**Non- Linear Models Continued**

The general form for a model with one explanatory variable x is

Y = f(x) + 

We have methods for estimating the parameters of the model when f(x) is

f(x) = ax + b

f(x) = aerx.  (using log)

f(x) = axr. (using log)

f(x) = a + bx2. (using substitution, u = x2)

f(x) = a + bx4. (using substitution u = x4)

**What about a general quadratic model Y = a0 + a1x + a2x2 ?**

**There are two problems with this and our methods so far:**

1. **Log and substitution won’t work**
2. **There are three parameters.**

**More good news about R**

R has a procedure that creates estimated models in the non-linear cases directly.

**A Simulation Example.**

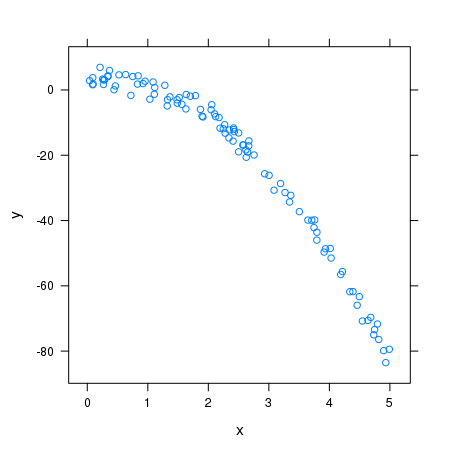
> x<-runif(100,0,5)

> error<-rnorm(100,0,2)

> y<-a0+a1\*x+a2\*x\*x+error

> ex<-data.frame(y,x)

> xyplot(y~x,data=ex)



> model<-nls(y~a0+a1\*x+a2\*x^2,data=ex, start=list(a0=1,a1=2,a2=1))

> summary(model)

Formula: y ~ a0 + a1 \* x + a2 \* x^2

Parameters:

Estimate Std. Error t value Pr(>|t|)

a0 2.7970 0.5472 5.111 1.61e-06 \*\*\*

a1 2.5116 0.5087 4.937 3.30e-06 \*\*\*

a2 -3.9306 0.0991 -39.665 < 2e-16 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.978 on 97 degrees of freedom

Number of iterations to convergence: 1

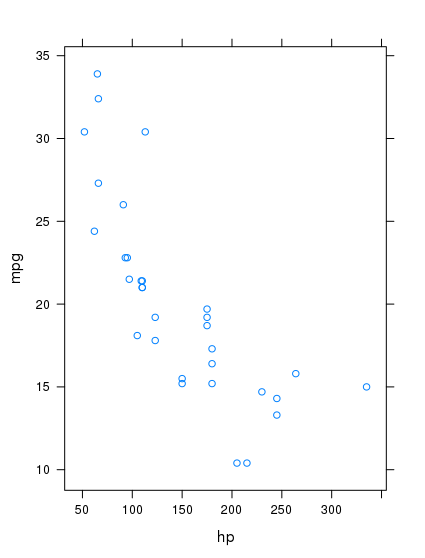
Achieved convergence tolerance: 4.717e-08

**Estimated Model:** Y = 2.797 + 2.512x + -3.931x2 + 

**Example**

The data-frame mtcars contains data on 1974 models, including horsepower (**hp**) and efficiency (**mpg**).

> xyplot(mpg~hp,data=mtcars)



It looks like an exponential model might fit this data.

mpg = aer\*hp + 

**This model has 2 parameters and can be linearized using logs**.

log(mpg) = log(a) + r\*hp

> linmod<-lm(log(mpg)~hp,data=mtcars)

> summary(linmod)

Call:

lm(formula = log(mpg) ~ hp, data = mtcars)

Residuals:

Min 1Q Median 3Q Max

-0.41577 -0.06583 -0.01737 0.09827 0.39621

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 3.4604669 0.0785838 44.035 < 2e-16 \*\*\*

hp -0.0034287 0.0004867 -7.045 7.85e-08 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1858 on 30 degrees of freedom

Multiple R-squared: 0.6233, Adjusted R-squared: 0.6107

F-statistic: 49.63 on 1 and 30 DF, p-value: 7.853e-08

**Parameter Estimates (using log)**

**log(a) = 3.4605, so a = e3.4605 = 31.83**

**r = - .0034.**

**(Note:** That this estimated value of r is small might lead you to think that the true value of r might be 0, but p-value: 7.853e-08 says that the evidence is VERY strong that r is not 0.**)**

**We can also do the estimation using nls method**

> hpmpg<-nls(mpg~a\*exp(r\*hp), data = mtcars, start=list(a=1,r=0))

> summary(hpmpg)

Formula: mpg ~ a \* exp(r \* hp)

Parameters:

Estimate Std. Error t value Pr(>|t|)

a 35.9013612 2.5413733 14.127 8.54e-15 \*\*\*

r -0.0042396 0.0005556 -7.631 1.64e-08 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

**Parameter Estimates (using nls)**

**A = 35.90 r = -.0042**

**A simulation example (using 1000 data points)**

**Model y = axr +**  a = 2, r = 3

**r a**

Using logs 3.033 1.928

Using nls 2.996 2.017

**One more example**

The data-frame **BallDrop** (contained in the package fastR2) contains 30 measurements of the **height** (in meters) from which a ball is dropped and the **time** (in seconds) it falls. These measurements were made in a Calvin physics lab. **Use this data to estimate the gravitational acceleration constant g.**

**Is a linear model appropriate? If not, what kind of model is appropriate? Physics answers this question.**

**height = ½\*g\*time^2**

**Using substitution and a linear model**

**height = g/2\*u. where u = time^2**

> ballmodel<-lm(height~(time^2), data = BallDrop)

> summary(ballmodel)

Call:

lm(formula = height ~ (time^2), data = BallDrop)

Residuals:

Min 1Q Median 3Q Max

-0.074475 -0.031637 -0.003003 0.039694 0.072784

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -0.73606 0.03239 -22.73 <2e-16 \*\*\*

time 3.91814 0.08581 45.66 <2e-16 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.03979 on 28 degrees of freedom

Multiple R-squared: 0.9867, Adjusted R-squared: 0.9863

F-statistic: 2085 on 1 and 28 DF, p-value: < 2.2e-16

**The estimate of g:**

**½\*g = 3.918 g = 7.836 m/sec2**

**Using nls to estimate the model directly**

> ballmodel2<-nls(height~a\*time^2, data = BallDrop, start = list(a=5))

> summary(ballmodel2)

Formula: height ~ a \* time^2

Parameters:

Estimate Std. Error t value Pr(>|t|)

a 4.97729 0.02269 219.4 <2e-16 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.01947 on 29 degrees of freedom

Number of iterations to convergence: 1

Achieved convergence tolerance: 7.432e-10

**The estimate of g**

**1/2g = 4.977 g = 9.954 m/sec2**

**Why does one method give a much better estimate of g?**

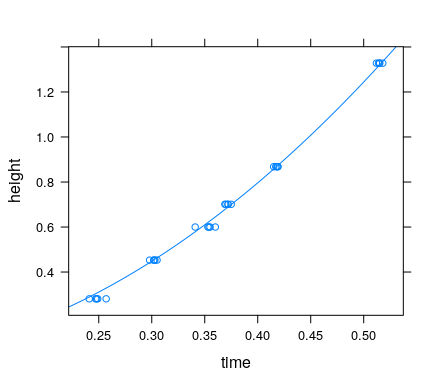
The linear model procedure assumes that the actual form of the substitution model is **height = a\*u + b;** i.e., there is a constant term and it uses the data to estimateboth a and b. In our case, the estimates are a = 3.918 and b = -0.736. The advantage of the nls approach is that we can specify precisely the form of the model and we get a much more accurate result.

**We can have R plot the graph of the estimated model function on the scatter plot of the data.**

> f<-makeFun(ballmodel2)

> xyplot(height~time, data = BallDrop)

> plotFun(f(time)~time, add = TRUE)



**Exercises 17**

1. The data-frame **CoolingWater** contains temperature measurements of a mug of water over time. The variables are **time** (sec) and **temp** (C).
2. Which is the explanatory variable and which is the response variable?
3. Create a scatterplot of the data. Include the R command and the plot.
4. Hopefully, it is clear to you that a linear model is not appropriate. What kind of model is appropriate? From the scatterplot and a bit of physics, it looks like an exponential model (with a negative coefficient in the exponent might be appropriate. Try a model of the form y = aert. (**In R, the syntax would be y = a\*exp(r\*t))** Use nls to create the estimate for such a model. Use a = 100 and r = -1 as your starts. Name your model **Cooling1**. Include the R command that creates the model and the summary output as your answer**.**
5. Follow the pattern on the previous page to create a scatterplot of the data and a graph of the estimated function. Include the R commands and the plot as your answer.
6. Does it look like the model fits the data?
7. What is going wrong? For our model, at t gets larger and larger, the value of the function goes to 0, which is not true for a mug of water in an ordinary room (not a freezer). So, we have to modify the model to take account of an ambient room temperature that is greater than 0. So, our original (bad) model should be modified to y = aert + b (where b is the ambient temperature). The new model has three parameters: a,r,b.

Use nls to create an estimate of the new model. Call this model **Cooling2**. Include the R command creating it and the summary output as your answer.

1. Redo (d) using the new model Cooling2. Include the R commands and the plot as your answer.
2. How does the model Cooling2 do in fitting the data? What is the estimate of the room’s ambient temperature?
3. The data-frame **Spheres (contained in the package fastR2)** contains the measured diameters (m) and masses (kg) for 12 ball bearings made from the same substance. The column headings are **diameter** and **mass**. Let diameter be the explanatory variable and mass the response variable.
   1. What form should the model have?

mass =

* 1. Use the nls method to create an estimated model. Call this model diama. Include the R command and the summary for this.
  2. The density a the ball is

mass/volume = a\*diameter^3/(1/6\*pi\*diameter^3) = 6a/pi.

What is the estimated of density of the material these balls are made of?

* 1. Create a scatterplot of the original data with the graph of the diama model function added.