The next weeks

March 9	§10.6 Overdispersion and quasi-likelihood, GEEs
March 16	§10.7 Semiparametric models
March 23	Generalized additive models and lasso
March 30	Finishing pieces, + review

Homework 3: due April 2, 5 pm

Final Test: April 17, 1 - 3 pm – posted by Monday, March 19

marks due 20th

Estimating functions and quasi-likelihood

- suppose we assume only that
  $E(Y_j) = \mu_j(\beta), \quad \mathrm{Var}(Y_j) = \phi a_j V(\mu_j)$, as in most glm's
- and we use the glm estimates of β, defined by the score equation

$$\sum_{j=1}^{n} \frac{y_j - \mu_j}{a_j V(\mu_j)} \frac{x_{jr}}{g'(\mu_j)} = 0 \quad (*)$$

- n.b. Davison calls LHS $g(Y;\beta)$, different g

$g(Y;\tilde{\beta}) = 0$

- using only (*), we have

$$E\{g(Y;\beta)\} = 0; \qquad E\{-\frac{\partial g(Y;\beta)}{\partial \beta}\} = \mathrm{Var}\{g(Y;\beta)\}$$

- thus g has two properties in common with the the score function from a log-likelihood

... estimating functions and quasi-likelihood

$E(.) = 0$

- $$g(Y;\beta) = \sum_{j=1}^{n} \frac{Y_j - \mu_j}{a_j V(\mu_j)} \frac{x_{jr}}{g'(\mu_j)} = 0$$
- ϕ $g(Y;\beta)$ is the β-derivative of
  $$Q(\beta;Y) = \sum_{j=1}^{n} \int_{Y_j}^{\mu_j} \frac{Y_j - \mu}{\phi a_j V(\mu)} du,$$
  u, u
  $\partial Q / \partial \beta = 0$
- $w_j = 1/\{g'(\mu_j)^2 \phi a_j V(\mu_j)\}$ as usual, have assumed
- Quasi-likelihood estimator $\tilde\beta \dot\sim N(\beta, (X^T W X)^{-1})$, where $W =$ $\mathrm{diag}(w_1, \ldots, w_n)$
- inflate estimated standard errors for $\tilde\beta_j$ by $\hat\phi^{1/2}$
- relative lik.
  $$\hat\phi = \frac{1}{n-p} \sum_{j=1}^{n} \frac{(y_j - \hat\mu_j)^2}{a_j V(\hat\mu_j)}$$
- F-test to compare nested models

... estimating functions and quasi-likelihood

```
> toxo.glm1 = glm(cbind(r,m-r) ~ rain + I(rain^2) + I(rain^3),
family = quasibinomial)
> summary(toxo.glm1)
...
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -2.902e+02  1.215e+02  -2.388   0.0234 *
rain         4.500e-01  1.876e-01   2.398   0.0229 *
I(rain^2)   -2.311e-04  9.616e-05  -2.404   0.0226 *
I(rain^3)    3.932e-08  1.635e-08   2.405   0.0225 *
...
> (74.212 - 62.635)/3/1.94
[1] 1.989175
> pf(.Last.value, 3, 30, lower.tail = F)
[1] 0.1368155
```

$\hat{\phi}$

Null dev.   resid dev. $\hat{\phi} = 1.94$

$\tilde{\beta} \xrightarrow{d} N(\beta, \cdot)$

... estimating functions and quasi-likelihood

- even if $V(\mu)$ is incorrectly specified, $\tilde{\beta}$ is still consistent

$E\left(-\frac{\partial g}{\partial \beta}\right)$ $\qquad$ $E\left(-\frac{\partial g}{\partial \beta}\right)$

- $$\text{a.Var}(\tilde{\beta}) = (X^T W X)^{-1} \underline{\text{Var}\{g(Y;\beta)\}} \underline{(X^T W X)^{-1}}$$ ← if right = I

- often is well approximated by $(X^T W X)^{-1}$ in any case

- when extended to dependent data, called generalized estimating equation method

- reference: Liang & Zeger (1986, Biometrika)

Sandwich est. of variance

Example: estimation of spatial intensity (Yongtao Guan, Mar. 15)

- Poisson process on spatial area W, indexed by spatial locations s — (long, lat)
- $N(s)$ counts "events" at location s, $N(s) \sim Pois\{\lambda(s)\}$
  (s, s+ds)
- generalized linear model $\lambda(s) = \exp\{Z(s)^T\beta\}$
  species
  elev
- introduce correlation by assuming two points s_1 and s_2 have a joint intensity function
  soil
- $\lambda_2(s_1, s_2) = \lambda(s_1)\lambda(s_2)g(||s_1 - s_2||)$
- estimation using mean $\lambda(\cdot)$ and variance $\lambda_2(\cdot,\cdot)$ only as in GEE

Generalized estimating equations $\phi V(\mu_j)\alpha_j$

- $Y_j = (Y_{j1}, \ldots, Y_{jn_j})$; $\quad E(Y_j) = \mu_j$; $\quad \mathrm{var}(Y_j) =$ $V(\mu_j; \alpha)$

n ind't Y_j's

$n_j \times n_j$ matrix

- estimating equation for β:

(canonical link)

$$\sum_{j=1}^{n} \left(\frac{\partial \mu_j}{\partial \beta^T} \right) V(\mu_j; \alpha)^{-1}(y_j - \mu_j) = \underset{p\times 1}{0}$$

X^T

GLM type eq =

- multivariate version of quasi-likelihood equation
- needs some specification of $V(\cdot;\cdot)$ called "working covariance matrix"
- `gee in library(gee) offers several choices: independent, exchangeable,` $AR(p)$, `etc.`
- $\ast$ estimate of β is consistent, even if $V(\cdot;\cdot)$ is mis-specified
- but estimates of $\mathrm{Var}(\tilde{\beta})$ will be incorrect if V is "
- there is no quasi-likelihood that corresponds to this more general model

Dependence through random effects

Ch 9 $\quad Y_j \in \mathbb{R}$

- Example: longitudinal data
- $Y_j = (Y_{j1}, \dots, Y_{jn_j})$ vector of observations on jth individual
- recall random effects model (normal theory):

$$\underline{Y_j = X_j\beta + Z_jb_j + \epsilon_j}; \quad \underbrace{b_j \sim N(0, \sigma^2\Omega_b)}_{q}, \underbrace{\epsilon_j \sim N(0, \sigma^2\Omega_j)}_{n_j}$$

- marginal distribution:

$$Y_j \sim N(X_j\beta, \sigma^2\Upsilon_j^{-1})$$

Y Ch 9

$(Y_1, \dots, Y_n)$

- sample of n i.i.d. such vectors leads to

$$Y \sim N(X\beta, \sigma^2\Upsilon^{-1}), \quad \Upsilon^{-1} = (\Omega + Z\tilde{\Omega}_bZ^T)$$

- $\Omega = \mathrm{diag}(\Omega_1, \dots, \Omega_n), \quad \tilde{\Omega}_b = \mathrm{diag}(\Omega_b, \dots, \Omega_b),$
- $\Upsilon^{-1} = \mathrm{diag}(\Upsilon_1^{-1}, \dots, \Upsilon_n^{-1})$

Generalized linear mixed models

- simplify as in last slide to canonical link $\boxed{(\theta_j = \eta_j)}$
- $$f(y_j \mid \theta_j, \phi) = \exp\{\frac{y_j\theta_j - b(\theta_j)}{\phi a_j} + c(y_j; \phi a_j)\}$$
- random effects
$$\theta_j = \underline{x_j^T}\beta + \underline{z_j^T}b, \quad \underline{b \sim N(0, \Omega_b)}$$
- likelihood
$$L(\beta, \phi; y) = \prod_{j=1}^{n} \int f(y_j \mid \beta, b, \phi) f(b; \Omega_b) db$$

... generalized linear mixed models

- likelihood

simplify

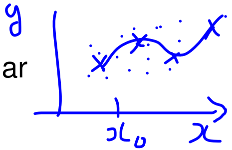
$$L(\beta, \phi; y) = \prod_{j=1}^{n} \int f(y_j \mid \beta, b, \phi) f(b; \Omega_b) db$$

$\prod_{kk'} f_2(y_{jk}, y_{jk'})$

(composite)

- doesn't simplify unless $f(\cdot)$ is normal
- solutions proposed include
  - numerical integration, e.g. by quadrature
  - integration by MCMC
  - penalized quasi-likelihood – use Laplace approximation to the integral
- reference: `MASS` library and book (§10.4): `glmmNQ`, `GLMMGibbs`, `glmmPQL`, all in `library(MASS)` `glmer` in `library(lme4)`
- see also Faraway (*Extending the Linear Model with* R), Ch. 10

Semiparametric Regression §10.7



- model $y_j = g(x_j) + \epsilon_j, \quad j = 1, \ldots, n \quad x_j$ scalar
- mean function $g(\cdot)$ assumed to be "smooth"

$E(y \mid x) = g(x)$

- introduce a kernel function $w(u)$ and define a set of weights

$$w_j = \frac{1}{h} w\left(\frac{x_j - x_0}{h}\right)$$

1) $w(u) = \phi(u) = \dfrac{e^{-\frac{1}{2}u^2}}{\sqrt{2\pi}}$

- estimate of $g(x_0)$:

$$\hat{g}(x_0) = \frac{\sum_{j=1}^{n} w_j y_j}{\sum_{j=1}^{n} w_j}$$

- Nadaraya-Watson estimator (10.40) – local averaging

(tricube) 2) $(1 - |u|^3)^3 \quad |u| \leq 1$

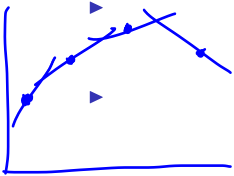
... kernel smoothing

- better estimates can be obtained using local regression at point x

$\frac{\sum w_i y_i}{\sum w_i}$

-

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} 1 & (x_1 - x_0) & \cdots & (x_1 - x_0)^k \\ \vdots & \vdots & & \vdots \\ 1 & (x_n - x_0) & \cdots & (x_n - x_0)^k \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_k \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix},$$



- $$\hat{\beta} = (X^T W X)^{-1} X^T W y$$

$W = \mathrm{diag}(w_j)$

- $$\hat{g}(x_0) = \hat{\beta}_0$$

$w_j = \frac{1}{h} w\left(\frac{x_0 - x_j}{h}\right)$

- usually obtain estimates $\hat{g}(x_j), j = 1, \ldots, n$

... kernel smoothing

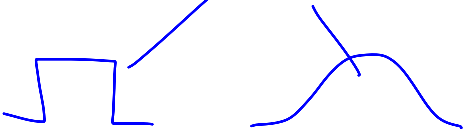
- odd-order polynomials work better than even; usually local linear fits are used
- kernel function is often a Gaussian density, or the tricube function (10.37) h
- choice of bandwidth controls smoothness of function
- kernel estimators are biased
- larger bandwidth = more smoothing – increases bias, decreases variance
- some smoothers allows variable bandwidth depending on density of observations near x_0
- `ksmooth` computes local averages; `loess` computes local linear regression (robustified)

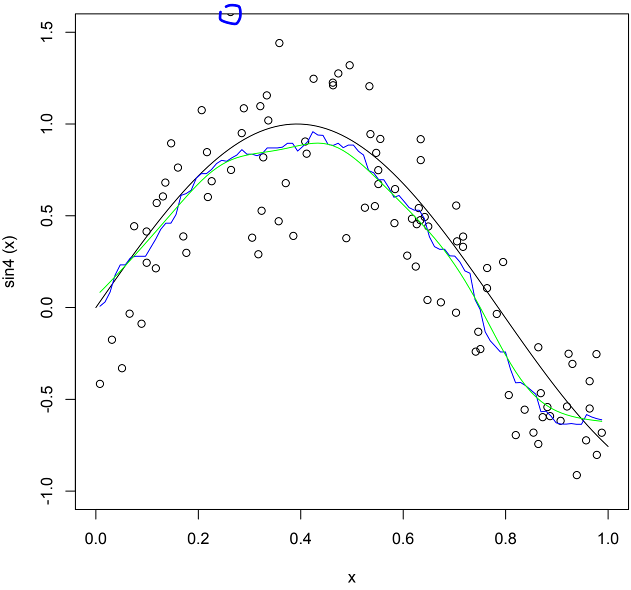
$$E\hat{g}(x_0) \neq g(x_0)$$

Example: weighted average

```
ksmooth(x,y,kernel=c("box","normal"),bandwidth=0.5,
         range.x=range(x),
         n.points=max(100,length(x)), x.points)
```

```
> eps<-rnorm(100,0,1/3)
> x<-runif(100)
> sin4 <- function(x){sin(4*x)}
> y<-sin4(x)+eps
> plot(sin4,0,1,type="l",ylim=c(-1.0,1.5),xlim=c(0,1))
> points(x,y)
> lines(ksmooth(x,y,"box",bandwidth=.2),col="blue")
> lines(ksmooth(x,y,"normal",bandwidth=.2),col="green")
```

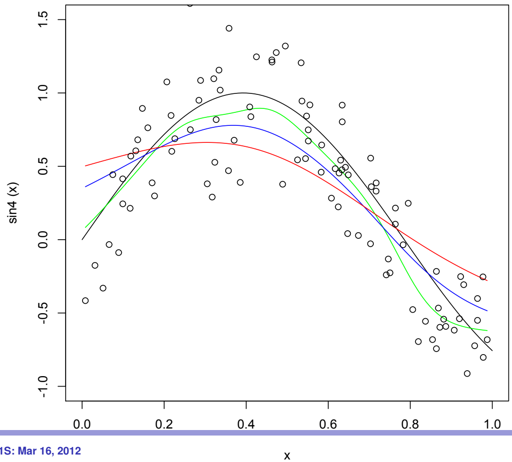




1.5
1.0
0.5
0.0
-0.5
-1.0
sin4 (x)
0.0
0.2
0.4
0.6
0.8
1.0
x

... Example

```
> plot(sin4,0,1,type="l",ylim=c(-1.0,1.5),xlim=c(0,1))
> lines(ksmooth(x,y,"normal",bandwidth=.2),col="green")
> lines(ksmooth(x,y,"normal",bandwidth=0.4),col="blue")
> lines(ksmooth(x,y,"normal",bandwidth=0.6),col="red")
```



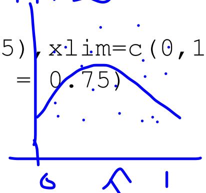
Fitting in R

- `scatter.smooth` fits a `loess` curve to a scatter plot
- `loess` takes a `family` argument : `family = gaussian` gives weighted least squares using K_λ as weights and `family=symmetric` gives a robust version using Tukey's biweight
- `supsmu` implements "Friedman's super smoother": a running lines smoother with elaborate adaptive choice of bandwidth
- Library `KernSmooth` has `locpoly` for local polynomial fits, and by setting `degree = 0` gives a kernel smooth
- as usual more smoothing means larger "bias", smaller "variance"

$$w(u) = e^{-\frac{1}{2}u^2} / \sqrt{2\pi}$$

Example: local linear smoothing

MASS



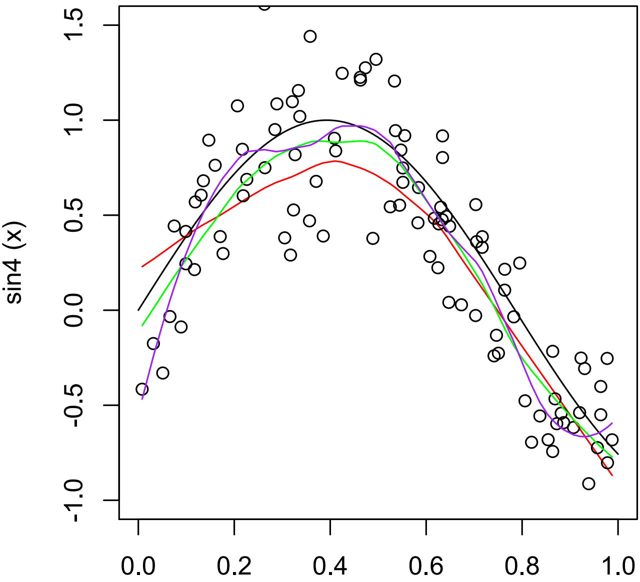

```
> plot(sin4,0,1,type="l",ylim=c(-1,1.5),xlim=c(0,1), xla
> lo1 = loess(y ~ x, degree = 1, span = 0.75)

> attributes(lo1)
$names
 [1] "n"           "fitted"      "residuals" "enp"         "s"
 [7] "two.delta"   "trace.hat"   "divisor"   "pars"        "kd
[13] "terms"       "xnames"      "x"         "y"           "we

$class
[1] "loess"
```

runif(0,1)

(ord = ?)

ord = order(lo1$x)

```
> lines(lo1$x[ord],lo1$fitted[ord],col="red")
> lo2 = loess(y~x, degree=1, span=0.4)
> lo3 = loess(y~x, degree=2, span=0.4)
> lines(lo1$x[ord],lo2$fitted[ord],col="green")
> lines(lo1$x[ord],lo3$fitted[ord],col="purple")
```



sin4 (x)
1.5
1.0
0.5
0.0
-0.5
-1.0
0.0
0.2
0.4
0.6
0.8
1.0

Scatter Plot with Smooth Curve Fitted by Loess

Description

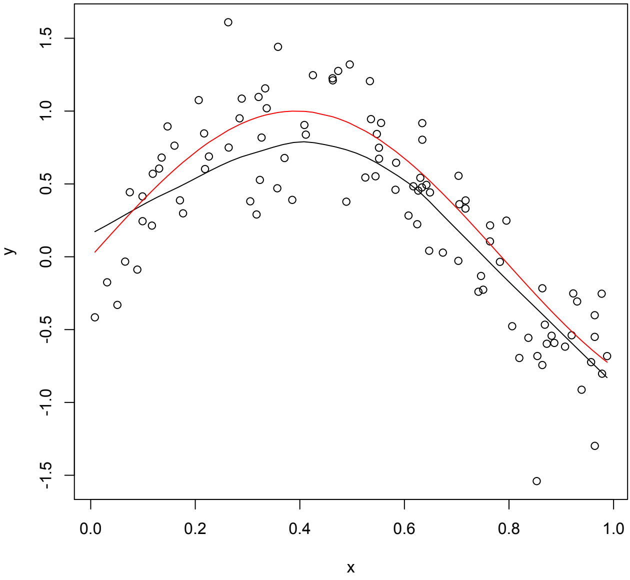
Plot and add a smooth curve computed by `loess` to a scatter plot.

Usage

```
scatter.smooth(x, y = NULL, span = 2/3, degree = 1,
    family = c("symmetric", "gaussian"),
    xlab = NULL, ylab = NULL,
    ylim = range(y, prediction$y, na.rm = TRUE),
    evaluation = 50, ...)

loess.smooth(x, y, span = 2/3, degree = 1,
    family = c("symmetric", "gaussian"), evaluation = 50, ...)
```

Arguments

`x,y`	the `x` and `y` arguments provide the x and y coordinates for the plot. Any reasonable way of the coordinates is acceptable. See the function `xy.coords` for details.
`span`	smoothness parameter for `loess`.
`degree`	degree of local polynomial used.
`family`	if `"gaussian"` fitting is by least-squares, and if `family="symmetric"` a re-descending M is used.
`xlab`	label for x axis.
`ylab`	label for y axis.



y
x
1.5
1.0
0.5
0.0
-0.5
-1.0
-1.5
0.0
0.2
0.4
0.6
0.8
1.0

Friedman's SuperSmoother

Description

Smooth the (x, y) values by Friedman's 'super smoother'.

Usage

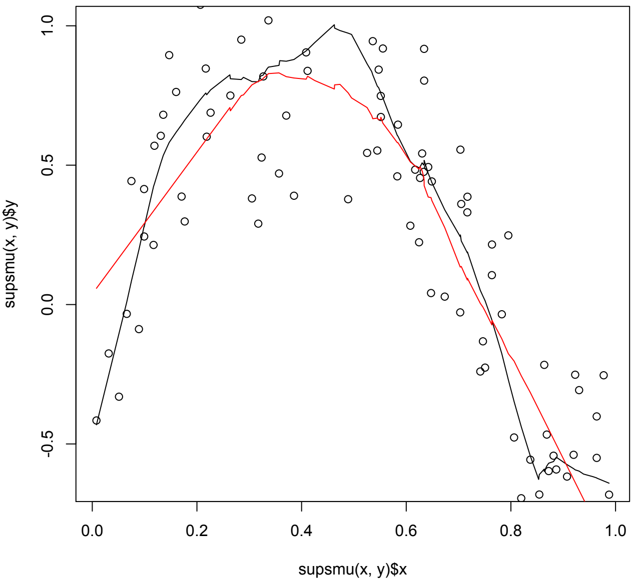
```
supsmu(x, y, wt, span = "cv", periodic = FALSE, bass = 0)
```

Arguments

`x`	x values for smoothing
`y`	y values for smoothing
`wt`	case weights, by default all equal
`span`	the fraction of the observations in the span of the running lines smoother, or `"cv"` to choose th leave-one-out cross-validation.
`periodic`	if `TRUE`, the x values are assumed to be in `[0, 1]` and of period 1.
`bass`	controls the smoothness of the fitted curve. Values of up to 10 indicate increasing smoothness.

Details

`supsmu` is a running lines smoother which chooses between three spans for the lines. The running lines sr are symmetric, with `k/2` data points each side of the predicted point, and values of `k` as `0.5 * n`, `0.2 *` `0.05 * n`, where `n` is the number of data points. If `span` is specified, a single smoother with span `span *` used.


1.0
0.5
0.0
-0.5
supsmu(x, y)$y
0.0
0.2
0.4
0.6
0.8
1.0
supsmu(x, y)$x

Inference from smooth functions

$\left(\frac{10.39}{n - 2\nu_1 + \nu_2}\right)$

- $\hat\beta = (X^T WX)^{-1} X^T Wy$ (x_0)
- $W = \text{diag}(w_1, \ldots, w_n)$ $w_j = \frac{1}{h} w\left(\frac{x_0 - x_j}{h}\right)$
- $\hat g(x_0) = \sum_{j=1}^n S(x_0; x_j, h) y_j \equiv \hat\beta_0$ use $\hat\beta_0$ from regn
- $S(x_0; x_1, h), \ldots, S(x_0; x_n, h)$ first row of "hat" matrix $(X^T WX)^{-1} X^T W$
- $E\{\hat g(x_0)\} = \sum_{j=1}^n S(x_0; x_j, h) g(x_j) \neq g(x_0)$
- $\text{var}\{\hat g(x_0)\} = \sigma^2 \sum_{j=1}^n S(x_0; x_j, h)^2$ $S_h = n \times n$

$\hat\beta = Hy$

- similarly $\hat g = (\hat g(x_1), \ldots, \hat g(x_n)) = \mathcal{S}_h y$ $S^h_{ij} = S(x_i; x_j, h)$

$p \simeq 2\nu_1 - \nu_2$

- $\nu_1 = \text{tr}(S_h)$, $\nu_2 = \text{tr}(S_h^T S_h)$ suggested as $H^T H = I$

omit y_j

$y_{(-j)} \sim g(x_{(\cdot j)}) \rightarrow \hat{g} \rightarrow \hat{y}_j$ $\Sigma(\hat{y}_j - y_j)^2 \leftarrow$

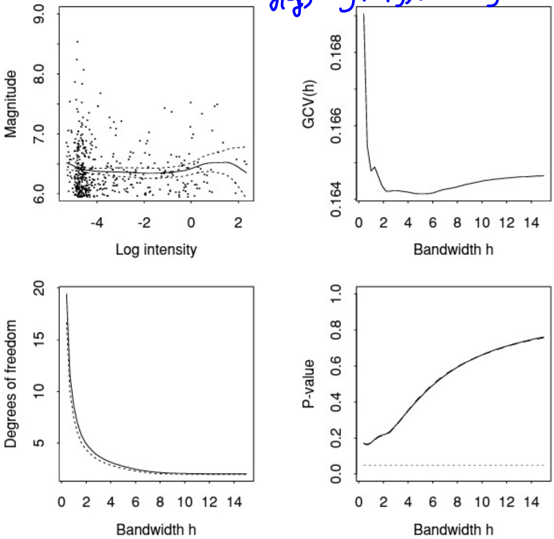


Figure 10.16 Smooth analysis of earthquake data. Upper left: local linear regression of magnitude on log intensity just before quake (solid), with 0.95 pointwise confidence bands (dots). Upper right: generalized cross-validation criterion GCV(h) as a function of bandwidth h. Lower left: relation between degrees of freedom ν_1 (solid), ν_2 (dots), and h. Lower right: significance traces for test of no relation between magnitude and log intensity, based on chi-squared approximation (dots) and saddlepoint approximation (solid). The horizontal line shows the conventional 0.05 significance level.

Extension

- original model $y_j = g(x_j) + \epsilon_j$ ← gen LS
- extend to $y_j \sim f(\cdot; \beta, x_j)$ gen GLM
-

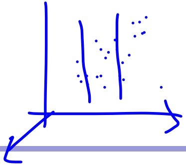
$$\max_\beta \sum \log f(y_j; \beta, x_j) \longrightarrow \max_\beta \sum \frac{1}{h} w\left(\frac{x_j - x_0}{h}\right) \log f(y_j; \beta, x_j)$$

- local likelihood fitting ← local likelihood
- more than 1 covariate $g_2(x_{1j}, x_{2j})$
- $E(Y_j) = g_1(x_{1j}) + g_2(x_{2j}) + \cdots + g_p(x_{pj})$
- or

additive models

GAM $h(EY_j) =$

log logit …



Example 10.32

528 $\quad$ *10 · Nonlinear Regression Models*

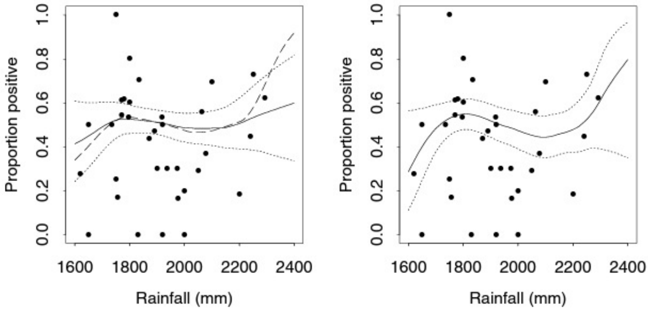


Figure 10.17 Local fits to the toxoplasmosis data. The left panel shows fitte probabilities $\widehat{\pi}(x)$, with the fit of local linear logistic model with $h = 400$ (solid) and 0.95 pointwise confidence bands (dots). Also shown is the local linear fit with $h = 300$ (dashes). The right panel shows the loc quadratic fit with $h = 40$ and its 0.95 confidence band. Note the increased variability due to the quadratic fit, and its stronger curvature at the boundaries.

Flexible modelling using basis expansions

(§10.7.2)

- $y_j = g(x_j) + \epsilon_j$

- Flexible linear modelling

$$g(x) = \Sigma_{m=1}^{M} \beta_m h_m(x)$$

- This is called a linear basis expansion, and h_m is the mth basis function
- For example if X is one-dimensional:
  $g(x) = \beta_0 + \beta_1 x + \beta_2 x^2$, or
  $g(x) = \beta_0 + \beta_1 \sin(x) + \beta_2 \cos(x)$, etc.
- Simple linear regression has $h_1(x) = 1$, $h_2(x) = x$

Piecewise polynomials

- piecewise constant basis functions
  $h_1(x) = I(x < \xi_1), \quad h_2(x) = I(\xi_1 \le x < \xi_2),$
  $h_3(x) = I(\xi_2 \le x)$
- fitting by local averaging

- piecewise linear basis functions , with constraints
  $h_1(x) = 1, \quad h_2(x) = x$
  $h_3(x) = (x - \xi_1)_+, \quad h_4(x) = (x - \xi_2)_+$
- windows defined by knots $\xi_1, \xi_2, \ldots$

- piecewise cubic basis functions
  $h_1(x) = 1, h_2(x) = x, h_3(x) = x^2, h_4(x) = x^3$

- continuity $h_5(x) = (x - \xi_1)^3_+, \quad h_6(x) = (x - \xi_2)^3_+$

- continuous function, continuous first and second derivatives

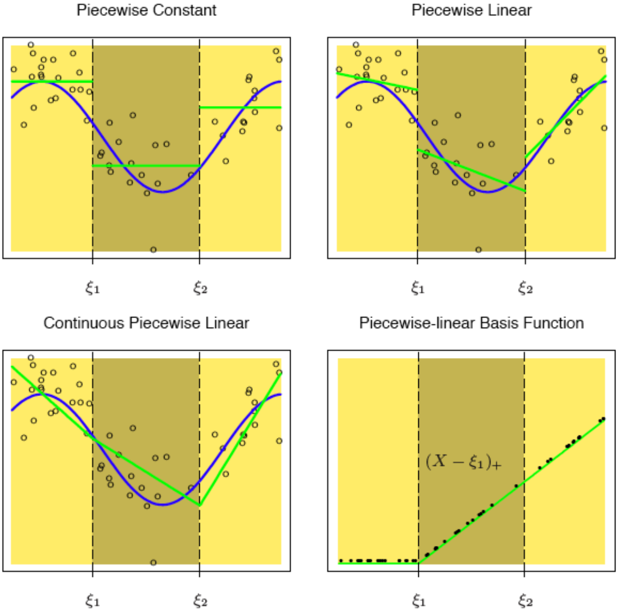


FIGURE 5.1. *The top left panel shows a piecewise constant function fit to some artificial data. The broken vertical lines indicate the positions of the two* knots *ξ_1 and ξ_2. The blue curve represents the true function, from which the data were*

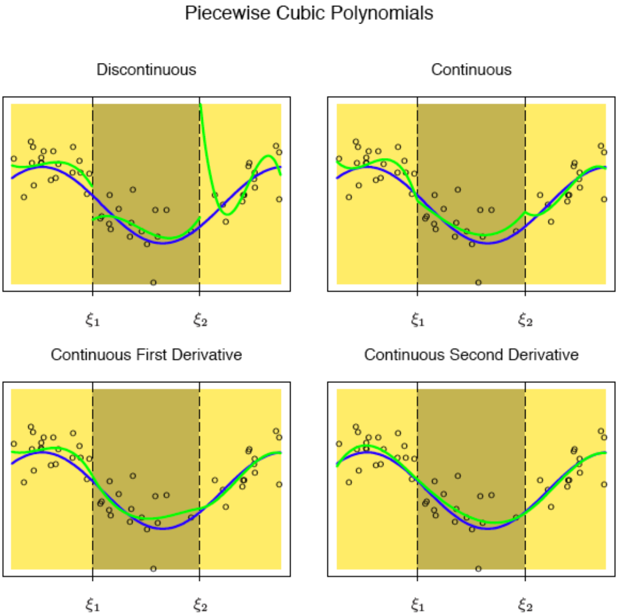


FIGURE 5.2. *A series of piecewise-cubic polynomials, with increasing orders of continuity.*