ORIGIN ≡ 0

Localized Polynomial Regression & Splines

Localized Polynomial Regression, termed "loess" or "lowess" more-or-less synonymously (although some authors imply weighted for the latter), comprise a family of methods designed to fit complex curvilinear data. The general principal involves using a variant of least-squares regression to fit various order polynomial functions, sometimes weighted, to local regions of the curve, followed by seamless splicing of all fits into a single, often complex, overall curve. Within usual implementations of the method are parameters that allow one to define complexity of each sub-curve and the size of local regions. These parameters often need adjustment in order to produce visually acceptable results. The general purpose for most applications of loess is graphic visualization and heuristic description of the data without further testing. Recent advancements with General Additive Models (GAM), however, allow users to compare curvilinear models in a way analogous to Linear Models. For examples of testing with GAM, see Worksheet AM 020. For further details on the method of loess fitting, see: http://en.wikipedia.org/wiki/Local_regression.

This example is drawn from Zuur et al. 2009, *Mixed Effects Models and Extensions in Ecology with R,* although the data is not used as they intended.

Example in R:

```
#READING AND SORTING DATA FRAME ON LENGTH
setwd("c:/DATA/Models")
L=read.table("clams.txt",header=T)
L
LL=L[order(L$LENGTH) , ]
#order() SORTS DATA FRAME BY ROWS USING VARIALBLE LENGTH
#NOTE USE OF , TO INDICATE SORT BY ROWS
LL
length(LL$AFD)
```

The data "clams.txt" was input and then the entire dataframe was sorted based on values in the column labeled "LENGTH". This is a useful example of the syntax in R that allows effortless sorting. However, many users may prefer to sort in a spreadsheet such as MS Excel.

```
> LL
    MONTH LENGTH   AFD LNLENGTH  LNAFD
278     4   5.66 0.002    1.733 -6.215
279     4   5.82 0.002    1.761 -6.502
277     4   5.99 0.002    1.790 -6.266
276     4   7.13 0.003    1.964 -5.952
274     4   7.46 0.004    2.010 -5.449
...
6      11  28.13 0.187    3.337 -1.679
1      11  28.38 0.248    3.346 -1.394
229     2  28.83 0.271    3.361 -1.307
253     4  29.01 0.250    3.368 -1.387
227     2  29.13 0.342    3.372 -1.074
245     4  30.25 0.274    3.409 -1.295
196     2  30.32 0.366    3.412 -1.006
212     2  30.37 0.336    3.413 -1.092
205     2  30.59 0.314    3.421 -1.157
198     2  30.85 0.312    3.429 -1.165
211     2  31.32 0.338    3.444 -1.086
200     2  31.63 0.420    3.454 -0.868
7      11  32.58 0.361    3.484 -1.020
199     2  34.19 0.565    3.532 -0.572
```

```
> length(LL$AFD)
[1] 398
```

Total length of the dataset is 398. Sorting messes up order of numerals in the first colum, but this doesn't affect R's use of the data.

```
#USING loess() FIT OF DATA IN R BASE PACKAGE
LOW=loess(AFD~LENGTH,data=LL)
summary(LOW)
```

```
#PLOTTING ORIGINAL DATA & loess() PREDICTION
#BY CALLING VARIABLES WITHIN LL USING $
plot(LL$LENGTH,LL$AFD,pch=19,col='blue',xlab="LENGTH",ylab="AFD")
points(LL$LENGTH,predict(LOW),type='l',col='red')
```

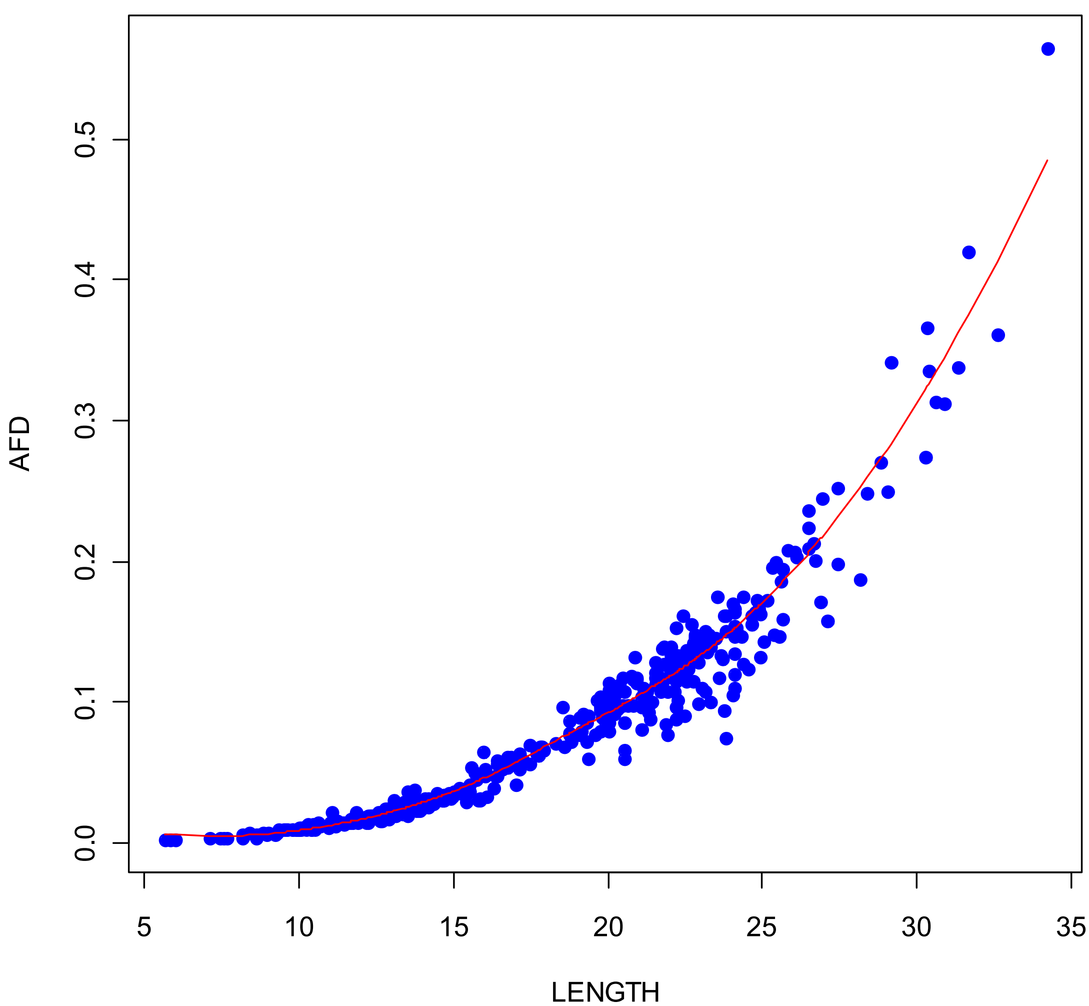
> summary(LOW)

```
Call:
loess(formula = AFD ~ LENGTH, data = LL)

Number of Observations: 398
Equivalent Number of Parameters: 4.77
Residual Standard Error: 0.01557
Trace of smoother matrix: 5.21

Control settings:
  normalize:  TRUE
  span      :  0.75
  degree    :  2
  family    :  gaussian
  surface   :  interpolate          cell = 0.2
> LOW
Call:
loess(formula = AFD ~ LENGTH, data = LL)

Number of Observations: 398
Equivalent Number of Parameters: 4.77
Residual Standard Error: 0.01557
```



Cubic (and other) Splines

An increasingly popular alternative to Polynomial curve fitting is the use of Splines (Cubic Spline, Thin-plate Splines, and others) to fit a curvilinear pattern. As a means for visualizing the shape of a complex curve, differences between all methods are usually very small. The {mgcv} package in R implements Spline fits called "smoothers" as the basis for GAM models. The same data will be fit with a Cubic Spline for comparison here.

```
#CUBIC SPLINE
library(mgcv)
LLG=gam(AFD~s(LENGTH,fx=F,k=-1,bs='cr'),data=LL)
LLG
summary(LLG)
```

```
> summary(LLG)

Family: gaussian
Link function: identity

Formula:
AFD ~ s(LENGTH, fx = F, k = -1, bs = "cr")

Parametric coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 0.0842136  0.0007729     109   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
           edf Ref.df    F p-value
s(LENGTH) 6.47  7.511 1222  <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =  0.959   Deviance explained = 95.9%
GCV score = 0.00024231  Scale est. = 0.00023777  n = 398
```

^ see Worksheet AM 020 for interpretation of the summary(), in turn calling summary.gam(), results.

```
#CALCULATING FIT & CONFIDENCE BOUNDS
LLGpred=predict(LLG,se=T,type='response')
LLGpredF=LLGpred$fit
LLGpredU=LLGpred$fit+2*LLGpred$se
LLGpredL=LLGpred$fit-2*LLGpred$se
```

Predicted Values for the cubic spline fit are extracted by the generic function predict(), which calls predict.gam(), which in turn makes the object LLGpred. From this, one extracts the values of fit using the construction LLGpred$fit. Zuur et al. also use the standard error of the prediction (fit) in LLGpred$se to calculate an approximate 95% Confidence Interval. This is done by adding and subtracting LLG$fit by 2*LLGpred$se.

```
#PLOTTING DATA & gam() FIT WITH APPROXIMATE 95% CONFIDENCE BOUNDS
plot(LL$LENGTH,LL$AFD,pch=19,col='blue',xlab="LENGTH",ylab="AFD")
points(LL$LENGTH,LLGpredF,type='l',col='green')
points(LL$LENGTH,LLGpredU,type='l',col='red')
points(LL$LENGTH,LLGpredL,type='l',col='brown')
```