standard error: 0.5804 on 109 degrees of freedom
Multiple R-Squared: 0.5548
F-statistic: 135.8 on 1 and 109 degrees of freedom, the p-value is 0

Correlation of Coefficients:
     (Intercept)
temp -0.9926
```



2 Multiple regression

Now using the `logoz` variable, we briefly review some multiple regression ideas.

We begin by looking at the scatterplot matrix.

```
402 > pairs(logoz.data)
```

(the plot is omitted from the handout)

Now, let's make the `mlogoz.lm` model so that all the predictors are there.

```
402 > mlogoz.lm _ lm(logoz ~ rad + temp + wind, data=logoz.data)
402 > summary(mlogoz.lm)
Call: lm(formula = logoz ~ rad + temp + wind)
Residuals:
    Min      1Q    Median     3Q   Max
 -2.062 -0.2997 -0.002233 0.3077 1.236

Coefficients:
              Value Std. Error t value Pr(>|t|)
(Intercept) -0.2612  0.5534    -0.4719  0.6379
        rad  0.0025  0.0006     4.5176  0.0000
       temp  0.0492  0.0061     8.0776  0.0000
       wind -0.0616  0.0157    -3.9222  0.0002

Residual standard error: 0.5085 on 107 degrees of freedom
Multiple R-Squared: 0.6645
F-statistic: 70.65 on 3 and 107 degrees of freedom, the p-value is 0

Correlation of Coefficients:
     (Intercept)     rad    temp
 rad  0.0499
temp -0.9425     -0.2681
wind -0.6927     -0.0227  0.4849

402 > anova(mlogoz.lm)
Analysis of Variance Table

Response: logoz

Terms added sequentially (first to last)
           Df Sum of Sq  Mean Sq  F Value        Pr(F)
      rad   1  17.15661 17.15661  66.3508 0.0000000000
     temp   1  33.66815 33.66815 130.2069 0.0000000000
     wind   1   3.97777  3.97777  15.3835 0.0001552777
Residuals 107  27.66744  0.25857
```

We can think about dropping a term at a time

```
402 > drop1(mlogoz.lm)
Single term deletions

Model:
logoz ~ rad + temp + wind
       Df Sum of Sq      RSS       Cp
<none>              27.66744 29.73603
   rad  1   5.27708 32.94452 34.49596
  temp  1  16.87145 44.53889 46.09033
  wind  1   3.97777 31.64520 33.19665
```

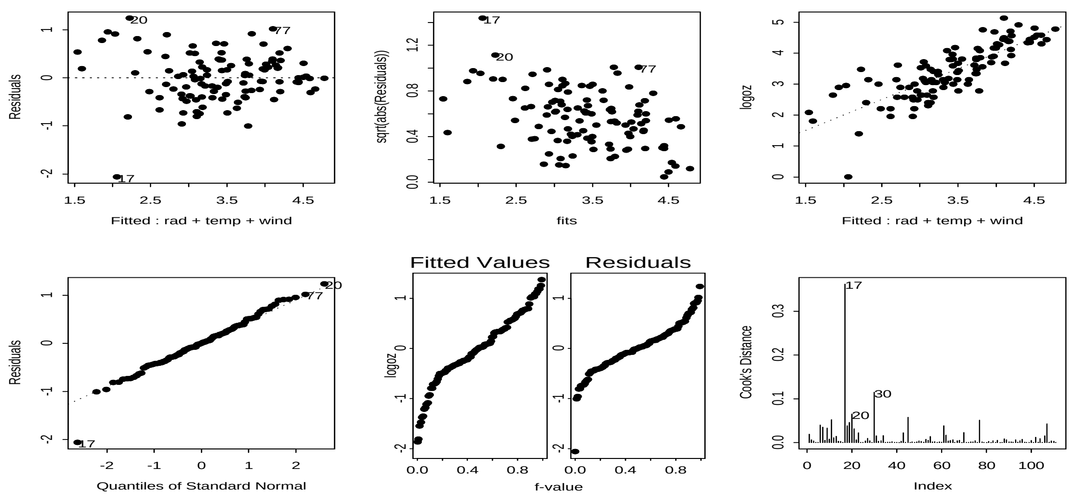
There is also a `add1` command to see what adding a term at a time does to a model; and a `update` command that can be used to add terms to a model that has already been computed.

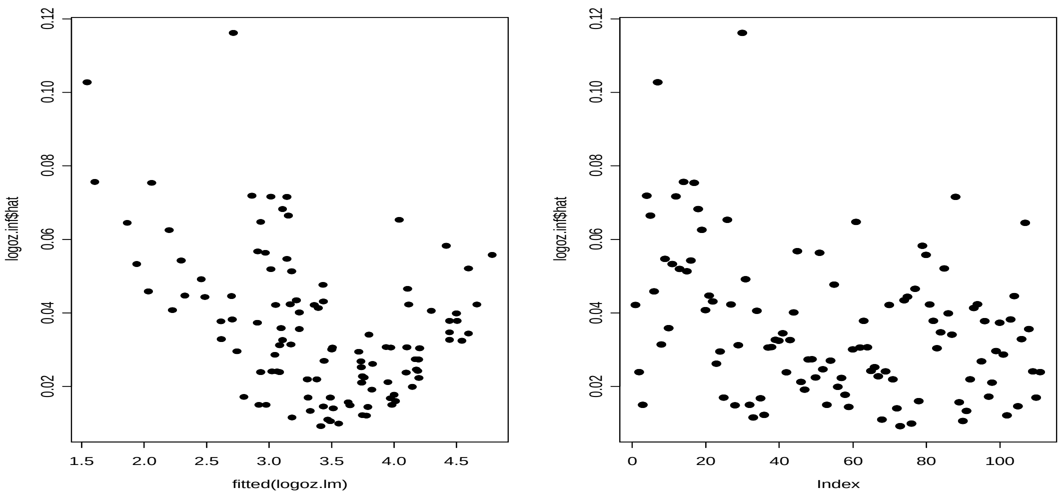
And of course one can try stepwise regression to find the "best" model, by considering a sequence of single-term additions and deletions to the model, in terms of C_p or something similar.

```
402 > step(logoz0.lm, ~ rad*temp*wind)

                      [some deleted output here]

Single term additions

Model:
logoz ~ temp + rad + wind

scale:  0.749727

          Df Sum of Sq      RSS       Cp
   <none>               27.66744 33.66525
 rad:temp  1  0.214866 27.45257 34.94984
 rad:wind  1  0.589388 27.07805 34.57532
temp:wind  1  1.194155 26.47328 33.97055
Call:
lm(formula = logoz ~ temp + rad + wind, data = logoz.data)

Coefficients:
 (Intercept)       temp         rad       wind
  -0.2611739 0.04916304 0.002514709 -0.0615925

Degrees of freedom: 111 total; 107 residual
Residual standard error (on weighted scale): 0.5085019
```

Note that this is the `mlogoz.lm` model that we started with. For this model we also consider the standard diagnostic plots, and a plot of hat matrix diagonals vs. fitted values.

```
402 > par(mfrow=c(2,3))
402 > plot(mlogoz.lm)
402 > mlogoz.inf _ lm.influence(mlogoz.lm)
402 > names(mlogoz.inf)
[1] "coefficients" "sigma"        "hat"
402 > help(lm.influence)
402 > par(mfrow=c(1,2))
402 > plot(fitted(mlogoz.lm),mlogoz.inf$hat)
402 > plot(mlogoz.inf$hat)
```

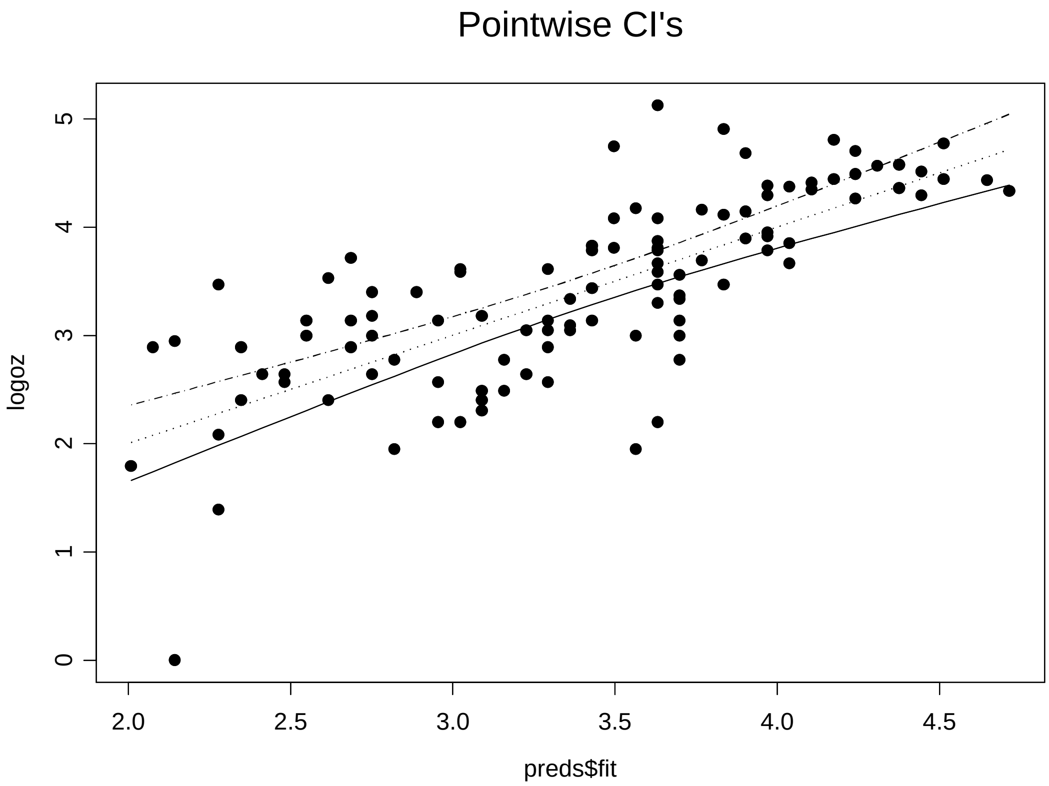




3 Prediction and confidence intervals

The main tool for making predictions in SPLUS is the `predict()` function.

```
402 > newtemp <- c(60, 62, 64, 66, 68, 70, 72)
402 > predict(logoz.lm, data.frame(temp=newtemp))
        1        2        3        4        5        6        7
 2.211842 2.347187 2.482532 2.617878 2.753223 2.888568 3.023914
402 > par(mfrow=c(1,1))
402 > preds _ pointwise(predict(logoz.lm,se.fit=T))
402 > plot(x=preds$fit,y=logoz)
402 > ord _ order(preds$fit)
402 > matlines(x=preds$fit[ord],y=cbind(preds$lower[ord],preds$fit[ord],preds$upper[ord]))
402 > title("Pointwise CI's")
```



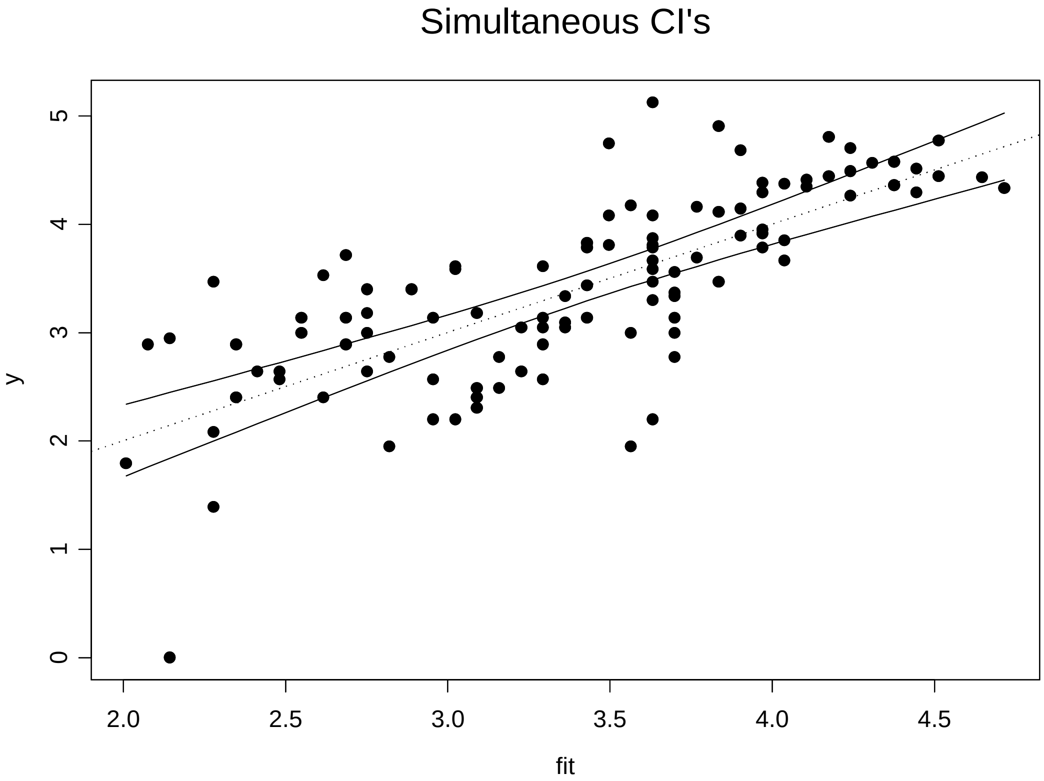
Even though each individual interval—which is based on the usual t-distriubtion calculations—has 95intervals no longer has simultaneous confidence 95we need to replace the usual t calculation,

$$\hat{y} - t(\alpha, n-2) \cdot (se.fit) \leq \hat{y} + t(\alpha, n-2) \cdot (se.fit)$$

with an F-distribution calculation. Here is a function that does this, and graphs the results. Here is a function from the SPLUS Guide to Statistics, that does the trick:

```
"confint.lm"<-function(object, alpha = 0.05, plot.it = T, ...)
{
   f <- predict(object, se.fit = T)
   p <- length(coef(object))
```

```
   fit <- f$fit
   adjust <- (p * qf(1 - alpha, p, length(fit) - p))^0.5 * f$se.fit
   lower <- fit - adjust
   upper <- fit + adjust
   if(plot.it) {
      y <- fit + resid(object)
      plot(fit, y)
      abline(0, 1, lty = 2)
      ord <- order(fit)
      lines(fit[ord], lower[ord])
      lines(fit[ord], upper[ord])
      invisible(list(lower = lower, upper = upper))
      }
   else list(lower = lower, upper = upper)
}

402 > confint.lm(logoz.lm)
402 > title("Simultaneous CI's")
```



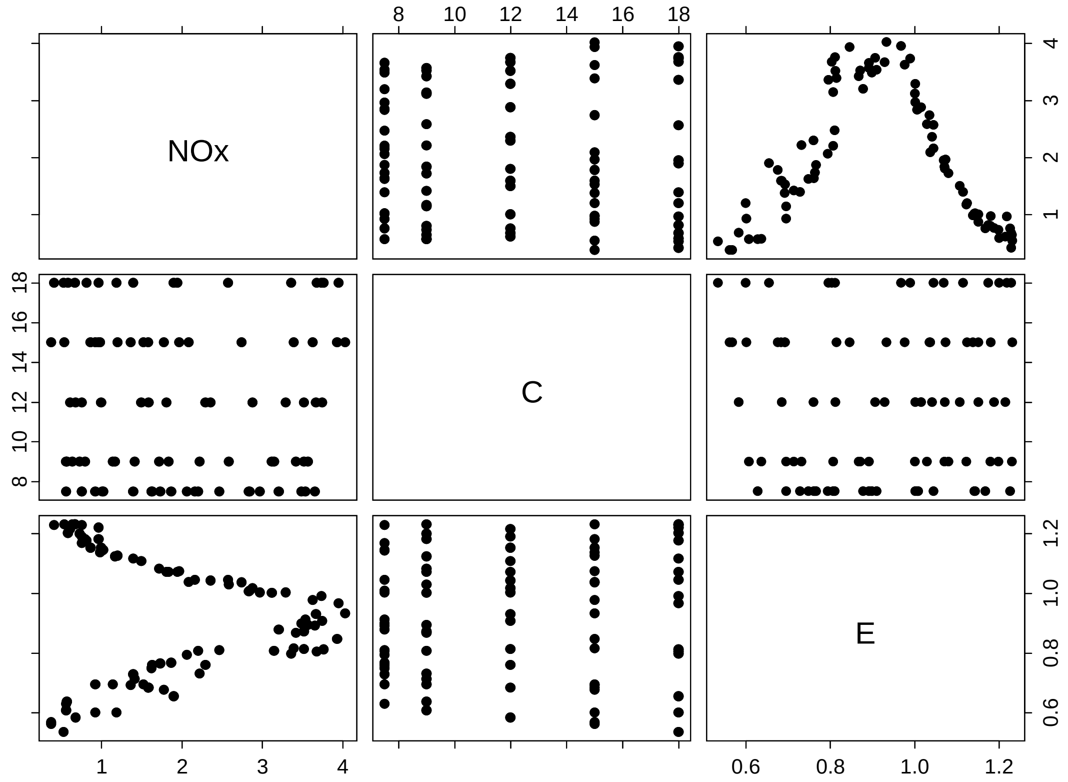
4 Polynomial Regression and Smoothing

4.1 Polynomial Regression

We will start with a new data set, `ethanol`, and a new question, what is the relationshop between E, a measure of how rich the engine was running, and NOx, concentration of NO and NO2 in the exhaust.

```
ethanol    Measurement of Exhaust from Burning Ethanol

SUMMARY

The ethanol data frame records 88 measurements (rows) for three
variables (columns) NOx, C, and E from an experiment in which ethanol
was burned in a single cylinder automobile test engine.

The gas data frame records 22 measurements of NOx and E from a similar
experiment involving gasoline in place of ethanol.


DATA DESCRIPTION

        NOx the concentration of nitric oxide (NO) and nitrogen
            dioxide (NO2) in engine exhaust, normalized by the work done
            by the engine.
        C  the compression ratio of the engine.
        E  the equivalence ratio at which the engine was run---a
           measure of the richness of the air/ethanol mix.


SOURCE

Brinkman, N.D. (1981) Ethanol Fuel---A single-cylinder Engine Study of
Efficiency and Exhaust Emissions.  SAE transactions Vol. 90, No
810345, pp. 1410--1424.

John M. Chambers and Trevor J. Hastie, (eds.)  Statistical Models in
S, Wadsworth and Brooks, Pacific Grove, CA 1992, pp. 75, 222,
272--280, 331.


EXAMPLES

eth1 <- gam(NOx ~ C + lo(E, degree = 2), data = ethanol)
```

First we look at a scatterplot matrix with `pairs(ethanol)`:



Clearly there are strange things going on:

- The two variables we care about, E and NOx, have a very nonlinear relationship;
- Apparently only a few distinct compression ratios were tried.

For now we will concentrate on E vs NOx; next week we will come back to the discreteness of C.

A first thing to try with data like this is *polynomial regression.* This is still a kind of linear regression because the model looks like

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \varepsilon$$

so that the model is still "linear in the beta's", which is the important part about "linearity" in linear models.

Let's try a 4^{th}-degree polynomial first:

```
02 > eth4.lm _ lm(NOx ~ poly(E,4))
402 > summary(eth4.lm)

Call: lm(formula = NOx ~ poly(E, 4))

Residuals:
     Min      1Q   Median     3Q    Max
```

```
 -0.8125 -0.1445 -0.02927 0.1607 1.017

Coefficients:
               Value Std. Error  t value Pr(>|t|)
(Intercept)   1.9574   0.0393    49.8407   0.0000
poly(E, 4)1  -1.0747   0.3684    -2.9170   0.0045
poly(E, 4)2  -9.2606   0.3684   -25.1367   0.0000
poly(E, 4)3  -0.4879   0.3684    -1.3243   0.1890
poly(E, 4)4   3.6341   0.3684     9.8644   0.0000

Residual standard error: 0.3684 on 83 degrees of freedom
Multiple R-Squared: 0.8991
F-statistic: 184.9 on 4 and 83 degrees of freedom, the p-value is 0

Correlation of Coefficients:
            (Intercept) poly(E, 4)1 poly(E, 4)2 poly(E, 4)3
poly(E, 4)1 0
poly(E, 4)2 0           0
poly(E, 4)3 0           0           0
poly(E, 4)4 0           0           0           0
```

The coefficients listed above aren't very meaningful. They are computed for a set of orthogonal polynomials, instead of the simpler powers of E (or x, or whatever). The function `poly.transform` can get the inpterpretable coefficients back for you.

```
402 > poly.transform(poly(E,4),eth4.lm$coeff)
      x^0       x^1      x^2       x^3     x^4
 174.3601 -872.2071 1576.735 -1211.219 335.356
```

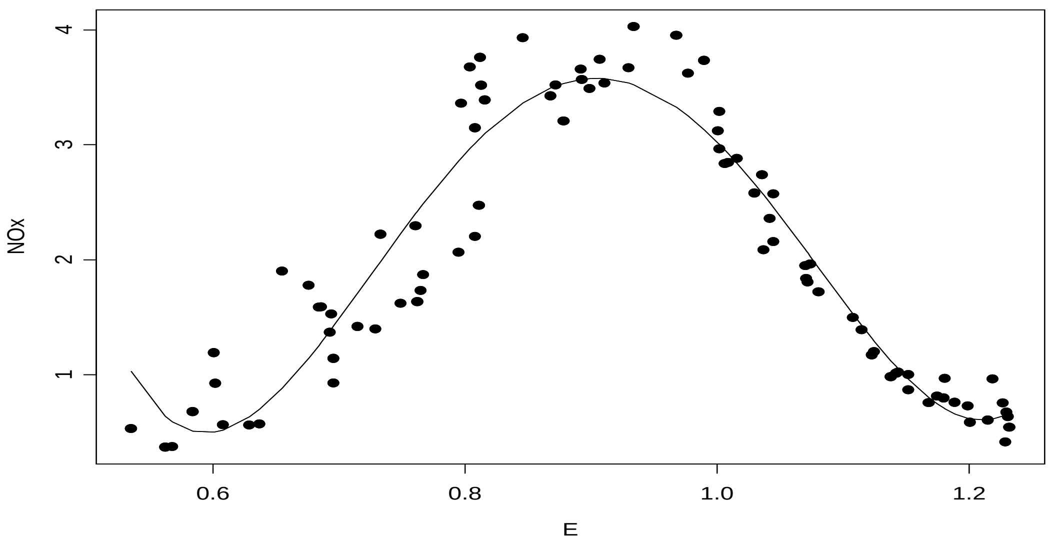
The actual polynomials used are as follows: Define $P(-1) = 0$, and $P(0) = 1$. Then $P(j+1)$ is proportional to $(x - \texttt{alpha}[j+1]) * P(j) - \texttt{norm2}[j+2]/\texttt{norm2}[\texttt{j}+1] * \texttt{P}(\texttt{j}-1)$, where `alpha` and `norm2` are fixed coefficients. These polynomials are orthogonal in the function space with inner product $\langle f, g \rangle = \int f \cdot g$, and the constants of proportionality are chosen so that they have unit length in that space. The reason for working with orthogonal polynomials instead of the more natural $1, x, x^2, x^3, \ldots$ is that the regression computations are always more stable when the predictors (columns of the X matrix) are orthogonal.

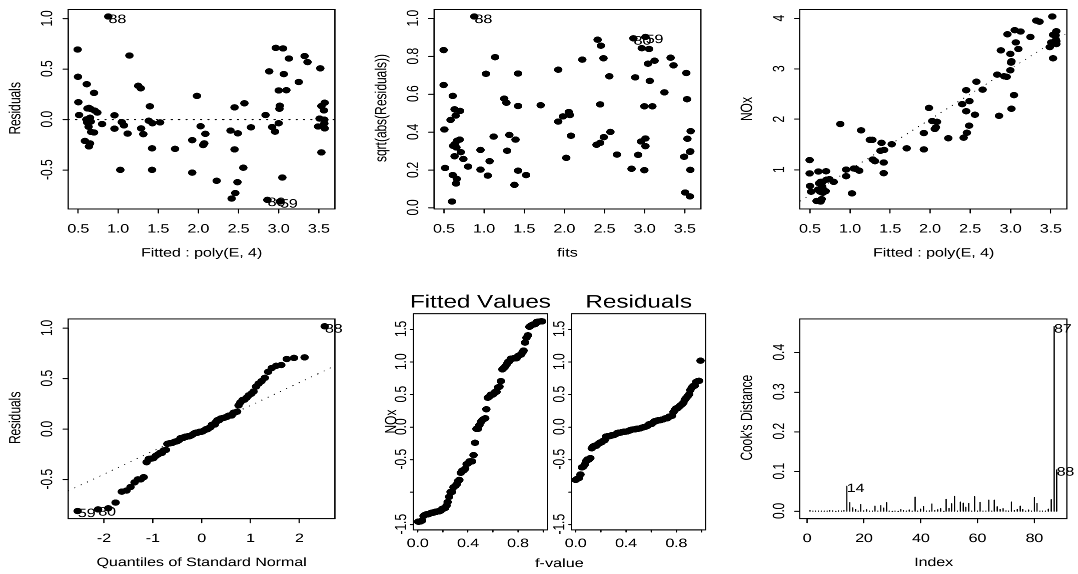
As far as I know, one would have to do one's own transformations to get standard errors, t-tests, etc. for these coefficients.

Now let's look at some diagnostic plots...

```
402 > par(mfrow=c(1,1))
402 > plot(E,NOx)
402 > plot(E,fitted(eth4.lm))
402 > plot(E,NOx)
402 > lines(E,fitted(eth4.lm))
402 > plot(E,NOx)
```

```
402 > ord _ order(E)
402 > lines(E[ord],fitted(eth4.lm)[ord])
402 > par(mfrow=c(2,3))
402 > plot(eth4.lm)
```





The fourth-degree polynomial is following the data fairly well, and the diagnostic plots show we are doing well in the regression (there is no problem using these diagnostic plots since we are still doing linear regression).

On the other hand, the residual plot and the fitted-vs-actual plot do show that some fine structure in the variation of NOx seems to be missed by the polynomial.

Now let's see what happens when we add C, so the model becomes `NOx ~ C + poly(E,4)`:

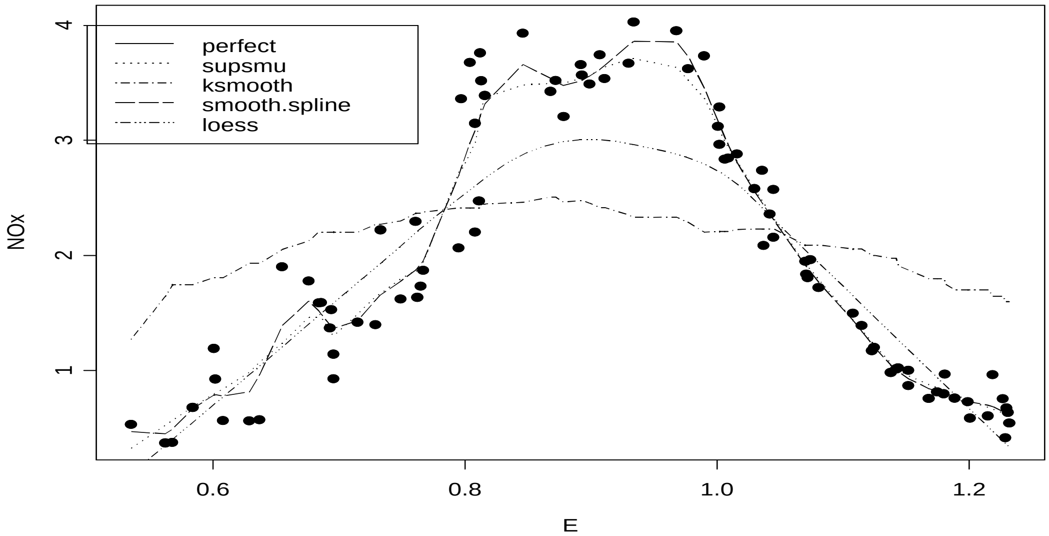
```
402 > summary(update(eth4.lm, ~ . + C))

Call: lm(formula = NOx ~ poly(E, 4) + C)
Residuals:
     Min      1Q  Median     3Q    Max
 -0.6471 -0.2196 0.07509 0.2452 0.6746

Coefficients:
               Value Std. Error  t value Pr(>|t|)
(Intercept)   1.2978   0.1100    11.7967   0.0000
poly(E, 4)1  -1.2042   0.3054    -3.9435   0.0002
poly(E, 4)2  -9.7860   0.3160   -30.9728   0.0000
poly(E, 4)3  -0.2223   0.3076    -0.7227   0.4719
poly(E, 4)4   3.4802   0.3056    11.3866   0.0000
          C   0.0548   0.0087     6.2747   0.0000

Residual standard error: 0.3047 on 82 degrees of freedom
Multiple R-Squared: 0.9318
F-statistic: 224.1 on 5 and 82 degrees of freedom, the p-value is 0

Correlation of Coefficients:
            (Intercept) poly(E, 4)1 poly(E, 4)2 poly(E, 4)3 poly(E, 4)4
poly(E, 4)1  0.0646
poly(E, 4)2  0.2532      0.0179
poly(E, 4)3 -0.1315     -0.0093     -0.0365
poly(E, 4)4  0.0767      0.0054      0.0213     -0.0110
          C -0.9554     -0.0676     -0.2650      0.1376     -0.0802
```

4.2 Smoothing in Regression

This is really a topic for another day, but since we are so close I decided to do a little of it.

In 401, you probably learned about some smooothing methods, such as "loess" or "lowess". Here are some smoothing methods, crudely compared:

```
402 > plot(E,NOx)
 lines(supsmu(E,NOx),lty=2)
 lines(ksmooth(E,NOx),lty=3)
 lines(smooth.spline(E,NOx),lty=4)
 lines(loess.smooth(E,NOx),lty=5)
 legend(0.5,4,c("perfect", "supsmu", "ksmooth",
 "smooth.spline", "loess"), lty=1:5)
```

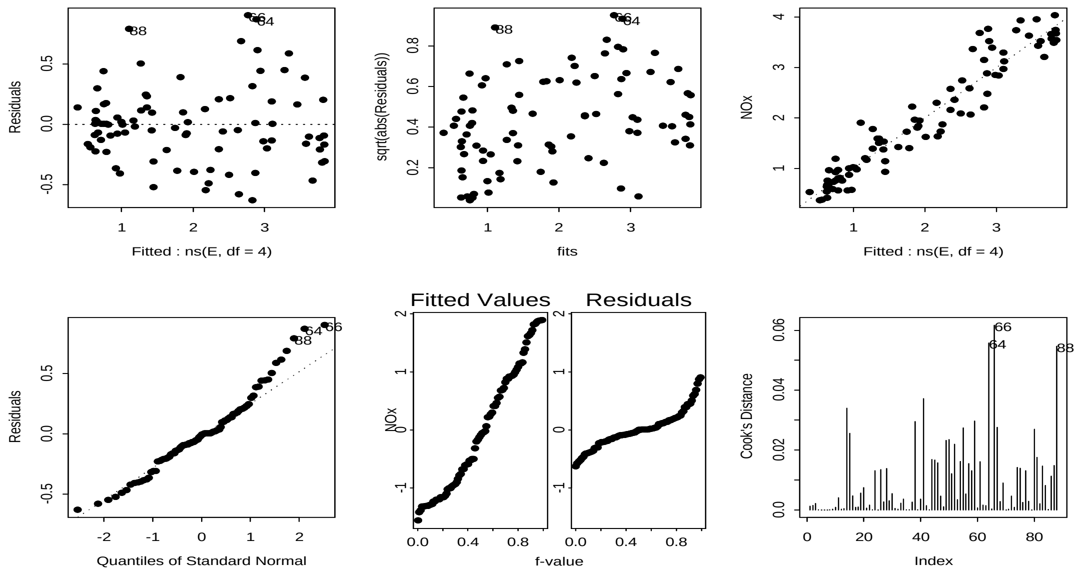


Each of these methods can be incorporated into regression techniques. This is a kind of nonlinear, nonparametric regression (about which I hope we talk more later in the semester). For right now I will illustrate some examples with `ns`.

Here are two quick examples.

```
402 > ns4.lm _ lm(NOx ~ ns(E,df=4),data=ethanol)
402 > summary(ns4)
Error: Object "ns4" not found
Dumped
402 > summary(ns4.lm)

Call: lm(formula = NOx ~ ns(E, df = 4), data = ethanol)
Residuals:
     Min      1Q   Median     3Q   Max
 -0.6305 -0.1951 -0.01297 0.1641 0.902

Coefficients:
                  Value Std. Error  t value Pr(>|t|)
   (Intercept)   0.3925   0.1610     2.4377   0.0169
ns(E, df = 4)1   4.7578   0.1927    24.6963   0.0000
ns(E, df = 4)2   0.3288   0.1881     1.7483   0.0841
ns(E, df = 4)3   1.2611   0.3937     3.2030   0.0019
ns(E, df = 4)4  -0.4121   0.1421    -2.8997   0.0048

Residual standard error: 0.3274 on 83 degrees of freedom
Multiple R-Squared: 0.9203
F-statistic: 239.6 on 4 and 83 degrees of freedom, the p-value is 0
```

```
Correlation of Coefficients:
                (Intercept) ns(E, df = 4)1 ns(E, df = 4)2 ns(E, df = 4)3
ns(E, df = 4)1 -0.5656
ns(E, df = 4)2 -0.6729      0.0683
ns(E, df = 4)3 -0.9445      0.3881         0.6719
ns(E, df = 4)4 -0.0363      0.4369        -0.2968        -0.0182

402 > par(mfrow=c(2,3))
402 > plot(ns4c.lm)
```



And the final example is similar to the one that is listed in the help file for the "ethanol" data set in the first place.

```
402 > ns4c.lm _ lm(NOx ~ ns(E, df = 4) +C,data = ethanol)
402 > summary(ns4c.lm)

Call: lm(formula = NOx ~ ns(E, df = 4) + C, data = ethanol)
Residuals:
     Min      1Q  Median     3Q    Max
 -0.5902 -0.1984 0.01569 0.1565 0.5279

Coefficients:
                  Value Std. Error  t value Pr(>|t|)
   (Intercept)  -0.4993   0.1669    -2.9910   0.0037
ns(E, df = 4)1   4.9752   0.1490    33.3886   0.0000
ns(E, df = 4)2   0.4612   0.1439     3.2048   0.0019
ns(E, df = 4)3   1.7513   0.3057     5.7291   0.0000
ns(E, df = 4)4  -0.5176   0.1089    -4.7551   0.0000
```

```
           C   0.0559   0.0071     7.8548   0.0000

Residual standard error: 0.2488 on 82 degrees of freedom
Multiple R-Squared: 0.9545
F-statistic: 344.3 on 5 and 82 degrees of freedom, the p-value is 0

Correlation of Coefficients:
               (Intercept) ns(E, df = 4)1 ns(E, df = 4)2 ns(E, df = 4)3
ns(E, df = 4)1 -0.5338
ns(E, df = 4)2 -0.5696      0.0884
ns(E, df = 4)3 -0.8166      0.4112         0.6772
ns(E, df = 4)4  0.0575      0.4031        -0.3070        -0.0429
             C -0.6801      0.1858         0.1171         0.2041
               ns(E, df = 4)4
ns(E, df = 4)1
ns(E, df = 4)2
ns(E, df = 4)3
ns(E, df = 4)4
             C -0.1233

402 > par(mfrow=c(2,3))
402 > plot(ns4c.lm)
```