4. Linear and Generalized Linear Models

4.1 Introduction

Linear models relate a response (or dependent variable) to a set of predictors (or independent variables) by a linear expression of unknown parameters which are estimated from the data, i.e. they are termed *linear* because we estimate a *linear function* of the unknown parameters. The actual response may depend in a non-linear way on the predictors, e.g. there may be a polynomial relationship but this can still be expressed as a *linear function of the parameters.*

If the response is a continuous quantitative variable then we may model the response as a Normal random variable with mean depending upon the predictors. The next section provides a brief review and illustration of regression models, including simple regression diagnostics. Ideas of *robust regression* and *bootstrapping* are covered.

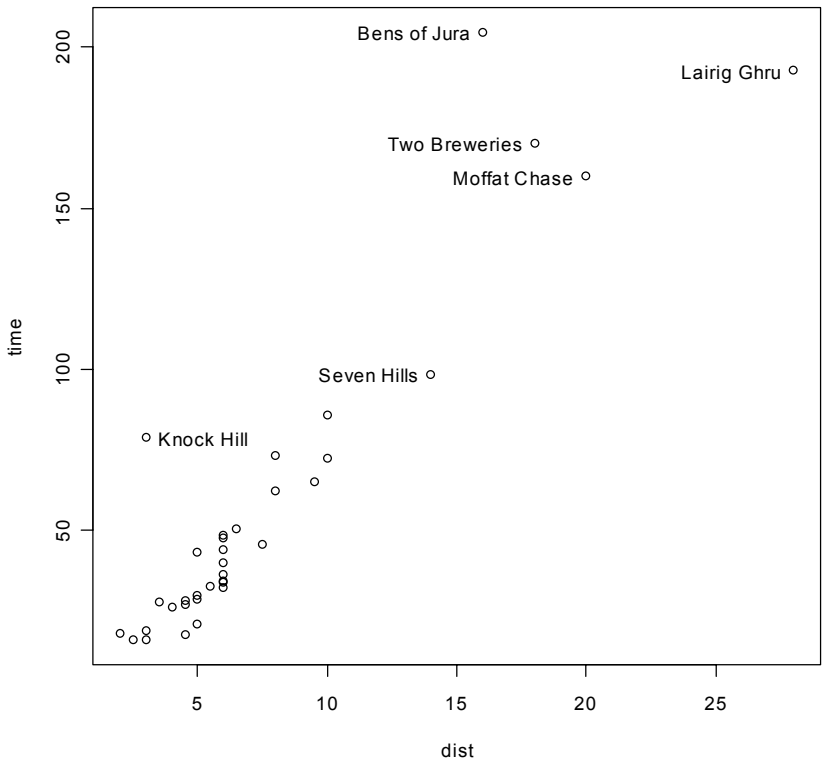
Generalized linear models cover situations where the response is some other measure, e.g. success/failure, and we may then model some other function of the response leading to techniques such as *log-linear* and *logistic regression* which are considered briefly in later sections.

All the methods will be illustrated on specific examples.

4.2 Simple Linear Regression

4.2.1 Example: Scottish Hill Races

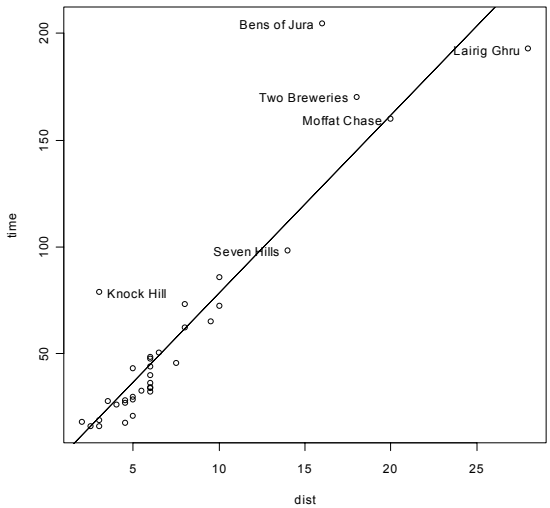
```
> library(MASS)
> data(hills)
> attach(hills)
> plot(dist,time)
> identify(dist,time,row.names(hills))
 [1]  7 11 17 18 33 35
```



Note the use of `identify()` which allowed several of the points to be named with the row names just by pointing at them with the mouse and clicking the left button. Clicking with the right button and choosing stop returns control to the Console Window. It is clear that there are some outliers in the data, notably Knock Hill and Bens of Jura which may cause some trouble.

Next, we fit a simple linear model to relate time to distance, storing the analysis in object `hillslm1`, and add the fitted line to the plot:

```
> hillslm1<- lm(time~dist)
> summary(hillslm1)

Call:
lm(formula = time ~ dist)

Residuals:
    Min      1Q  Median      3Q     Max
-35.745  -9.037  -4.201   2.849  76.170

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -4.8407     5.7562  -0.841    0.406
dist          8.3305     0.6196  13.446    6e-15 ***
---
Signif. codes:  0  `***'  0.001  `**'  0.01  `*'  0.05  `.'  0.1  ` '  1

Residual standard error: 19.96 on 33 degrees of freedom
Multiple R-Squared: 0.8456,     Adjusted R-squared: 0.841
F-statistic: 180.8 on 1 and 33 degrees of freedom,       p-value:
6.106e-015

> lines(abline(hillslm1))
>
```



Although this shows a highly significant result for the slope we need to investigate the adequacy of the model further.

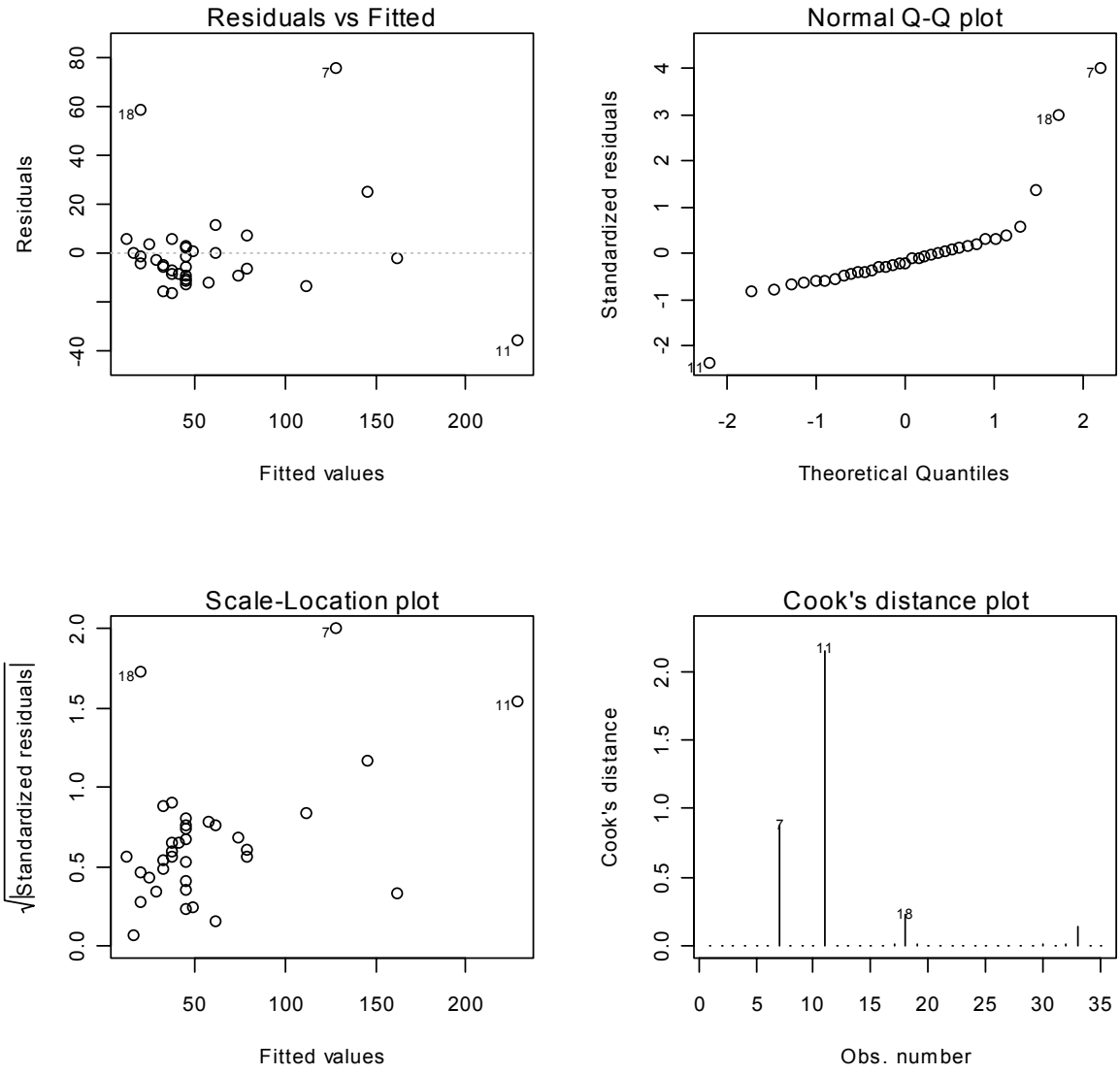
4.2.2 Regression Diagnostics:

The model that we have used is that if the time and distance of the i^{th} race are y_i and x_i respectively then our model is that $y_i=\alpha+\beta\ x_i+\varepsilon_i$ where the ε_I are independent observations from $N(0,\sigma^2)$. If we look at the ***residuals*** $\hat{\varepsilon}_i$ given by $\hat{\varepsilon}_i = y_i - \hat{y}_i = y_i - \hat{\alpha} - \hat{\beta}x_i$ then these residuals should look as if they are independent observations from $N(0,\sigma^2)$, and further they should be independent of the fitted values $\hat{y}_i$.

A further question of interest is whether any of the observations are ***influential***, that is, do any of the observations greatly affect the estimates of α and β. The way to check this is to leave each observation out of the data in turn and estimate the parameters from the reduced data set. ***Cooks Distance*** is a measure of how much the estimate changes as each observation is dropped.

All of these ***diagnostics*** can be performed graphically using the function `plot.lm()` which take as its argument the results of the `lm()` analysis (which was stored as an object `hillslm1`).

```
> par(mfrow=c(2,2))
> plot.lm(hillslm1)
> hills
> row.names(hills)
 [1] "Greenmantle"   "Carnethy"         "Craig Dunain"   "Ben Rha"
 [5] "Ben Lomond"    "Goatfell"         "Bens of Jura"   "Cairnpapple"
 [9] "Scolty"        "Traprain"         "Lairig Ghru"    "Dollar"
[13] "Lomonds"       "Cairn Table"      "Eildon Two"     "Cairngorm"
[17] "Seven Hills"   "Knock Hill"       "Black Hill"     "Creag Beag"
[21] "Kildcon Hill"  "Meall Ant-Suidhe" "Half Ben Nevis" "Cow Hill"
[25] "N Berwick Law" "Creag Dubh"       "Burnswark"      "Largo Law"
[29] "Criffel"       "Acmony"           "Ben Nevis"      "Knockfarrel"
[33] "Two Breweries" "Cockleroi"        "Moffat Chase"
```

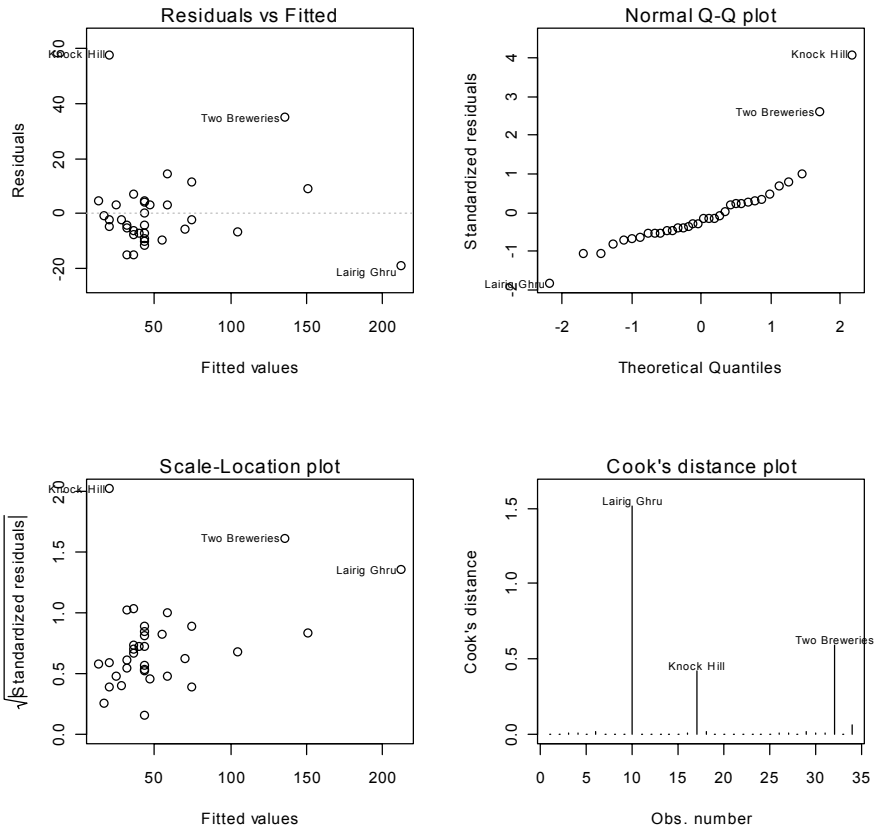
The function automatically identifies (with row numbers) the outlying and most influential points.

```
7: Bens of Jura, 11: Lairig Ghru, 18: Knock Hill
```

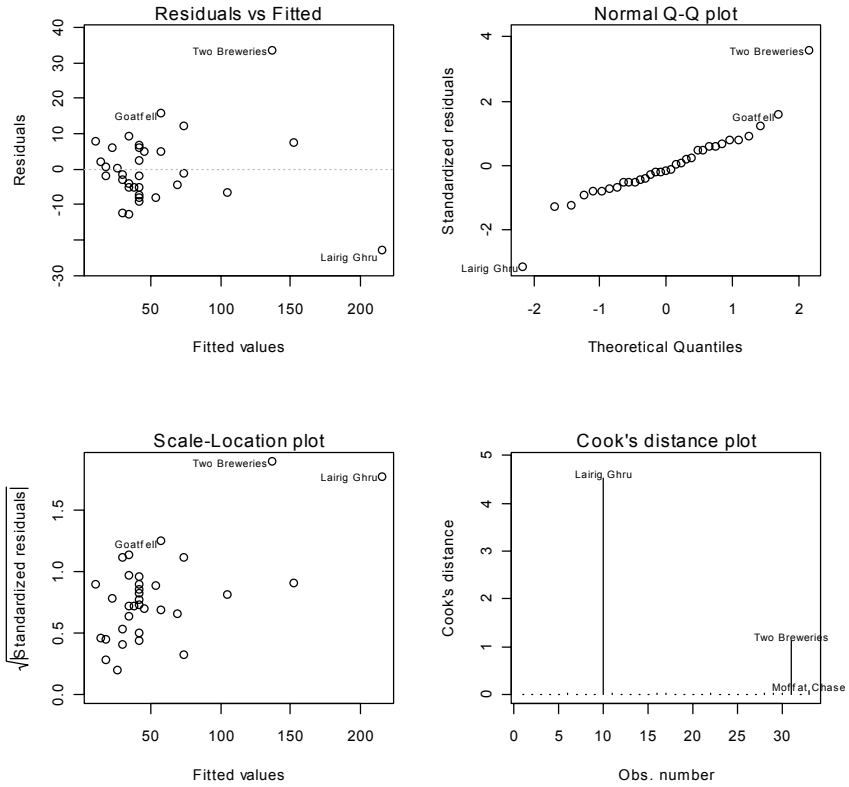
Of these, 7, Bens of Jura, is the most serious — it is both an outlier and is influential so the estimates depend strongly on it. 18, Knock Hill, is also an outlier but not nearly so influential so the results will not change so much if that observation is removed. 11, Lairig Ghru is very influential but does not appear to be 'out of line' with the others, it is just the longest race.

We can investigate the effect of dropping individual observations from the data set by `lm(time~dist,data=hills[-7,])` and `lm(time~dist,data=hills[-c(7,18),])`:

```
> hillslm2<- lm(time~dist,data=hills[-7,])
> summary(hillslm2)
Call:
lm(formula = time ~ dist, data = hills[-7, ])
Residuals:
    Min      1Q  Median      3Q     Max
-19.221  -7.412  -3.159   3.692  57.790
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -2.0630     4.2073   -0.49    0.627
dist          7.6411     0.4665   16.38   <2e-16 ***

Residual standard error: 14.48 on 32 degrees of freedom
Multiple R-Squared: 0.8934,     Adjusted R-squared: 0.8901
F-statistic: 268.3 on 1 and 32 degrees of freedom,       p-value:
0
> plot.lm(hillslm2)
```



```
> hillslm3<- lm(time~dist,data=hills[-c(7,18),])
> summary(hillslm3)
Call:
lm(formula = time ~ dist, data = hills[-c(7, 18), ])
Residuals:
    Min      1Q  Median      3Q     Max
-23.023  -5.285  -1.686   5.981  33.668
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -5.8125     3.0217  -1.924   0.0636 .
dist          7.9108     0.3306  23.926   <2e-16 ***

Residual standard error: 10.16 on 31 degrees of freedom
Multiple R-Squared: 0.9486,     Adjusted R-squared: 0.947
F-statistic: 572.5 on 1 and 31 degrees of freedom,       p-value:
0
> plot.lm(hillslm3)
```

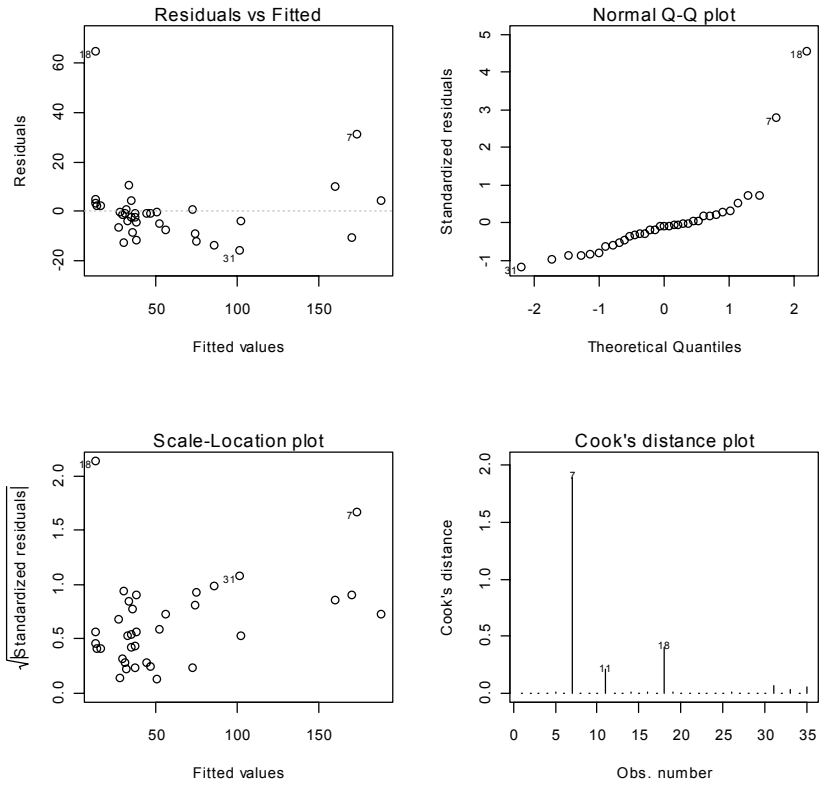


4.2.3 Several Variables

We can fit regression models involving several variables just by extending the formula in the `lm()` function in a natural way and we still have the same available diagnostics. To include the variable climb we have:

```
> hillslm4<-lm(time~dist+climb)
> summary(hillslm4)
Call:
lm(formula = time ~ dist + climb)

Residuals:
    Min      1Q  Median      3Q     Max
-16.215  -7.129  -1.186   2.371  65.121
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -8.992039   4.302734  -2.090   0.0447 *
dist         6.217956   0.601148  10.343 9.86e-12 ***
climb        0.011048   0.002051   5.387 6.45e-06 ***
Residual standard error: 14.68 on 32 degrees of freedom
Multiple R-Squared: 0.9191,     Adjusted R-squared: 0.914
F-statistic: 181.7 on 2 and 32 degrees of freedom,       p-value:
0
```


```
> plot.lm(hillslm4)
```



Note that the 11th observation is no longer the most influential one but we still have problems with outliers. These could be dropped in just the same way as previously.

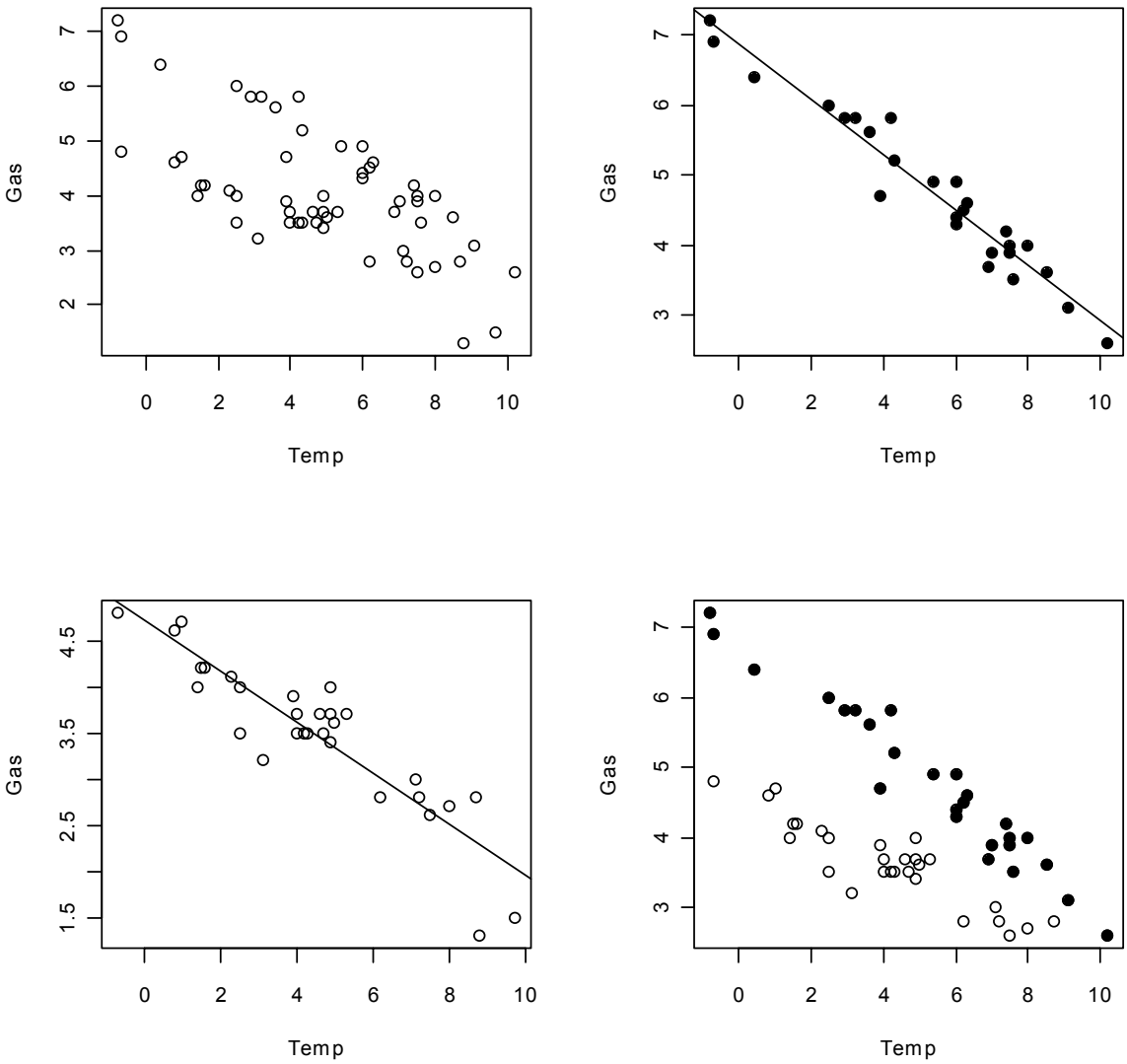
4.2.3 Extensions

To consider slightly more complex models involving factors, we consider another data set `whiteside` which consists of three variables, `Gas`, `Temp` and `Insul`. These give the gas consumption and outside temperature for 26 weeks before and 30 weeks after installing insulation, recorded as `Before` and `After` in the two-level factor `Insul`. If we plot the data we can see that there are two distinct groups and that these coincide with the 'Before' and 'After' classifications.

```
> data(whiteside)
> attach(whiteside)
> plot(Gas,Temp)
> plot(Gas[Insul==”Before”],Temp[Insul==”Before”],pch=19)
> lines(abline(lm(Gas~Temp,subset=Insul=="Before")))
> plot(Gas[Insul=="After"],Temp[Insul=="After"])
> lines(abline(lm(Gas~Temp,subset=Insul=="After")))
> plot(Gas[Insul==”Before”],Temp[Insul==”Before”],pch=19)
> points(Gas[Insul==”After”],Temp[Insul==”After”])
```

The above code produces the plots on the next page but the pictures would benefit from tidying-up and enhancing with labels and legends.

Note the use of `subset` to select a portion of the data when fitting the linear models (in `abline`) and the use of `Gas[Insul==”Before”]` etc to plot some of the points with different symbols. We could instead use `subset` in the plot command.



We now fit models separately to the ‘Before’ and ‘After’ subsets of the data and examine them.

```
> gasB<-lm(Gas~Temp,subset=Insul=="Before")
> gasA<-lm(Gas~Temp,subset=Insul=="After")
> print(gasB)

Call:
lm(formula = Gas ~ Temp, subset = Insul == "Before")

Coefficients:
(Intercept)         Temp
     6.8538      -0.3932
> print(gasA)

Call:
lm(formula = Gas ~ Temp, subset = Insul == "After")

Coefficients:
(Intercept)         Temp
     4.7238      -0.2779
```

Note the use of `print()` for a very brief summary of the results.

We could fit the two lines simultaneously using the model formula `Gas~Insul/Temp-1`

```
> gasBA<-lm(Gas~Insul/Temp-1)
> print(gasBA)

Call:
lm(formula = Gas ~ Insul/Temp - 1)

Coefficients:
InsulBefor   InsulAfter  InsulBefore:Temp   InsulAfter:Temp
   6.8538      4.7238           -0.3932           -0.2779
```

This has fitted the model

$Gas_{ik}=\alpha_k+\beta_k Temp_i+\varepsilon_{ik}$ where k=1 for 'Before' and k=2 for 'After'

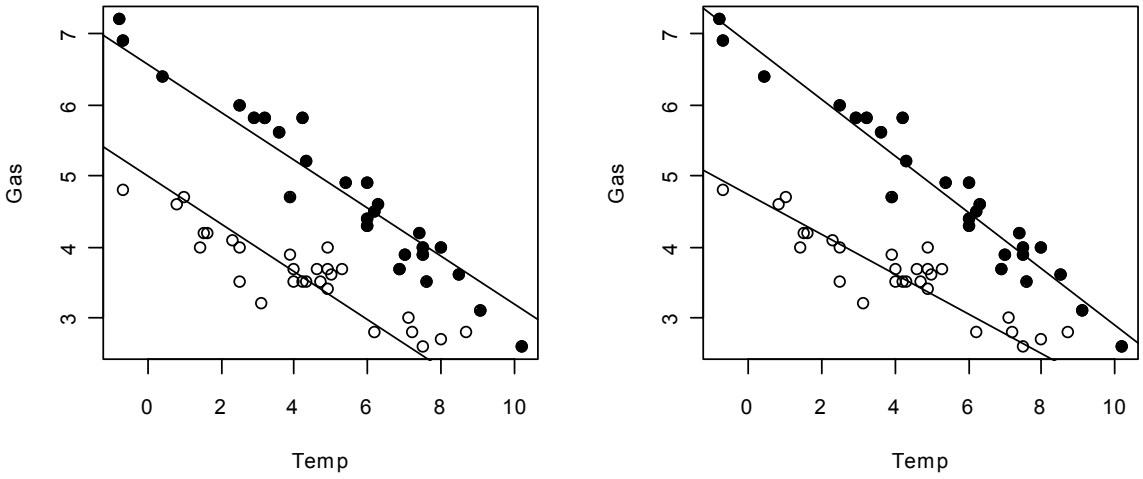
So α_1=`InsulBefor`, α_2=`InsulAfter`,β_1=`InsulBefore:Temp` and β_2=`InsulAfter:Temp`

To fit a model with a common slope, i.e. two parallel lines:

```
> gasPAR<-lm((Gas~Insul+Temp-1))
> print(gasPAR)

Call:
lm(formula = (Gas ~ Insul + Temp - 1))

Coefficients:
InsulBefore   InsulAfter         Temp
     6.5513       4.9861      -0.3367
```

We can plot the various fitted lines on the scatter plots:

```
> plot(Gas~Temp,pch=19,subset=Insul=="Before")
> points(Gas[Insul=="After"],Temp[Insul=="After"])
> lines(abline(6.5513,-0.3367))
> lines(abline(4.9861,-0.3367))
> plot(Gas~Temp,pch=19,subset=Insul=="Before")
> points(Gas[Insul=="After"],Temp[Insul=="After"])
> lines(abline(6.858,-0.3952))
> lines(abline(4.7328,-0.2779))
>
```

and we can see that visually the parallel line model looks as if it does not fit the points very well. Since the parallel line model is nested inside the separate line model we can test this formally with an analysis of variance as follows:

```
> anova(gasPAR,gasBA)
Analysis of Variance Table

Model 1: Gas ~ Insul + Temp - 1
Model 2: Gas ~ Insul + Insul:Temp - 1
  Res.Df    RSS Df Sum of Sq      F    Pr(>F)
1     53 6.7704
2     52 5.4252  1    1.3451 12.893 0.0007307 ***
```

which shows that the model with different slopes is significantly better fitting than the one with the slopes constrained to be parallel.

4.3 Robust Regression

Just as we have robust and resistant summary measures, or more technically robust estimates of location and scale, it is possible to define ***robust regression*** methods which are not so influenced by outliers. The only techniques provided in **R** is the function `rlm()`, (but in S-plus there are several others).

Consider again the Scottish Hill Races Data where it was found that observations 7 and 18 were outliers. The ordinary least squares fits to the full data set and after dropping the two outliers is given below:

```
> lm(time~dist+climb)

Call:
lm(formula = time ~ dist + climb)

Coefficients:
(Intercept)         dist        climb
   -8.99204      6.21796      0.01105

> lm(time~dist+climb,data=hills[-c(7,18),])

Call:
lm(formula = time ~ dist + climb, data = hills[-c(7, 18),
])

Coefficients:
(Intercept)         dist        climb
 -10.361646    6.692114    0.008047
```

Note how much the estimates change after dropping the two outliers.

Now consider the results of the robust regression using `rlm()`

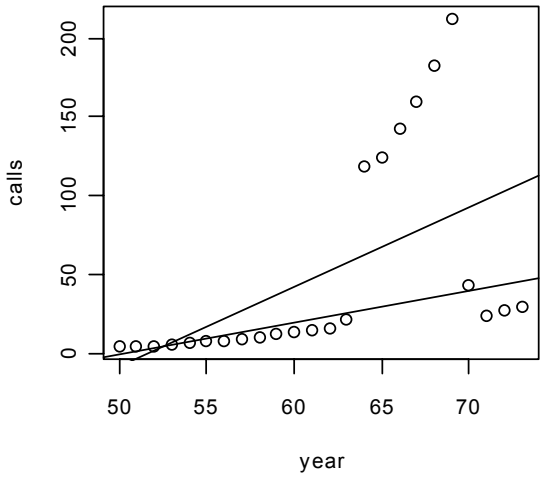
```
> rlm(time~dist+climb)

Call:
rlm.formula(formula = time ~ dist + climb)
Converged in 10 iterations

Coefficients:
 (Intercept)         dist        climb
-9.606716592  6.550722947  0.008295854

Degrees of freedom: 35 total; 32 residual
Scale estimate: 5.21
```

The estimates are nearly the same as using ordinary least square on the reduced data set.

Another example is a set of data giving the numbers of phone calls (in millions) in Belgium for the years 1950-73. In fact, there had been a mis-recording and the data for six years was recorded as the total length and not the number.

```
> plot(year,calls)

> lines(abline(lm(calls~year)))

> lines(abline(rlm(calls~year,maxit=50)))
```

gives a plot of the data with the fitted least squares line and the line fitted by a robust technique.



The diagnostic function `plot.lm()` can also be used with the results of a robust fit using `rlm()` and you are encouraged to try it, as well as investigating the full summary output of both `lm()` and `rlm()` for this data set.

4.4 Bootstrapping linear models

Care needs to be taken when bootstrapping in linear models. The obvious idea of taking samples with replacement of the actual data points $\{(x_i,y_i);I=1,\ldots,n\}$ is not usually appropriate if we regard the values of the independent variable as fixed. In the `whiteside` data set the x-values were temperatures and it might be argued that you could select pairs of points then.

The more usual technique is first to fit a model and then resample, with replacement, the residuals and create a bootstrap data set by adding on the resampled residuals to the estimated fits.

Specifically, if our model is $y_i=\alpha+\beta x_i+\varepsilon_i$ then we estimate α and β to obtain $\hat{\varepsilon}_i = y_i - \hat{\alpha} - \hat{\beta}x_i$, then we create a new bootstrap data set $\{(x_i, y_i^*);i = 1,\ldots,n\}$ where $y_i^* = \hat{\alpha} + \beta x_i + \varepsilon_i^*$ and the (ε_i^*) are resampled with replacement from the $(\hat{\varepsilon}_i)$.

More details of this and other bootstrap techniques are given in Davison & Hinkley (1997) *Boostrap Methods and their Applications, C.U.P.* The library boot contains routines described in that book. It may also be noted that the library Bootstrap contains routines from the book *An Introduction to the Bootstrap* by Efron & Tibshirani (1993), Chapman and Hall.

4.5 Generalized Linear Models

4.5.1 Introduction

Generalized linear models extend linear models to various non-normal response distributions and transformations to linearity. This section will consider models for binomial and Poisson data whose means depend upon predictor variables. Many other distributions can be handled, including gamma and inverse-Gaussian, in fact any distribution within the exponential family can in principle be handled by glms. The appropriate linearizing transformation (i.e. precisely what measure of response is used) is primarily dependent upon the distribution of the data, for example with Poisson data (i.e. *counts*) it is usually appropriate to model the *log(counts)* (or more specifically *log(mean)*) as a linear function of predictors, with binomial data (i.e. proportions of successes in Bernouilli trials) it is generally convenient to model the *logit(proportion)* (or more specifically *logit(p),* where *p* is the binomial probability of success) as a linear function of predictors.

The function used in **R** to perform the analysis is `glm()` and the informal interpretation of the results follows closely upon that of ordinary linear models although there are some important technical differences.

4.5.2 Logistic Regression

Logistic regression applies to observations from Binomial B(n,p) data.

The model is

$$\text{logit}(p)=\log\{p/(1-p)\}=\alpha+\beta_1 x_1+\ldots+\beta_k x_k$$

where $x_1,\ldots,x_k$ are the explanatory variables, either continuous covariates or factors. Transforming this equation gives the model as

$$p=\exp\{\alpha+\beta_1 x_1+\ldots+\beta_k x_k\}/[1+\exp\{\alpha+\beta_1 x_1+\ldots+\beta_k x_k\}]$$

and estimation of the β_i is by maximum likelihood based on the Binomial distribution. A particular case is with 0/1 data (i.e. the Binomial reduces to a Bernouilli variable reflecting 'success/failure'. Logit(p) is also known as the *log odds.*

Example: `smokerspulse.txt` is a data set recording whether the pulse rate in subjects is High or Low, whether they smoke (Yes or No) and their weight (in pounds). The question of interest is whether a high pulse rate is related to whether or not they smoke and to their weight.

```
> smokerspulse<- read.table(file="a:\\smokerspulse.txt")
> attach(smokerspulse)
> table(pulse,smokes)
       smokes
pulse  No Yes
  High 12  10
  Low  52  18
```


The `glm()` function works just as the `lm()` but needs an extra statement declaring that we are dealing with binomial data.

```
> glm1<- glm(pulse~smokes+weight, family=binomial)
> summary(glm1)

Call:
glm(formula = pulse ~ smokes + weight, family = binomial)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.0159   0.3107   0.5851   0.7668   1.3858

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.98710    1.67641  -1.185   0.2359
smokesYes   -1.19295    0.55221  -2.160   0.0307 *
weight       0.02502    0.01223   2.046   0.0407 *
---
Signif. codes:  0  `***'  0.001  `**'  0.01  `*'  0.05  `.'  0.1
` '  1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 101.21  on 91  degrees of freedom
Residual deviance:  93.64  on 89  degrees of freedom
AIC: 99.64

Number of Fisher Scoring iterations: 3

>
```


Interpretation: The response variable here has two values: High and Low, and these two are taking alphabetically and models the probability of the second 1, i.e. p=Prob[pulse=Low]. If the values had been coded as 0/1 then p=Prob[1].

The model `pulse~smokes+weight, family=binomial` is shorthand for

logit(p)=α_i+βweight, where i=1 for `smokes=No` and 2 for `Yes`.

We have that the estimate of α_1 is –1.987 and for α_2 it is –1.987–1.193, i.e. `smokesYes` gives the change in intercept for smokers. The p-value for this is 0.0307 showing that there is a significant difference in risk of a low pulse rate for smokers. Since we have the *log odds* for smokers is estimated as *less* than for non-smokers it indicates that smokers have a lower risk of a low pulse rate.

The estimate of β is 0.025 with p-value 0.0407, indicating that there is significant evidence of an increased risk of a low pulse rate with increasing weight.

It would be possible to test further models, e.g. allow for different slopes with `pulse~smokes/weight` and compare these using `anova(.,.)` just as we did with the `whiteside` data. In fact there is no evidence against a model with a common slope for weight.

A**nother Example:** The data below give the results of an experiment on toxicity of an insecticide on the tobacco budworm moth. The data are the numbers out of batches of 20 that were killed after application of a dose which was on a log scale (i.e. dose 2 was twice as strong as dose 1, dose 3 twice as strong as 2 etc).

	dose					
Sex	1	2	3	4	5	6
Male	1	4	9	13	18	20
Female	0	2	6	10	12	16

In this example we will type in the data from the keyboard and set up the factors.

These data are as numbers of 'successes' (i.e. numbers killed) out of 20 and so can be modelled as B(20,p) where p depends on the dose and the sex of the moth. To present these to `glm` we need to construct a two column matrix giving the numbers of 'successes' and 'failures', note the use of `cbind` to do this.

```
> ldose<- rep(0:5,2)
> ldose
 [1] 0 1 2 3 4 5 0 1 2 3 4 5
> sex<- factor(rep(c("M","F"),c(6,6)))
> sex
 [1] M M M M M M F F F F F F
Levels:  F M
> numdead<-c(1,4,9,13,18,20,0,2,6,10,12,16)
> SF<-cbind(numdead,20-numdead)
> budworm1<-glm(SF~sex*ldose,family=binomial)
```


```
> summary(budworm1)


Call:
glm(formula = SF ~ sex * ldose, family = binomial)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.39849  -0.32094  -0.07592   0.38220   1.10375

Coefficients:
```

	Estimate	Std. Error	z value	Pr(>|z|)	
(Intercept)	**-2.9935**	**0.5525**	**-5.418**	**6.02e-08**	*******
sexM	**0.1750**	**0.7781**	**0.225**	**0.822**	
ldose	**0.9060**	**0.1671**	**5.424**	**5.84e-08**	*******
sexM:ldose	**0.3529**	**0.2699**	**1.307**	**0.191**	

```
---
Signif. codes:  0  `***'  0.001  `**'  0.01  `*'  0.05  `.'  0.1
` '  1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 124.8756  on 11  degrees of freedom
Residual deviance:   4.9937  on  8  degrees of freedom
AIC: 43.104

Number of Fisher Scoring iterations: 3
```

This shows that there is strong evidence of an increase in probability of 'success' with dose, little evidence of a difference in slopes between the sexes and little evidence of a difference between the sexes at **zero log dose.**

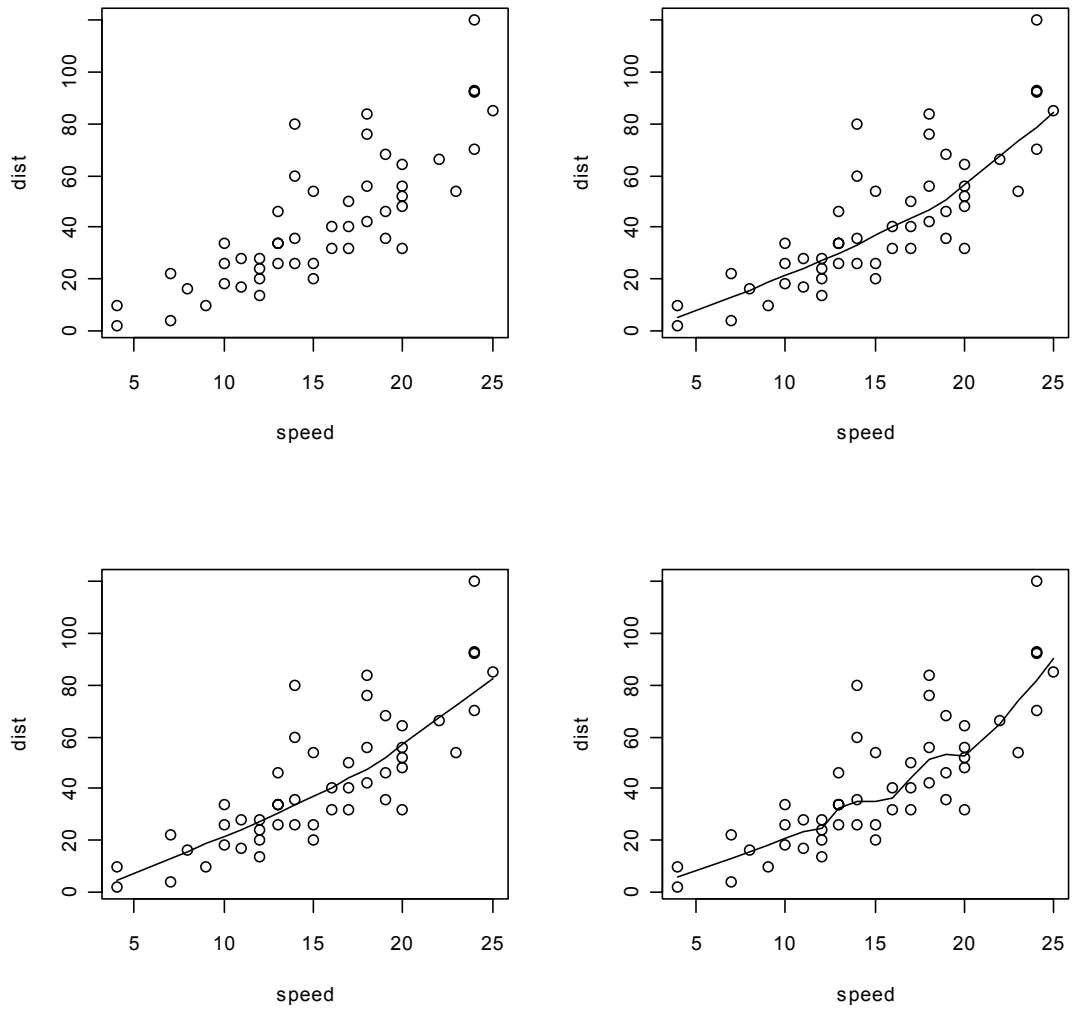
4.6 Scatterplot smoothing and smooth regression

A useful function for investigating structure in a scatterplot is the `lowess()` function. In outline, this fits a curve to a small window of the data, sliding the window across the data. The curve is calculated as a locally weighted regression, giving most weight to points close to the centre of the window with weights declining rapidly away from the centre to zero at the edge. It is possible to control the width of the window by a parameter which specifies the proportion of points included. The default value is 2/3 and larger values will give a smoother line, smaller ones a less smooth line.

Example: The data set `cars` gives the stopping distances for various speeds of 50 cars.

```
> plot(cars)
> plot(cars)
> lines(lowess(cars))
> plot(cars)
> lines(lowess(cars,f=0.8))
> plot(cars)
> lines(lowess(cars,f=0.3))
```

The plots are given on the next page. Note that for this data set of two variables we do not need to specify the names of the variables. If the data set had several variables the we would have to `attach` the data set to be able to plot specific variables, although if it is just the first two we need we could give just the name of the data set as here.

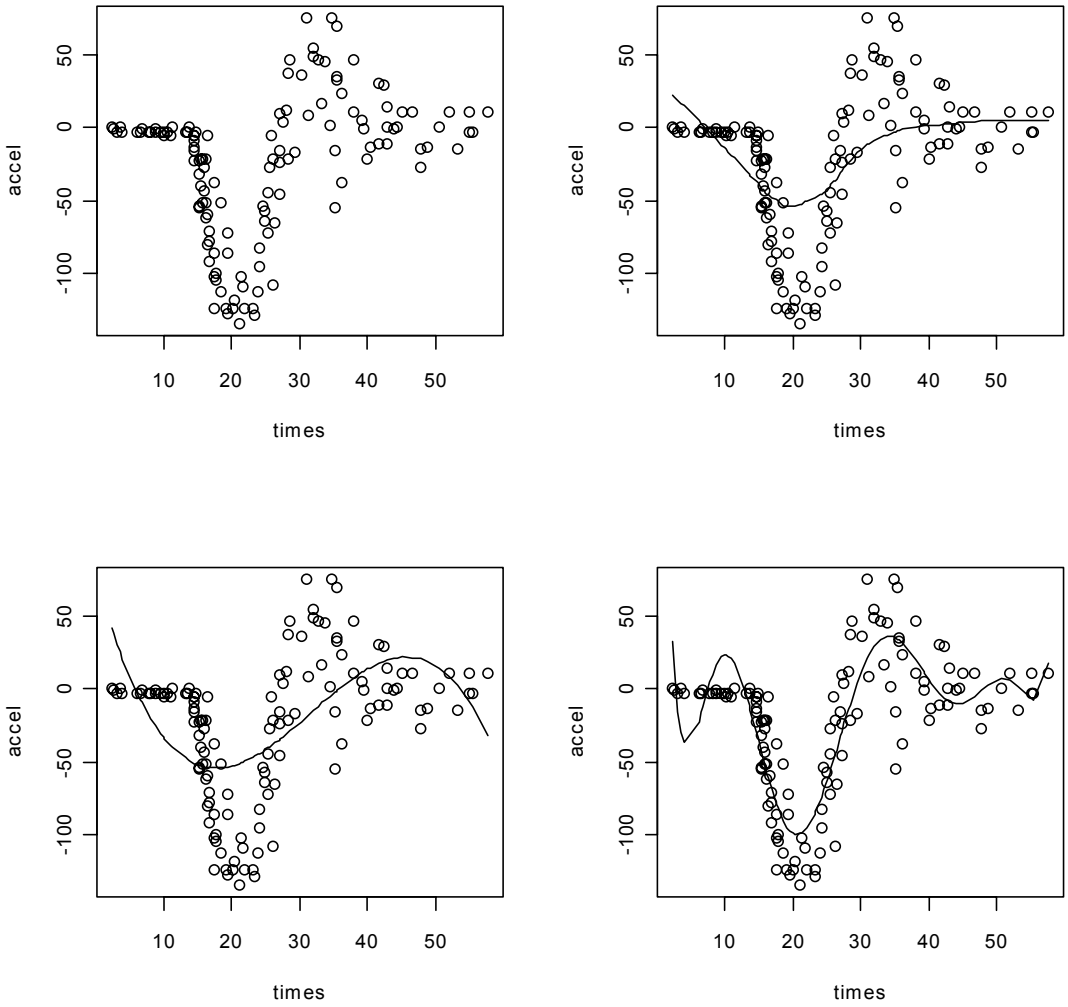


Although the data are very 'noisy' we can see definite evidence that the stopping distance increases more sharply with higher speeds, i.e. that the relationship is not completely linear. It provides an informal guide to how we should model the data.

The `lowess` smoother is an example of a smooth regression function.

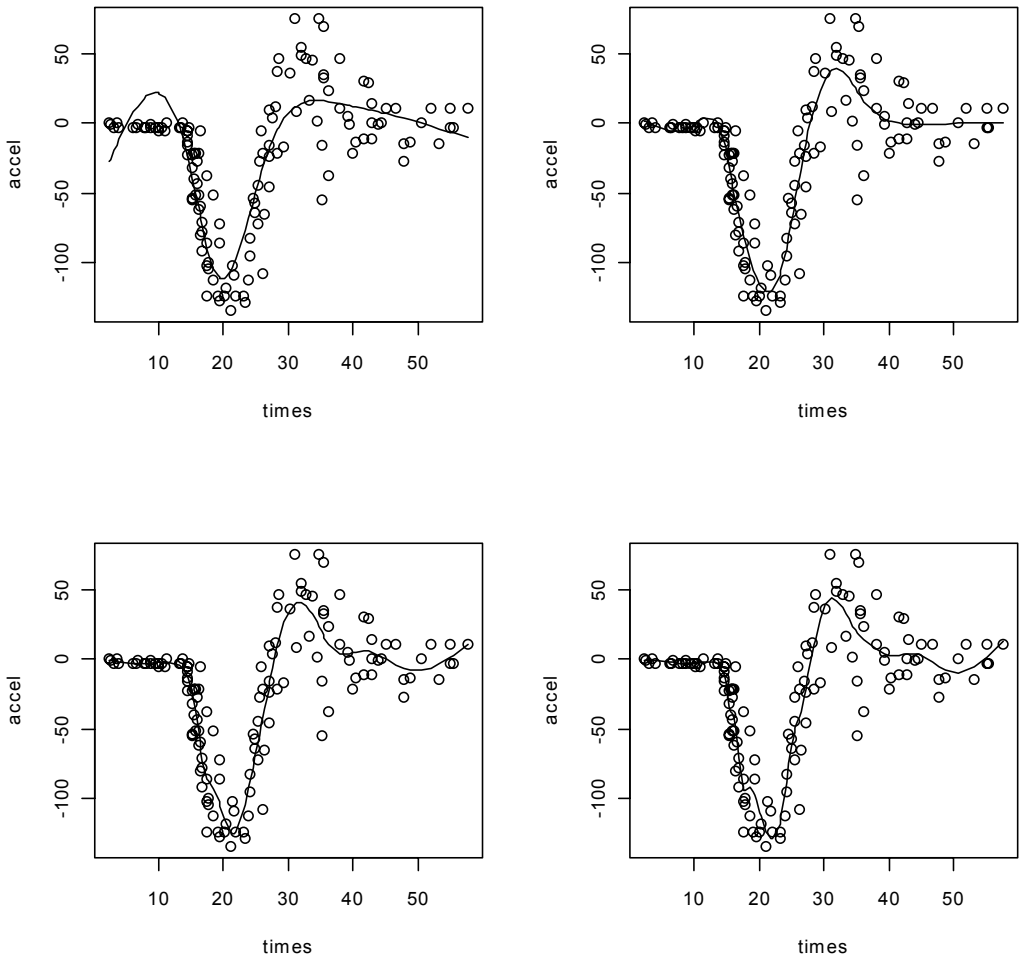
Other smooth regressions can be done with *natural splines* using function `ns()` in the `splines` library and kernel smoothing local regression. generally these work much more satisfactorily than polynomial regression. We illustrate some of these on a data set mcycle which gives the accelerations at times in milliseconds after impact.

```
> data(mcycle)
> attach(mcycle)
> summary(mcycle)
     times           accel
 Min.   : 2.40   Min.   :-134.00
 1st Qu.:15.60   1st Qu.: -54.90
 Median :23.40   Median : -13.30
 Mean   :25.18   Mean   : -25.55
 3rd Qu.:34.80   3rd Qu.:   0.00
 Max.   :57.60   Max.   :  75.00
> plot(mcycle)
> plot(mcycle)
> lines(lowess(mcycle))
> plot(mcycle)
> lines(times,fitted(lm(accel~poly(times,3))))
> plot(mcycle)
> lines(times,fitted(lm(accel~poly(times,8))))
```

Note how unsatisfactory even a very high order polynomial regression is.

```
> library(splines)
> plot(mcycle)
> lines(times,fitted(lm(accel~ns(times,df=5))))
> plot(mcycle)
> lines(times,fitted(lm(accel~ns(times,df=10))))
> plot(mcycle)
> lines(times,fitted(lm(accel~ns(times,df=15))))
> plot(mcycle)
> lines(times,fitted(lm(accel~ns(times,df=20))))
```

The degree of freedom parameter controls the smoothness and it can be seen that a sensible choice is some around 10 for this data set.

More general regression models are ***generalised additive models*** which have the form

$$Y = \alpha + \sum_{j=1}^{p} f_j(X_j) + \varepsilon$$

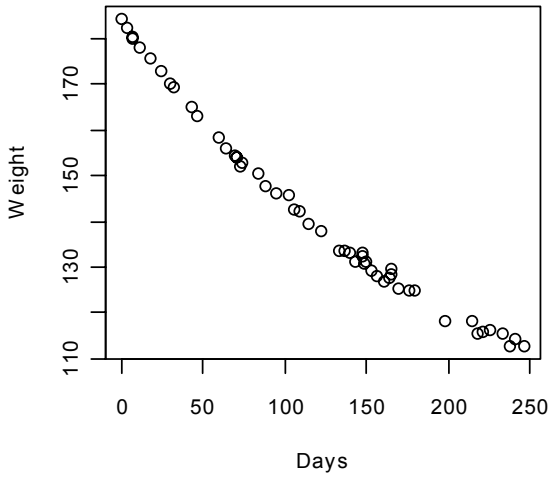
for some smooth functions $f_j(.)$

4.7 Example of non-linear regression

This example fits a model to data giving the weight loss in kg of a subject over a period of 250 days. The data are in dataset wtloss in the MASS library. There are strong theoretical grounds supporting a model of the form $y = \alpha + \beta.2^{-t/\theta} + \varepsilon$ which is linear in the parameters α and β but non-linear in the parameter θ.

To fit this model we need to have starting values for the iterative estimation of the parameters and this can be difficult to determine in general cases.

```
> data(wtloss)
> attach(wtloss)
> wtlossfm<-
> library(nls)
> wtlossfm<- nls(Weight~a+b*2^
         (-Days/theta),start=c(a=90,b=95,theta=120))
> wtlossfm
Nonlinear regression model
  model:  Weight ~ a + b * 2^(-Days/theta)
   data:  list
        a         b     theta
 81.37375 102.68417 141.91052
 residual sum-of-squares:  39.2447
> plot(Days,Weight)
```

A summary of the fitted model will produce standard errors of the estimates and inference can be carried out [almost] as with any other linear model. It is left as an exercise to plot the fitted model on the data.

4.8 Summary

- In this section we have presented the standard methods of fitting regression models on one or more variables, including cases where the independent variables are continuous covariates and/or factors with discrete levels.

- Diagnostic methods, including searches for **outliers** and **influential** observations, were mentioned and illustrated. Ideas of **robustness** and **bootstrapping** were extended to regression situations.

- Generalized linear models were illustrated in the particular case of **logistic regression** but other forms were mentioned.

- **Smooth regression** was introduced with the idea of the **lowess** function (also available in many other packages) and illustrations using **natural splines** were presented.

- An example of a **non-linear** regression model was given.

The overall theme of this chapter is that there are many widely different regression models that can be handled in very similar ways without necessarily knowing the full details of all the mathematical theory underlying them.

Exercises 2

1. Familiarize yourself with any of the worked examples in chapter 3 and 4 that you are unsure of.

2. Estimate the median (providing standard errors of the estimate) of the waiting times in the data set on the Old Faithful geyser eruptions, using (a) a kernel density estimate with a variety of appropriate bandwidths and (b) bootstrapping.

3. In the data on shoes, perform a randomisation assessment of a two-sample t-test.

4. For the hills data, fit a model of the form time=αdist$^{\beta}$climb$^{\gamma}$, removing any outliers and investigating other diagnostics.

5. Investigate the resistant regression function `lqs()` available in library `lqs` on the `hills` and `phones` data

[For Q3 you may find the following function helpful:

```
> ttest<- function(x,y) (mean(x)-
mean(y))/sqrt(var(x)/length(x)+var(y)/length(y))
```

Check:

```
> ttest(A,B)
[1] -0.3689106
> t.test(A,B)
         Welch Two Sample t-test
data:  A and B
t = -0.3689, df = 17.987, p-value = 0.7165
```