

22s:164:Lab11

Nov. 29, 2007

1. Nonlinear Regression

- The command `nls` is used to obtain OLS and WLS estimates for nonlinear regression models. `nls` differs from `lm` in a few key respects: first, the formula for defining a model is different. In `lm`, the formula specifies the names of the response and the terms, but not the names of the parameters. For a `nls` model, the formula specifies both terms and parameters. Secondly, factors are generally not used with `nls`. Last, a named list `start` of starting values must be specified. This argument tells the algorithm where to start, and also to name the parameters. Look at the example using data `turk0`:

```
> library(alr3)
> data(turk0);attach(turk0)
> n1 <- nls(Gain ~th1 +th2*(1-exp(-th3*A)),start=list(th1=620,th2=200,th3=10))
> summary(n1)
```

- Objects created by `nls` have `summary`, `plot` and `predict` methods, and there are used in a way that is similar to linear regression models.

```
> plot(A,Gain)
> x <- (0:44)/100
> lines(x,predict(n1,data.frame(A=x)))
```

- Lack-of-fit testing is possible if there are repeated observations. The idea is to compare the nonlinear fit to the one-way analysis of variance, using the levels of the predictor as a grouping variable:

```
> m1 <- lm(Gain~as.factor(A))
> anova(n1,m1)
```

- Unlike `lm`, `nls` does not allow factors in a formula. To use factors, you need to create dummy variables for the various levels of the factor. For example, the `turkey` data includes a variable `S` that is a factor with three levels. Also, we need to use weighted least squares in this case.

```
> data(turkey)
> attach(turkey)
#create the dummy variable
> S1=S2=S3 <- rep(0,dim(turkey)[1])
> S1[S==1] <- 1;S2[S==2] <- 1;S3[S==3] <- 1
# computes the weighted response
> wGain <- sqrt(m)*Gain
# fit the models
# common regression
> m4<- nls(wGain ~ sqrt(m)*(th1+th2*(1-exp(-th3*A))),
          start=list(th1=620,th2=200,th3=10))
# most general
> m1 <-nls(wGain ~sqrt(m)*(S1*(th11+th21*(1-exp(-th31*A)))+
          S2*(th12+th22*(1-exp(-th32*A)))+S3*(th13+th23*(1-exp(-th33*A)))),
          start=list(th11=620,th12=620,th13=620,th21=200,th22=200,th23=200,
                    th31=10,th32=10,th33=10))
```

```

# common intercept
> m2 <- nls(wGain~sqrt(m)*(th1+S1*(th21*(1-exp(-th31*A)))+
           S2*(th22*(1-exp(-th32*A)))+S3*(th23*(1-exp(-th33*A)))),
           start=list(th1=620,th21=200,th22=200,th23=200,th31=10,th32=10,th33=10))
# common intercept and asymptote
> m3 <- nls(wGain~sqrt(m)*(th1+th2*(S1*(1-exp(-th31*A))+S2*(1-exp(-th32*A))+
           S3*(1-exp(-th33*A))))),start=list(th1=620,th2=200,th31=10,th32=10,th33=10))
> anova(m4,m2,m1)
> anova(m4,m3,m1)

```

2. Serial Correlation (AR1)

- In the dataset bookstore from s164 library, errors are serially correlated. The function `acf` computes (and by default plots) estimates of the autocorrelation function. `Durbin.watson` numerically tests the autocorrelation.

```

> library(s164)
> book.lm = lm(Sales ~ Year + factor(Month), data=bookstore)
> summary(book.lm)
> acf(resid(book.lm), plot=FALSE)
> library(car)
> durbin.watson(book.lm)

```

```

lag Autocorrelation D-W Statistic p-value
  1      -0.2330984      2.450914  0.034
Alternative hypothesis: rho != 0

```

The ACF shows mild but stat. sig. evidence of a negative autocorrelation

- This function `gls` fits a linear model using generalized least squares. The errors are allowed to be correlated and/or have unequal variances.

```

> library(nlme)
> book.gls = gls(Sales ~ Year + factor(Month), data=bookstore, corr = corAR1())
> summary(book.gls)

```

Compare with results for `book.lm`. For example, the `se` for the Year effect is smaller and the estimate is similar, making the `t` ratio a bit more significant.

```

> intervals(book.gls)
# Approximate 95% confidence intervals
> acf(resid(book.gls), plot=FALSE)
> acf(resid(book.gls, type="normalized"), plot=FALSE)

```

The "normalized" residuals are "whitened" based on the estimated AR1 model
We don't see any autocorrelations in these.