STAT 621 Lecture Notes
Nonparametric Regression: Linear Smoothers

Next we consider nonparametric regression where minimal assumptions are made on the true regression function. The class of estimators we consider first are called *linear smoothers*. These are estimators that may be written as linear combinations of the responses (the Y's).

Data and Assumptions:

The estimators we will consider here are *local* estimators. That is,

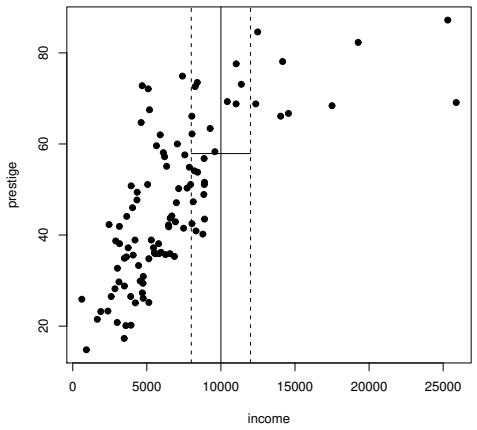
Local Averaging

- We would like to estimate the regression function at a certain point x_0, i.e., to estimate $g(x_0)$.
- If g is smooth, then observed values of g at points x near x_0 should be similar to $g(x_0)$.
- Define a bin or window of certain length, and center it at x_0.
- The estimate $\widehat{g}(x_0)$ is simply the average of the observed responses at x values that fall inside this window,
$$\widehat{g}(x_0) = \frac{\sum Y_i I(|X_i - x_0| < h)}{\sum I(|X_i - x_0| < h)}$$
- We repeat the process for a number of these x_0 values (usually equally spaced within the range of the observed data), resulting in estimates of g each time. You can picture sliding this window along the x-axis. As the window moves, some points leave the bin and some points enter the bin. Thus the data used in the average is always restricted to values near (determined by the width of the bin) the x where $f(x)$ is being estimated.

Example: These data come from a Canadian study. Occupations were given a prestige rating. The average income and education level of people working in these occupations was inferred from census data. Here's a sample of the data:

	education	income	women	prestige	census	type
ECONOMISTS	14.44	8049	57.31	62.2	2311	prof
SECONDARY.SCHOOL.TEACHERS	15.08	8034	46.80	66.1	2733	prof
MEDICAL.TECHNICIANS	12.79	5180	76.04	67.5	3156	wc
SHIPPING.CLERKS	9.17	4761	11.37	30.9	4153	wc
OFFICE.CLERKS	11.00	4075	63.23	35.6	4197	wc
COMMERCIAL.TRAVELLERS	11.13	8780	3.16	40.2	5133	wc
FIREFIGHTERS	9.47	8895	0.00	43.5	6111	bc
RADIO.TV.REPAIRMEN	10.29	5449	2.92	37.2	8537	bc
BUS.DRIVERS	7.58	5562	9.47	35.9	9171	bc
TYPESETTERS	10.00	6462	13.58	42.2	9511	bc

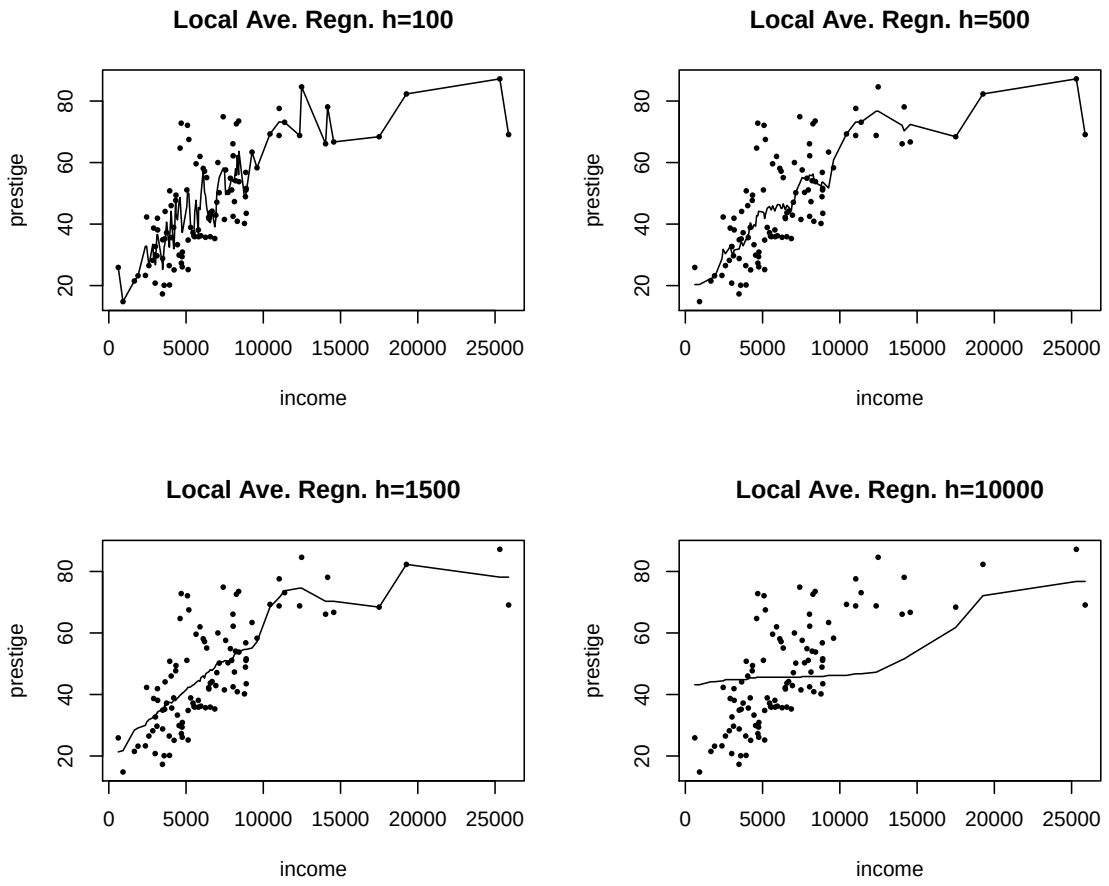
Consider the relationship between income and prestige rating. We'll fit a local average regression to predict prestige (Y) by income (X). The following figure illustrates estimating the function at $X = 10,000$.



A simple R function is given below to compute $\widehat{g}(x)$ over a grid of values.

```
loc.ave=function(x,y,h,xgrid)
{
fhat=rep(NA,length(xgrid))
for(i in 1:length(xgrid))
  fhat[i]=mean(y[abs(x-xgrid[i])<h])
fhat
}

fhat=loc.ave(income,prestige,100,income)  # took obs. income as x-grid
temp=data.frame(income,fhat)              # put x-grid in order for plot
temp=temp[order(income),]
plot(income,prestige,pch=16,cex=.6)
lines(temp$income,temp$fhat)
title("Local Ave. Regn. h=100")
```

Here's the estimated function, using four different window widths.



Comments?

What will happen if we want to estimate $g(x_0)$ for an x_0 that is more that h away from the nearest observation?

k Nearest Neighbor Methods

This is a general approach that allows window width to vary. Here at each point x_0, the window width is defined as the width needed to capture k data points. The local average estimator can be modified to be a nearest neighbor estimator as follows.

$$\widehat{g}(x_0) = \sum_{i=1}^{n} w_i(x_0) Y_i$$

where $w_i(x_0) = 1/k$ if x_i is one of the k nearest points to x_0. Otherwise $w_i(x_0) = 0$. A sketch:

Before moving on, remember I claimed that these estimators are linear smoothers. Explain.

Kernel Estimation

This estimator was developed independently by Nadaraya (1964) and Watson (1964).

- Like local averaging, kernel estimation computes $\widehat{g}(x_0)$ by averaging responses that fall near x_0. However now responses will be weighted depending on exactly how close the x points are to x_0.
- Consider estimating f at x_0. For each data point, a distance to x_0 is measured as $(x_i - x_0)/h$ were h is the bandwidth and essentially controls the window width.
- A Kernel function (K) is used to assign weights. Kernels are symmetric and reach (or approach) 0 as $|x_i - x_0|$ increases. Specifically weights are given by

$$w_i = K\left(\frac{x_i - x_0}{h}\right)$$

- The estimated value $\widehat{f}(x_0)$ is weighted local average of the responses,

$$\widehat{g}(x_0) = \frac{\sum_{i=1}^n w_i y_i}{\sum_{i=1}^n w_i}.$$

  Note again that $\widehat{f}(x_0)$ is a linear smoother.

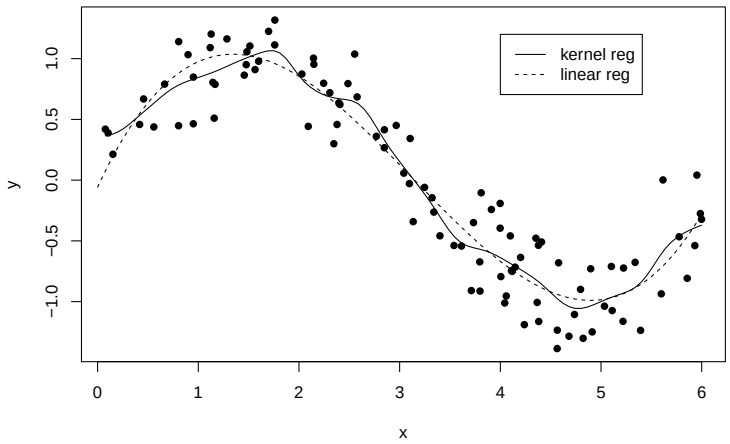
- As we noted in our discussion of density estimation, the choice of kernel is generally unimportant. Common kernels include the Gaussian, tricube, rectangular, etc.

- Typically the bandwidth h is constant. However nearest-neighbor approaches may also used.

Example: The R function `ksmooth` will compute the Nadaraya-Watson kernel regression estimator. This isn't a very sophisticated function. Options for kernel functions are limited to `normal` and `box`. A crude method for bandwidth selection is employed by default, but may fail if data are sparsely sampled in some ranges.

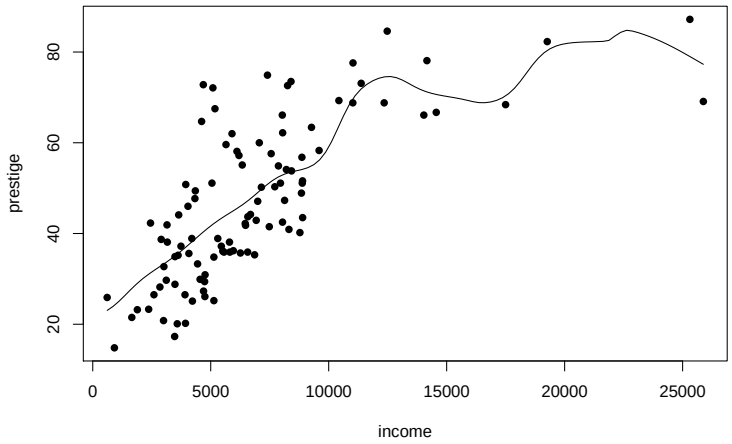
Here is a simple simulated example. I fit both the kernel regression and a linear model that is polynonmial in x.

```
x=runif(100,0,6)
y=sin(x)+rnorm(100)*.25
plot(x,y,pch=16)
ghat=ksmooth(x,y,kernel="normal")
lines(ghat$x,ghat$y)

ghat.quad=lm(y~poly(x,3))
xvals=seq(0,6,.05)
pred.quad=predict.lm(ghat.quad,data.frame(x=xvals))
lines(xvals,pred.quad,lty=2)
```



Example: Below the regression function for prestige and income is estimated using kernel regression. Note the default bandwidth rule failed here and I picked a value myself.

```
plot(income,prestige)
kreg=ksmooth(income,prestige,bandwidth=2500,kernel="normal")
lines(kreg$x,kreg$y)
```



Next up we'll define local polynomial regression. But if there's time, let's consider the justification for the form of these linear smoothers. Sketch the derivation of the Nadaraya-Watson estimator.

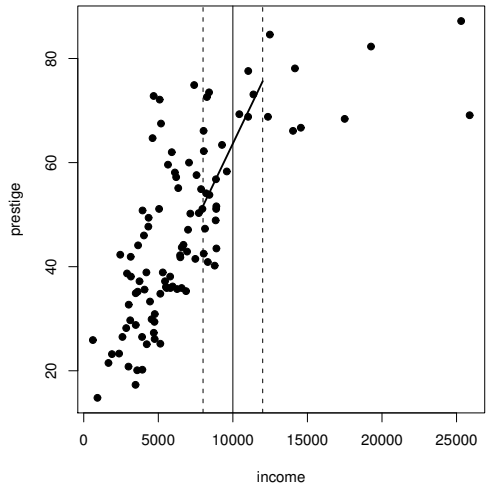
Local Polynomial Regression

- Local polynomial regression estimates $g(x_0)$ with a locally-weighted regression function.
- A polynomial regression function of order p,

$$y_i = \alpha + \beta_1 x_i + \beta_2 x_i^2 + \ldots + \beta_p x_i^p + e_i,$$

  is fit to the data using weights determined by the same kernel function described above, $w_i = K\{(x_i - x_0)/h\}$. Parameters are typically estimated with weighted least squares. Once the parameters are estimated, $\widehat{g}(x_0)$ is computed as the fitted value of the regression at $x = x_0$.
- Note that $p = 1$ gives local linear regression, $p = 2$ gives local quadratic, and $p = 0$ is just the local average. Loess (lowess) regression is a type of local polynomial regression.

Example: The plot below depicts local linear estimation of $g(x_0)$ at $x_0 = 10000$ for the prestige data.

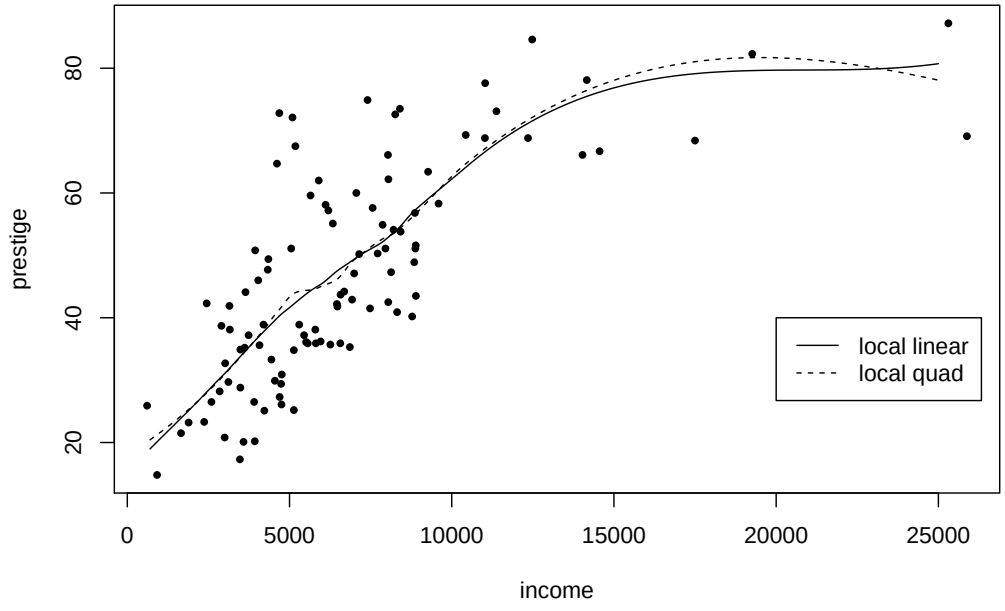


The R function loess() computes a local polynomial regressions up to order 2. Here's some code and output using the prestige data. Note `span` is the argument that controls the smoothing parameter or bandwidth. The default is .75 (???).

```
attach(Prestige)
loreg1=loess(prestige~income,span=.6,degree=1)

names(loreg1)
 [1] "n"         "fitted"    "residuals" "enp"       "s"         "one.delta"
 [7] "two.delta" "trace.hat" "divisor"   "pars"      "kd"        "call"
[13] "terms"     "xnames"    "x"         "y"         "weights"


summary(loreg1)
```

```
   Call:  loess(formula = prestige ~ income, span = 0.6, degree = 1)
   Number of Observations: 102
   Equivalent Number of Parameters: 4.3
   Residual Standard Error: 11.19
   Trace of smoother matrix: 5.05  (exact)
   Control settings:
     span     :  0.6
     degree   :  1
     family   :  gaussian
     surface  :  interpolate   cell = 0.2
     normalize:  TRUE
    parametric:  FALSE
   drop.square:  FALSE
```

```
loreg2=loess(prestige~income,span=.6,degree=2)

plot(income,prestige,pch=16,cex=.75)
xdat=seq(0,25000,100)
lines(xdat,predict(loreg1,xdat))
lines(xdat,predict(loreg2,xdat),lty=2)
legend(20000,40,c("local linear","local quad"),lty=1:2)
```



Questions: How can we objectively choose a value for the smoothing parameter? Once we fit a model using a linear smoother, what can we learn from them? Can we do any statistical inferences?

Choice of Smoothing Parameter for Nonparametric Regression

Ideally the smothing parameter h would be chosen to minimize the mean integrated squared error (MISE). In the regression context, this is,

As we did with density estimation, we can work with this expression to get a sense of how MISE behaves a function of h. We will see that here it is subject to the same variance-bias tradeoff that we saw with density estimation. Also finding an "optimal" value of h is an interesting theoretical exercise, but not practically useful since it will depend on asymptotic arguments.

For practical situations, we need a rule for estimating h from the data. A popular method is cross-validation. This aims to mininize the average mean squared error,

$$\frac{1}{n}\sum_{i=1}^{n} E\{g(X_i) - \widehat{g}(X_i)\}^2$$

The leave-one-out cross validation score is defined as

The best h is the value that minimizes $CV(h)$. So what would you need to do to find h for a given data set? Sketch out the steps.

Luckily there is a shortcut! Using the fact that $\widehat{g}(x)$ is a linear smoother, one can show that

$$CV(h) = \frac{1}{n}\sum\left(\frac{Y_i - \widehat{g}(X_i)}{1 - L_{ii}}\right)^2$$

Describe L_{ii}. Why is this expression useful?

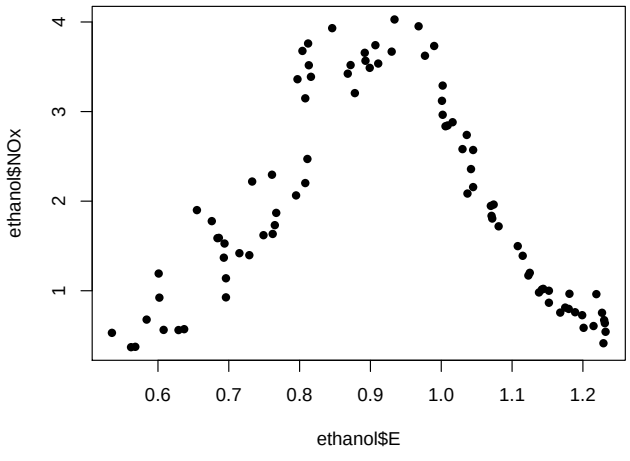
A related measure that is even quicker to compute is called Generalized Cross Validation (GCV). It's basically an approximation to the CV score. This is computed as

$$\frac{1}{n(1-\nu/n)^2}\sum(Y_i - \widehat{g}(X_i))^2$$

where ν =trace(L).

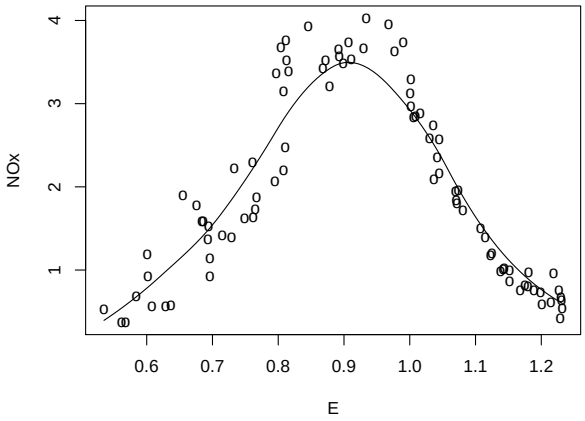
Example: Functions for fitting local polynomial models and computing the GCV bandwidth estimate can be found in the R package `locfit` (the package `locpoly` is another option). This example uses the data set `ethanol` to illustrate these functions. We'll model `NOx` exhaust emissions from a single cylinder engine as a function of `E`, the engine's equivalence ratio. Here's a plot of the data. Would nonparametric regression be a good choice here?

```
library(locfit)
data(ethanol)    # data set in locfit package
plot(ethanol$E, ethanol$NOx, pch=16)
```



Next I use the function `locfit` to fit a local linear regression (`deg=1`) to the data. The argument `alpha` has two components. The second specifies the bandwidth. The first specifies a nearest neighbor fraction, and is ignored if a bandwidth is supplied. Here I arbitrarily chose a bandwidth of $h = 0.15$.

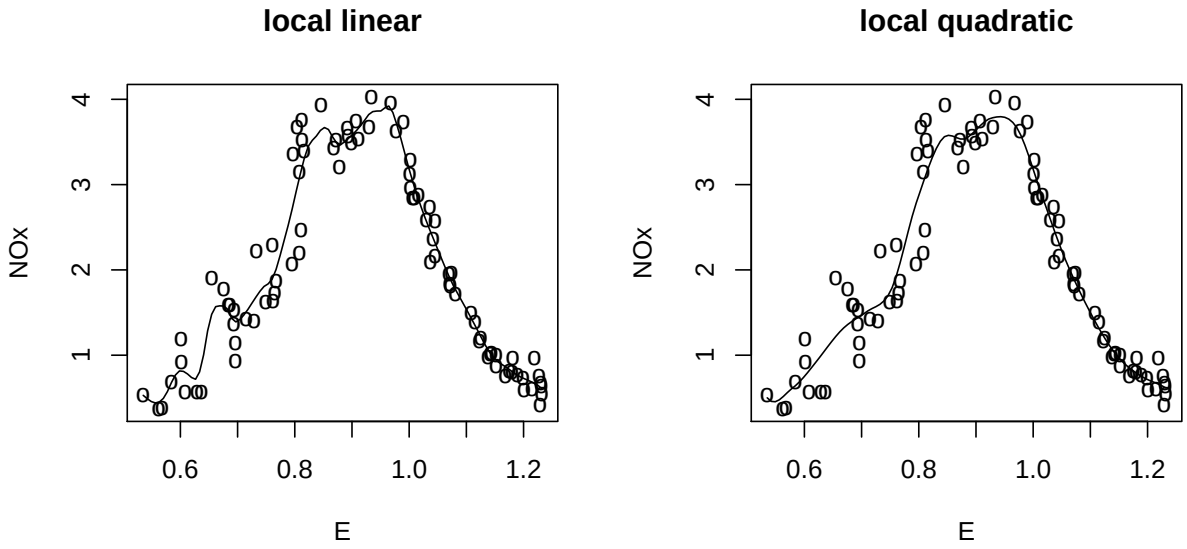
```
fit1=locfit(NOx~E, deg=1, alpha=c(0,.15), data=ethanol)
plot(fit1, get.data=T)
```



The function `gcvplot` will compute the GCV score for provided values of h (or the nearest neighbor argument). Below I compute GCV for a number of these values, and pick out the h that gives the smallest GCV. I use this h to fit the local linear model and make the plot. Plot is below on the right. What do you think?

```
# choose h with min gcv score
h1=seq(0.02,0.1,by=0.01)
alphamat=matrix(0,ncol=2,nrow=length(h1))
alphamat[,2]=h1
gcv.h1=gcvplot(NOx~E,data=ethanol, deg=1, alpha=alphamat)
cbind(gcv.h1$values,h1)
                      h1
      [1,] 0.13937412 0.02
      [2,] 0.10470494 0.03
      [3,] 0.09830998 0.04
      [4,] 0.10151750 0.05
      [5,] 0.10543455 0.06
      [6,] 0.10827901 0.07
      [7,] 0.11075935 0.08
      [8,] 0.11363879 0.09
      [9,] 0.11726739 0.10

h1.opt=h1[gcv.h1$values==min(gcv.h1$values)]
h1.opt
fit2=locfit(NOx~E, deg=1, alpha=c(0,h1.opt), data=ethanol)

par(mfrow=c(1,2))
plot(fit2, get.data=T)   # hmmmmm
title("local linear")
```

Seems a little undersmoothed. Any ideas why? I repeated the process above to fit a local quadratic model (`deg=2`). The optimal bandwidth was estimated as $h = 0.1$, and the fitted function is below on the right.



A few additional things about the `locfit` function and package.

- You can use the same process to find an optimal fit using the nearest neighbor fraction instead of the bandwidth. Supplying a sigle value to the argument `alpha` will fit the model based on a given nearest neighbor fraction, e.g., `alpha=.5`.
- The `summary` command will print out some basic information about the fit. For example

```
summary(fit3)

     Estimation type: Local Regression

     Call:  locfit(formula = NOx ~ E, data = ethanol, deg = 2, alpha = c(0,h2.opt))

     Number of data points:  88
     Independent variables:  E
     Evaluation structure: Rectangular Tree
     Number of evaluation points:  17
     Degree of fit:  2
     Fitted Degrees of Freedom:  11.649
```

- Fitted values can be obtained using `fitted(fit3)`.
- Residuals can be obtained using `residuals(fit3)`.
- To plot GCV for different values of h (or nbr fraction) do this

```
plot(gcvplot(NOx~E,data=ethanol, deg=2, alpha=alphamat2))   # x-axis in DF
```

Inferences With Linear Smoothers

The linear smoothers we've studied essentially give us an estimate of the regression function, which we can use to estimate or predict responses. What kind of inferences might we be interested in? How will these differ from inferences in a typical linear model?

A number of procedures may be defined to allow these kind of inferences. But they will require some more assumptions:

Estimating σ^2: Recall the general form of the estimator of σ^2 in linear regression,

We use the same basic estimator here except that the appropriate degrees of freedom isn't quite as obvious. It turns out that if $\boldsymbol{L}$ is the smoothing matrix for the linear smoother then the *equivalent number of parameters* is measured as

$$2\nu - \widetilde{\nu} \qquad \text{where} \qquad \nu = tr(\boldsymbol{L}), \quad \widetilde{\nu} = tr(\boldsymbol{L}^T\boldsymbol{L})$$

By analogy then the estimate of σ^2 is

A model fitted with the `locfit` function contains information about effective degrees of freedom and variance estimation in a component cryptically named `dp`. For example, with our fit of the quadratic model to the emissions data,

```
fit3$dp
  nnalph     fixh    adpen      cut       lk      df1      df2       rv      swt      rsc
 0.00000  0.10000  0.00000  0.80000 -3.48140 12.61434 11.64925  0.09356  0.00000  1.00000

sum(residuals(fit3)^2)
    [1] 6.962802
```

Here `df1` reports ν, `df2` reports $\widetilde{\nu}$ and `rt` reports $\widehat{\sigma}^2$. Verify this variance estimate.

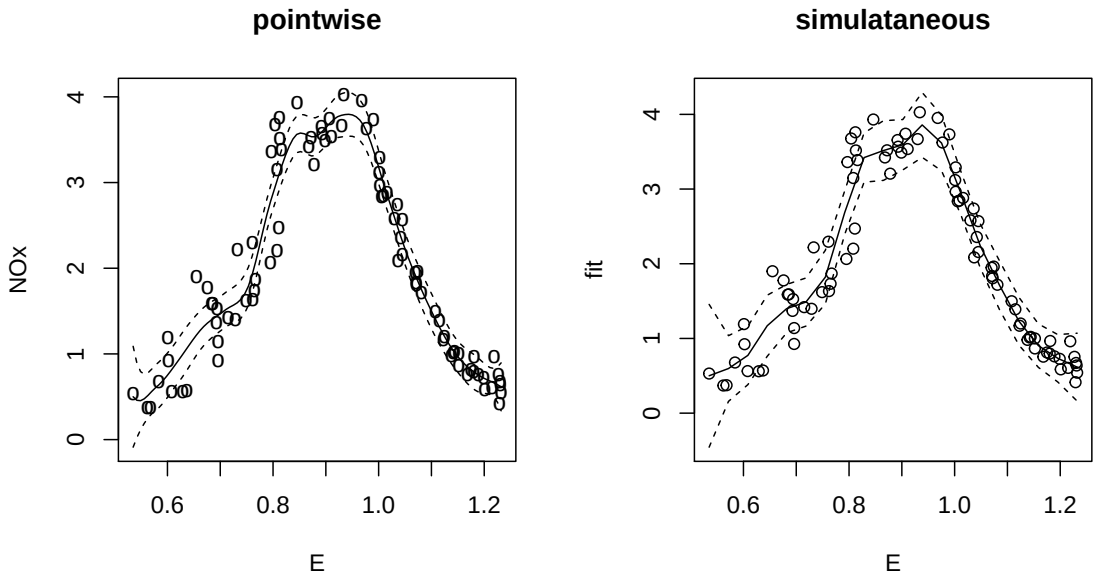
Confidence Bands: Because $\widehat{g}(x)$ is a linear smoother, its variance at a given point x can be expressed as (under our assumption of constant variance)

Thus if errors are assumed normal, a confidence interval for the function at the point x would be given by

Confidence bands are constructed pointwise, by computing the upper and lower limits on the function over a range of x values, and plotting with the estimated function. A couple of things to note.

- Since $\widehat{g}(x)$ is not an unbiased estimator, these bands aren't either. They should be taken as approximations.
- These intervals are really only interpretable as pointwise limits. Not as limits for the function as a whole (think multiple comparisons). A solution to make them more global is to adjust the multiplier $z_{\alpha/2}$. The `kappa0` funciton will return this multiplier. Alternaives, use the `scb` function to compute simultaneous confidence bands as shown below.

```
par(mfrow=c(1,2))
plot(fit3,get.data=T,band="global")  #global specifies use variance est sigma.hat^2
title("pointwise")
new.fit3=scb(NOx~E, deg=2, alpha=c(0,h2.opt), data=ethanol)
plot(new.fit3)
title("simulataneous")
```



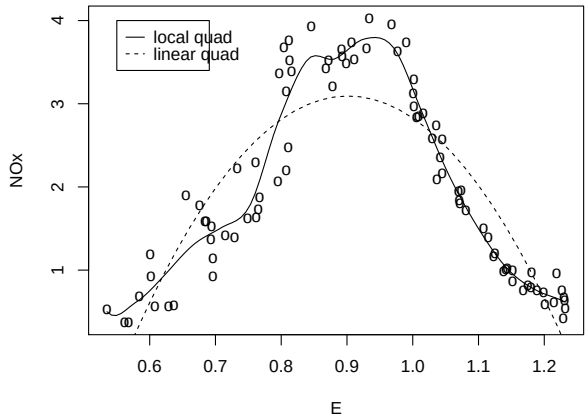
Hypothesis Testing

It is straightforward to define hypothesis test procedures for compaing nested models, when those models include models fit with linear smoothers. First recall the general F-test procedure for model comparisons.

Under the assumptions of normal regression errors with constant variance, this same F-test extends to our nonparametric regression models. Why? What is the nesting structure?

Example: For the `ethanol` data set, let's compare the fit of the local quadratic model to the regular linear quadratic model. The hypotheses are:

Here's a plot.



Below are the fits and some summaries. Compute the F statistic. Recall the degrees of freedom for the local polynomial model from page 14.

```
fit3=locfit(NOx~E, deg=2, alpha=c(0,h2.opt), data=ethanol)
lm.quad=lm(NOx~poly(E, deg=2), data=ethanol)
sse.loc=sum(residuals(fit3)^2)
sse.lin=sum(residuals(lm.quad)^2)

sse.loc
    [1] 6.962802
sse.lin
    [1] 24.71015
```

```
pf(14.55,10.6,74.42,lower.tail=F)
  [1] 2.819543e-14
```