4.0 - Smoothing

4.1 - Nonparametric regression

Nonparametric regression model

Parametric regression models are usually of the form

$$Y_i = g(\theta, X_i) + \varepsilon_i, \qquad i = 1, \ldots, N$$

where g and the X_i's are specified (i.e. "known"), and the additive noise term characterized, e.g. $\varepsilon \overset{iid}{\sim} \mathcal{N}(0, \sigma^2)$. The problem is then to estimate θ.

Nonparametric regression consists in leaving g unspecified; the problem becomes that of estimating g in

$$Y_i = g(X_i) + \varepsilon_i, \qquad i = 1, \ldots, N$$

Various forms may be considered for $g(x)$, such as additive regression models

$$g(x) = \alpha + g_1(x_1) + g_2(x_2) + \cdots + f_k(x_k)$$
$$g(x) = \alpha + \theta_1 x_1 + g_2(x_2) + \cdots + f_k(x_k)$$
$$g(x) = \alpha + g_{12}(x_1, x_2) + g_3(x_3) + \cdots + f_k(x_k)$$

g is often assumed to be continuous and smooth.

Local polynomial regression, **smoothing splines** and **(Nadaraya-Watson) kernel regression** are among the most popular "smoothing" techniques available to estimate the shape of g.

Local polynomial regression: simple regression

Consider the **simple regression problem** of estimating g in

$$Y_i = g(X_i) + \varepsilon_i$$

A p-th order weighted least squares (WLS) polynomial regression of observations $y = (y_1, \ldots, y_N)^T$ on design points $x = (x_1, \ldots, x_N)$ evaluated at a given point x_0 yields the approximation

$$y_i^0 = \alpha + \theta_1(x_i - x_0) + \theta_2(x_i - x_0)^2 + \cdots + \theta_p(x_i - x_0)^p$$

Using WLS, the weights are usually chosen to account for the distance between y_i and the reference design point x_0, using e.g. the tricube function

$$W(z) = \begin{cases} (1 - |z|^3)^3 & \text{for } |z| < 1 \\ 0 & \text{for } |z| \geq 1 \end{cases}$$

with $z_i^0 = \frac{x_i - x_0}{h}$ and h is a control parameter that is picked according to the scale of the data.

```
library(car)
attach(Prestige)
plot(income, prestige, xlab="Average income",
        ylab="Prestige")
lines(lowess(income, prestige, f=0.25, iter=0),
        lwd=2, col='blue')
lines(lowess(income, prestige, f=0.5, iter=0),
        lwd=2, col='red')
lines(lowess(income, prestige, f=1, iter=0),
        lwd=2, col='green')
```

degree of smoothing: too jittery
degree of smoothing: about right
degree of smoothing: too smooth
weight control: "do not refit to address outliers" (ref [Cox2002])

Local polynomial regression: multiple regression

Consider the **multiple regression problem** of estimating g in

$$Y_i = g(X_{i1}, \ldots, X_{ik}) + \varepsilon_i$$

Local polynomial regression consists in fitting a weighted polynomial, e.g. of the linear form at a given point $x_0 = (x_{01}, \ldots, x_{0k})$:

$$y_i^0 = \alpha + \theta_1(x_{i1} - x_{01}) + \theta_2(x_{i2} - x_{02})^2 + \cdots + \theta_k(x_{ik} - x_{0k})^p + \varepsilon_i$$

This approach requires choosing a distance metric and a weighting scheme for multivariate observations.

The default distance metric is commonly the Euclidean distance

$$D(x_i, x_0) = \left[\sum_{j=1}^{k} (z_{ij} - z_{0j})^2 \right]$$

using standardized values $z_{ij} = \frac{x_{ij} - \bar{x}_j}{s_j}$ (with sample mean and std dev)

The weights are usually defined using scaled distances between y_i and the reference point x_0, e.g.

$$w_i = W\left(\frac{D(x_i, x_0)}{h} \right)$$

where h denotes the bandwidth (e.g. the half-width of the neighbourhood).

```
mod.lo = loess(prestige~income+education,
         span=.5, degree=2)
summary(mod.lo)

# plot this smoothing...
inc <- seq(min(income), max(income), len=25)
ed <- seq(min(education), max(education), len=25)
newdata <- expand.grid(income=inc,education=ed)
fit.prestige <- matrix(predict(mod.lo, newdata), 25, 25)

persp(inc, ed, fit.prestige, theta=45, phi=30,
    ticktype='detailed', expand=2/3, shade=0.5
        xlab='Income', ylab='Education', zlab='Prestige')
```

Nadaraya-Watson regression

In this approach one likes to represent the regression problem based on

$$g(x) = \mathrm{E}(Y_i \mid X_i = x)$$

and to evaluate the importance of the neighbourhood $x \pm h$ around each design point x (for $h > 0$).

Then the average of observations Y_i's yields the following nonparametric estimator for g:

$$\hat{g}(x) = \frac{\sum_{i=1}^{N} \mathbb{1}(|X_i - x| \leq h) Y_i}{\mathbb{1}(|X_i - x| \leq h)} = \frac{\sum_{i=1}^{N} K\left(\frac{X_i - x}{h}\right) Y_i}{K\left(\frac{X_i - x}{h}\right)}$$

using Uniform kernels K.

Note that the estimator may also be defined by plugging $\hat{f}(x, y)$ into

$$g(x) = \frac{\int y f(x, y) dy}{f(x)}$$

This approach extends to the choice of other kernels; the Normal distribution is (once again) very commonly used.

The density estimator used should be well-defined (i.e. $\hat{f}(x) > 0$) for the kernel regression estimator to be well defined.

Kernel regression extends easily to multivariate problems by using multivariate kernel constructs (this is beyond our scope).

Prestige example

```
library(car)
attach(Prestige)

plot(income, prestige)
inc.100 <- seq(min(income), max(income), len=100)

mod.lo.inc <- loess(prestige ~ income, span=.7, degree=1)
pres <- predict(mod.lo.inc, data.frame(income=inc.100))
```

```
# This other implementation works better:
library(KernSmooth)
lp = locpoly(income, prestige, bandwidth=1500)
lines(lp, lwd=2, col='red')
```

car example

```
with(cars, {

    plot(cars$speed, cars$dist)
    ex2 = ksmooth(cars$speed, cars$dist,
            "normal", bandwidth = 2)
    ex5 = ksmooth(cars$speed, cars$dist,
            "normal", bandwidth = 5)

    lines(ex2, lwd=2, col = 2)
    lines(ex5, lwd=2, col = 3)

})
```

Smooth control!?

At this point one ought to wonder: how do we set (or select) the value for h?

This is a very common question for most nonparametric methods (cf. KDE's!)...

Typical approaches are:

- simulation study (using plotting and trial-and-error)
- some type of criterion optimization (usually some likelihood or MSE function)
- cross-validation (typically based on the MSE)

The choice of strategy usually depends on the problem at hand (and on the user's taste!)...

4.2 - Splines

Splines

- Spline = continuous function (i.e. curve) constructed by piecewise linkage of polynomials/functions
- Free splines (Bezier curves): not suitable for mortality curves
- Regression splines: use equidistant knots
- Smoothing splines: penalize roughness, knots are data points [6]

Bronze spline weights

B-splines

- Explain $f(x)$ in terms of a basis (B = 'basis')
- A B-spline is defined by its order m and number of interior knots K (x_0 and x_{K+1} are end-knots)

$$x_0 \leq x_1 \leq \cdots \leq x_K \leq x_{K+1}$$

- Polynomial is of order $m - 1$ (one often picks $m = 4$)
- Univariate construction:

$$S(x) = \alpha_0 + \sum_{j=1}^{J} \alpha_j B_j(x, m)$$

  where $J = K - 1 + m$ is the number of basis functions
- Basis elements are defined in a recursive manner [cf. lit.]
- Fitting: basis coefficients are calculated via linear regression

B-splines: specifications

- Placing knots can be achieved in a number of ways
- One way is to define regular intervals within [xmin,xmax]
- Another way is to define the interior knots as the quantiles from the empirical distribution of the underlying variable
  - enforces an equal number of observations in each interval
  - intervals have different lengths

B-splines: specifications

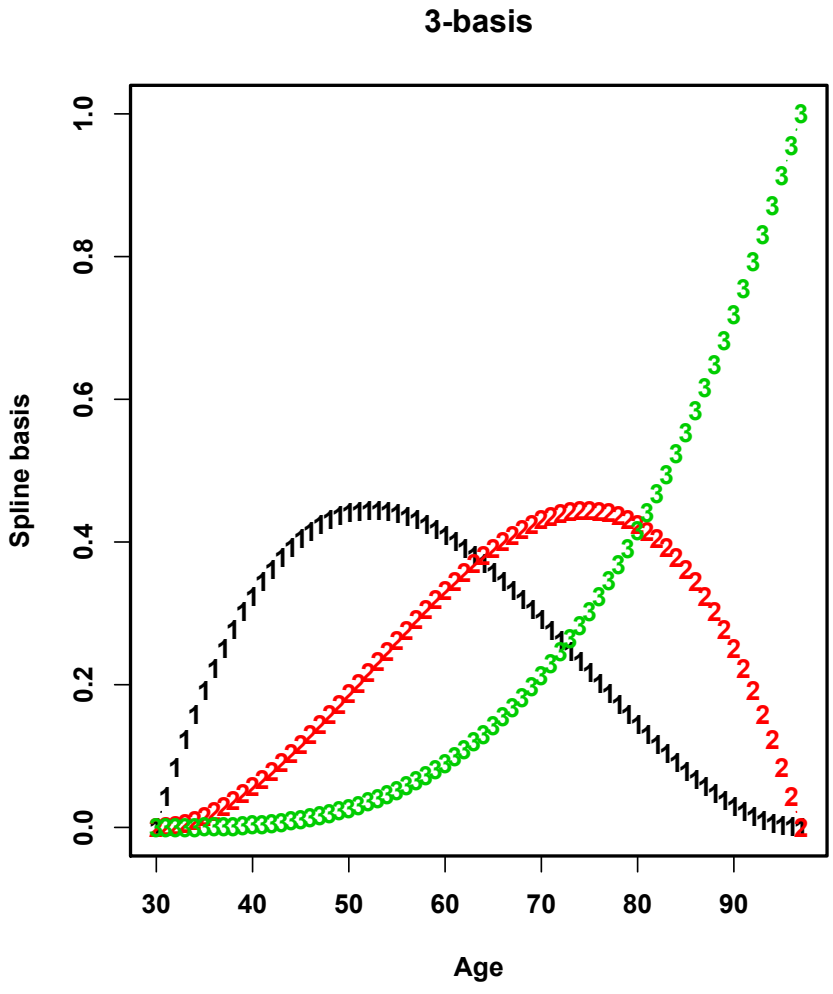
The basis elements (or design matrix) are evaluated at the control points for the specified number of knots. Ex: compare

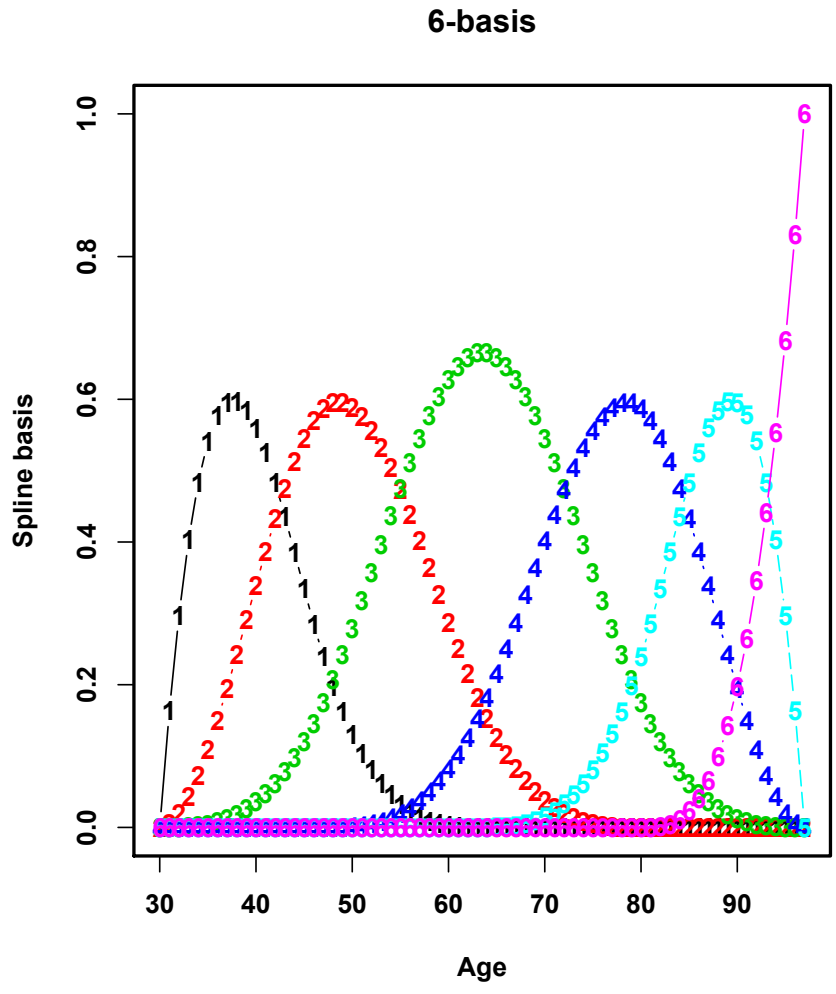
```
require(splines)
bs(dat$Age)
matplot(dat$Age,BM,xlab='Age')
```

with

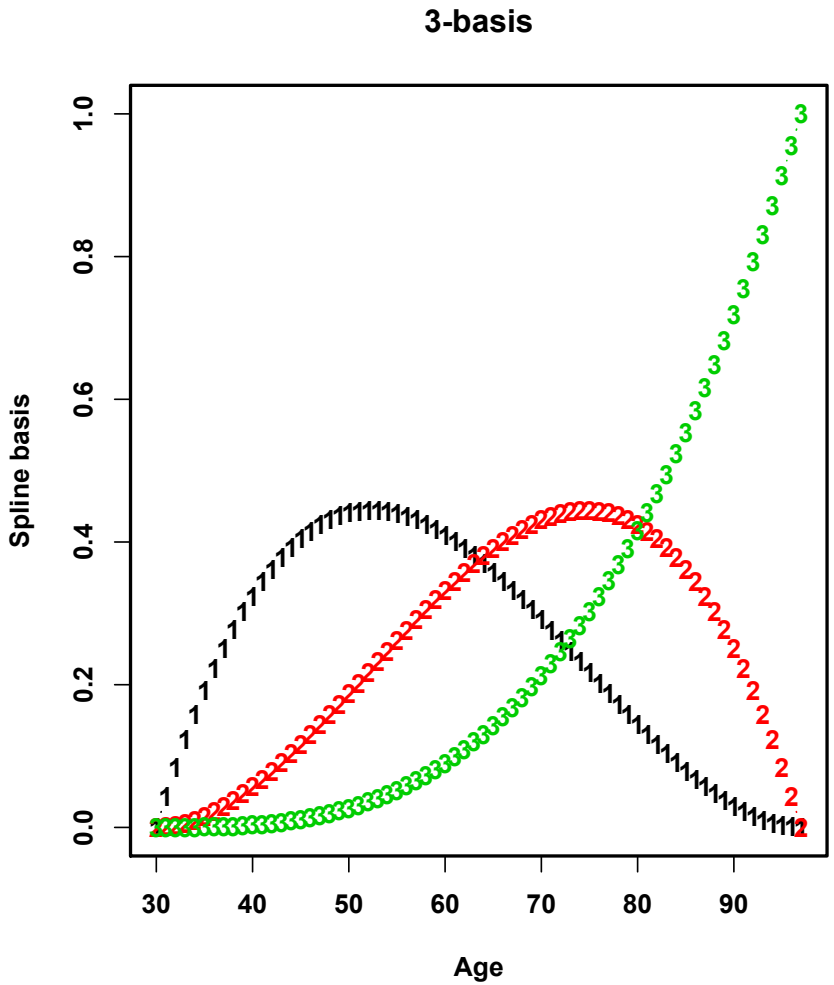
```
xs = seq(min(dat$Age),max(dat$Age),length=1000)
ks = quantile(xs,seq(0.1,0.9,by=.1))
(BM2 = bs(dat$Age,knots=ks))
matplot(dat$Age,BM2,xlab='Age',t='b')
abline(v=ks,lty=2,col=8)
```

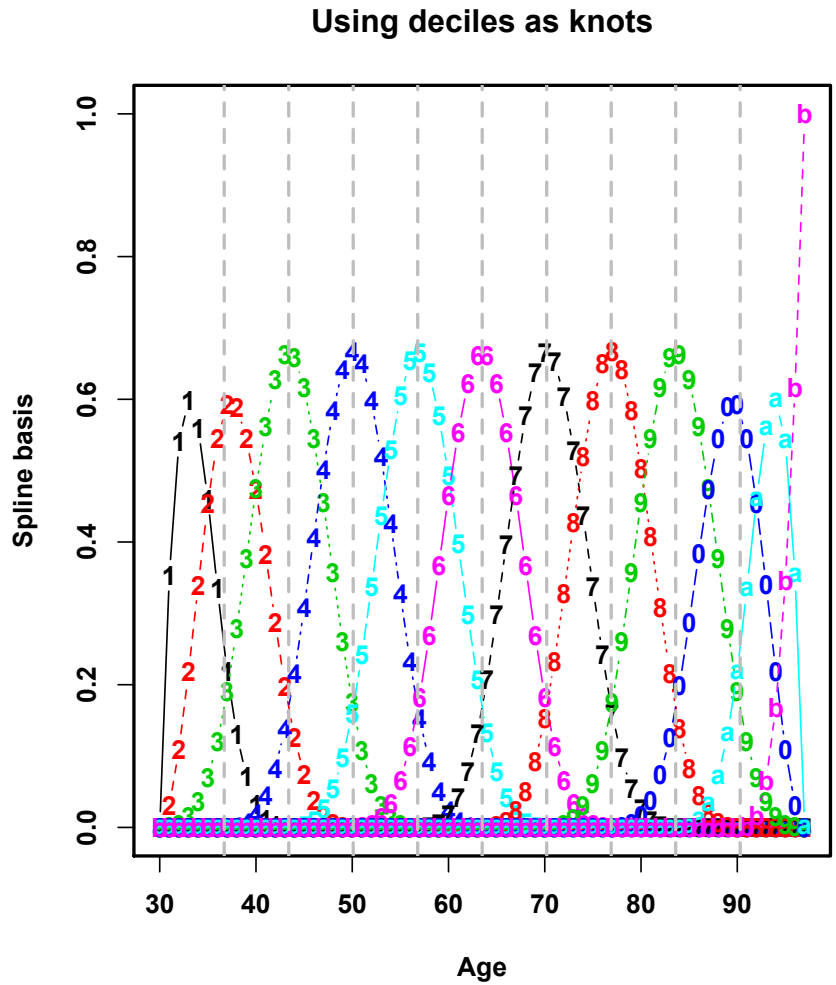
Choice of design matrix (Life dataset)





Choice of design matrix (Life dataset)



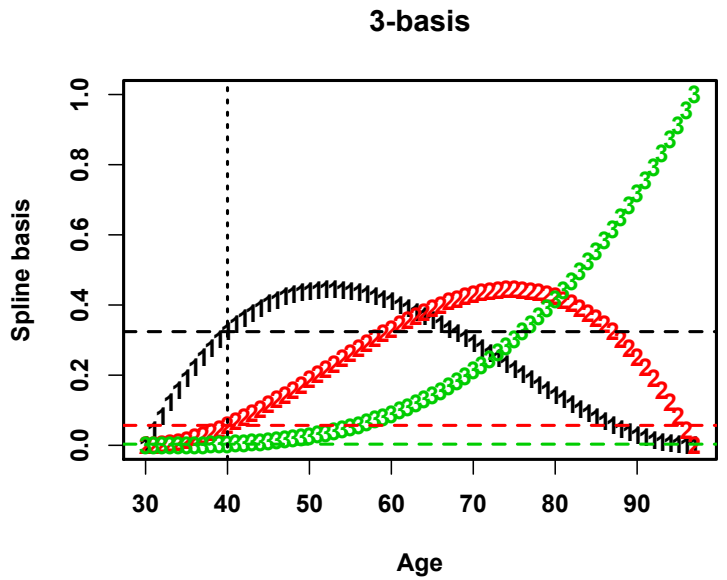


B-spline questions

Question

(a) Compute the design matrix for the Life dataset, and quote the projection of age 40 onto that basis (cf. figure below).

(b) Reconstruct the B-spline approximation to the crude male force data using linear regression. Plot the output over the data points and criticize.



B-spline questions

Question

(a) Continuing from the previous question, compute the B-spline as a power series, i.e. compute the traditional cubic spline representation

$$S(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \sum_{j=1}^{J} \alpha_j (x - x_j)^{3+}$$

where $(x - x_j)^{3+} = max((x - x_j)^3, 0)$, using $J = 3$ interior knots at 40, 60 and 80.

(b) Compare with the fitted values from the regression onto the design matrix obtained from `bs()` for those knots.

P-splines

Splines are piece-wise polynomial functions of the underlying target "response function". The pieces are joined at *knots* $\{X_1, \ldots, X_N\}$ to yield a continuous functional.

A simple smoothing spline $\hat{g}$ is defined as the function that **minimizes** the penalized sum of squares

$$SS_h(g) = \sum_{i=1}^{N} (Y_i - g(X_i))^2 + h \int_{X_{min}}^{X_{max}} \left(g''(u)\right)^2 du$$

defined over a given grid $[X_{min}, \ldots, X_{max}]$.

Smoothing splines

$$SS_h(g) = \sum_{i=1}^{N} (Y_i - g(X_i))^2 + h \int_{X_{min}}^{X_{max}} \left(g''(u)\right)^2 du$$

- P-splines use all data points as knots and penalize for smoothness
- Penalty term is usually controlled by one parameter (h)
- This smoothing parameter h is *controlled*
- Can also define periodic splines

$$SS_h(g) = \sum_{i=1}^{N} (Y_i - g(X_i))^2 + h \int_{X_{min}}^{X_{max}} \left(g''(u)\right)^2 du$$

- The solution (which defines the smoothing spline)

$$\hat{g} = \arg\min_g SS_h(g)$$

  is a cubic spline with knots located at the design points $X_1, \ldots, X_N$.
- The cubic spline approach assumes the existence of two continuous derivatives, which defines a roughness penalty in the integral term: the larger g'', the more jittery g.

```
library(car)
attach(Prestige)

plot(income, prestige)
inc.100 <- seq(min(income), max(income), len=100)

mod.lo.inc <- loess(prestige ~ income, span=.7, degree=1)
pres <- predict(mod.lo.inc, data.frame(income=inc.100))

ssp <- smooth.spline(income, prestige, df=3.85)

lines(inc.100, pres, lty=2, lwd=2)
lines(ssp, lwd=2, col='blue')
```

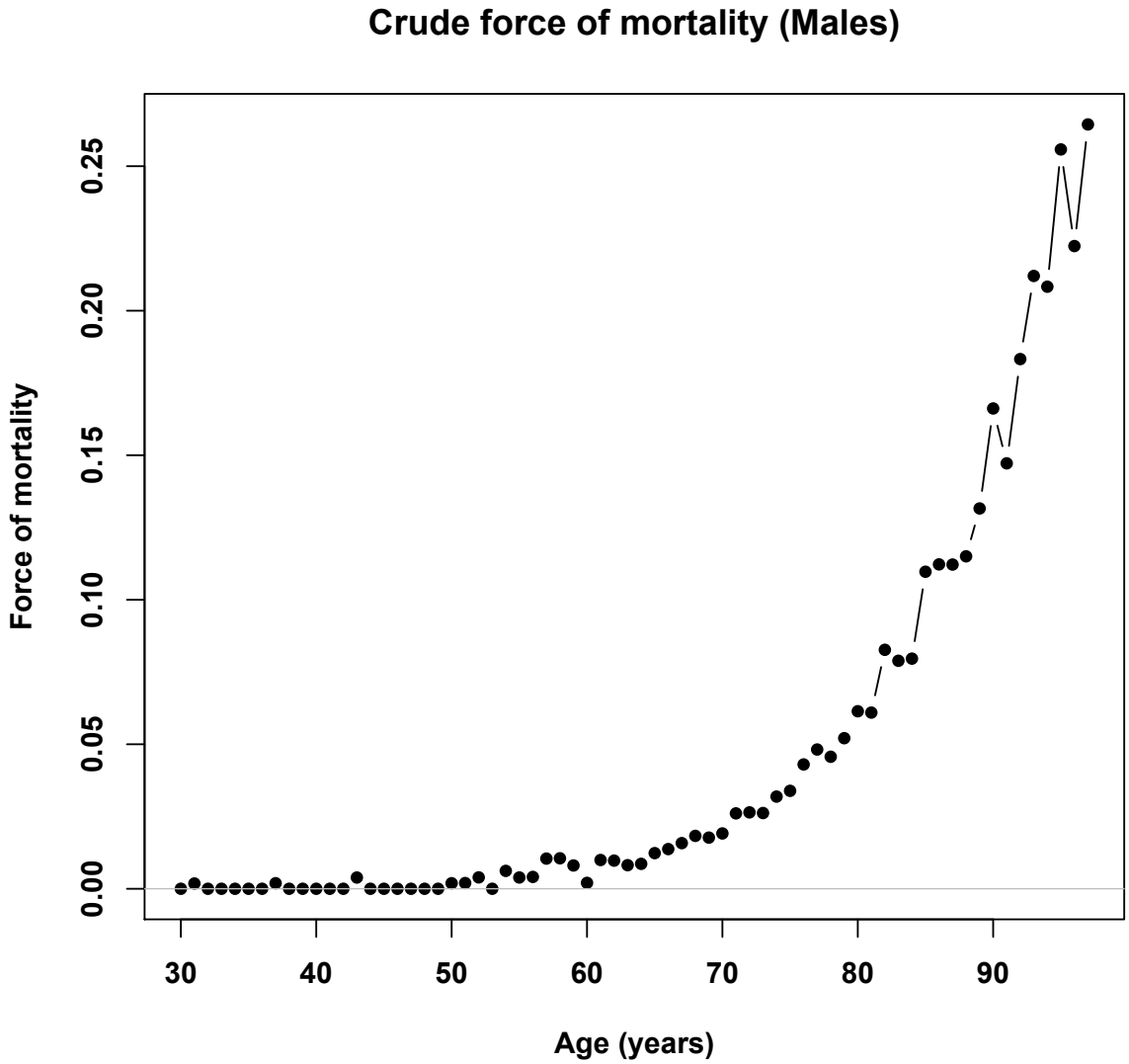
R implementations

There are many, many implementations of nonparametric regression methods in R, including

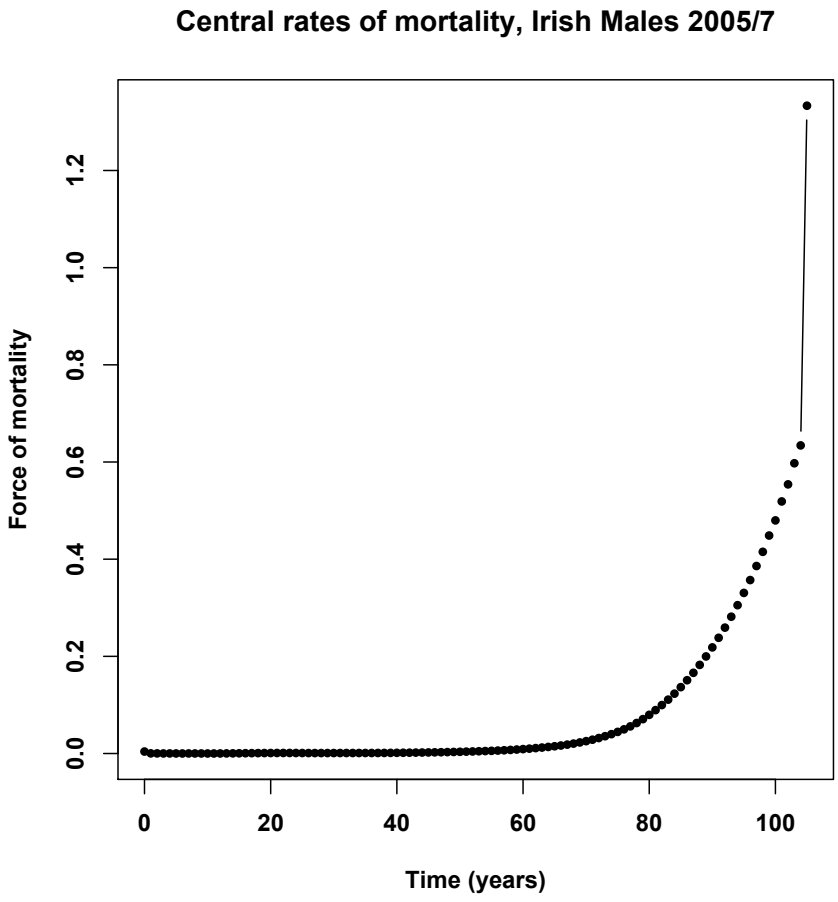
- `lowess` for simple regression, i.e. $g(x) = g_1(x_1)$
- `loess` for local polynomial regression
- `smooth.spline` for fitting simple regression smoothing splines
- `ksmooth` for Nadaraya-Watson kernel regression
- the `sm` library for local regression, local likelihood and pdf estimation
- the `gss` library for generalized smoothing splines and regression models
- `locfit`, `gam` and `mgcv` also come up regularly

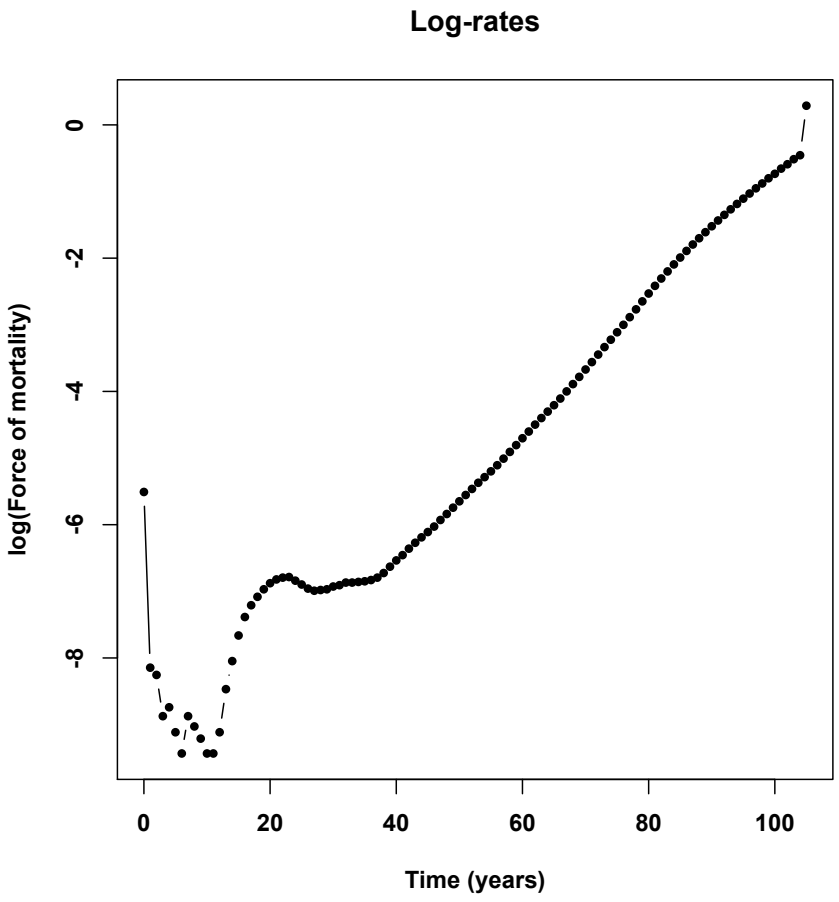
4.3 - Graduation

Dealing with rough data (simulated life data, D=585)



Or with unbalanced information (Irish Males, 2005-07)





Fitting mortality rates

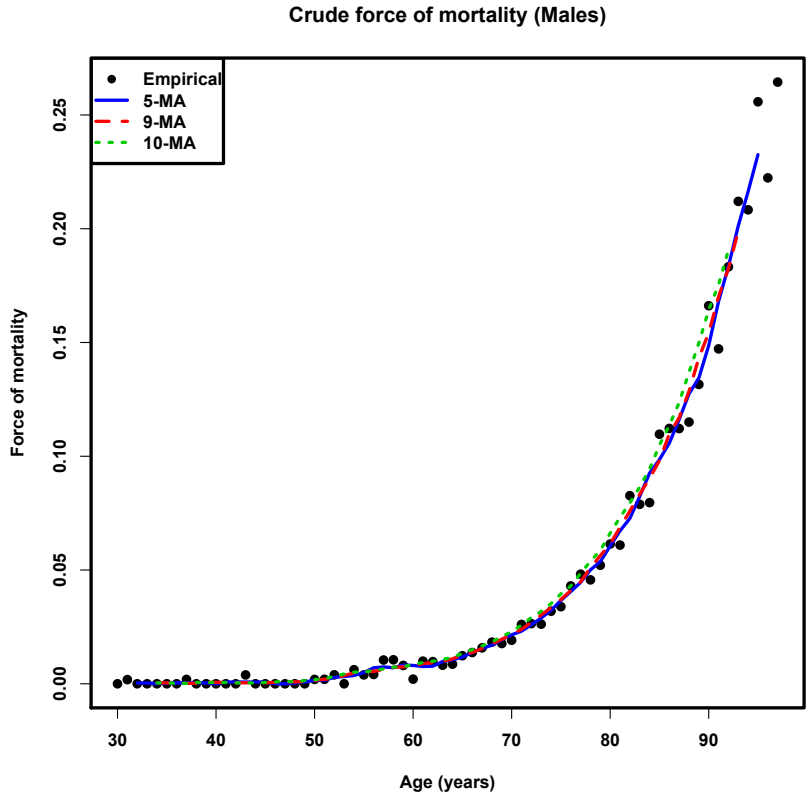
- **Objective:** to fit a curve of mortality rates from either dense information (e.g. life table from population census) or small, patchy sample (e.g. late-life characteristics for pensioners)
- **Steps:**
  - Obtain observed mortality rates
  - Fit some functional model to it
  - Forecast late-life, age-derived features, or future generations
- **Challenges:**
  - High variability due to small sample information
  - Missing information for specific ages
  - Parametric models maybe ill-suited (late-life)

Some standard smoothers

- Running mean / median
- Linear time-invariant filters
- Kernel smoothing
- Loess / polynomial regression
- Splines
- . . . and parametric models?

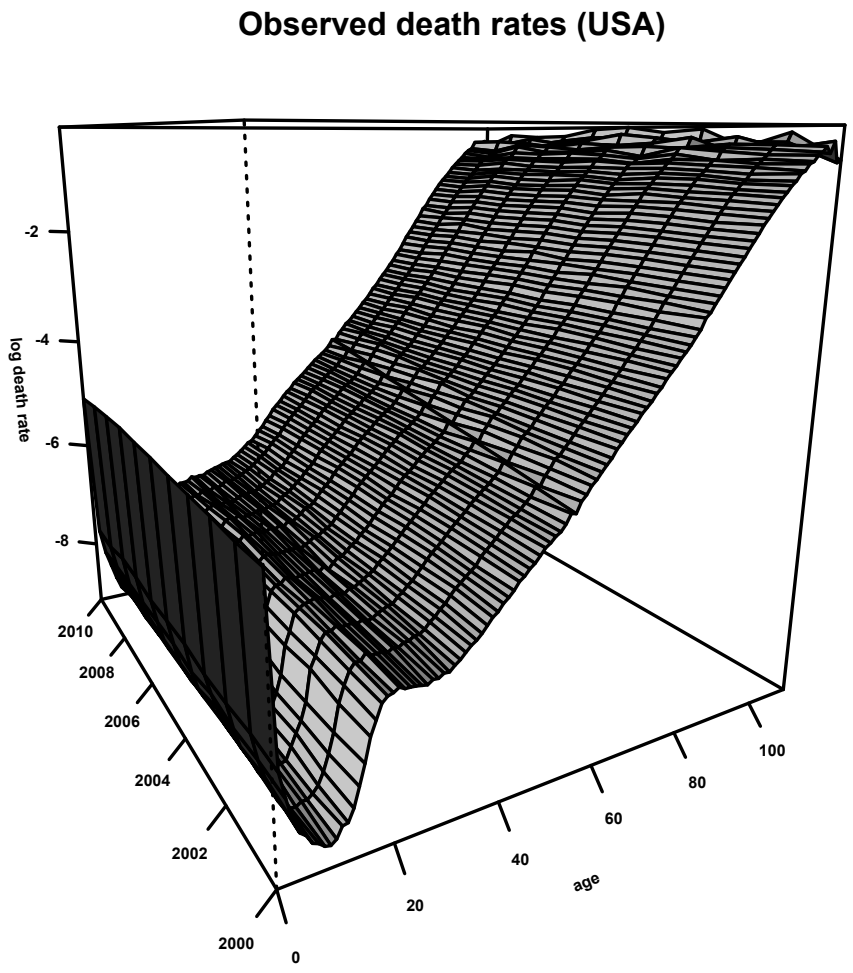
Running mean/median

- Write out an algorithmic formula for the running mean
- What calibration must be done? What must be controlled?
- What advantage(s)/inconvenient(s) do these methods have?



Running mean/median

cf. example on 2000-2010 USA mortality rates...



Linear time-invariant filters

- Algorithmic formula for an LTI?
- What calibration must be done? What must be controlled?
- What advantage(s)/inconvenient(s) does this method have?

Kernel smoothing

- Write out an algorithmic formula for a kernel smoother
- What calibration must be done? What must be controlled?
- What advantage(s)/inconvenient(s) do these methods have?

Polynomial regression

- Write out an algorithmic formula for a polynomial smoother
- What calibration must be done? What must be controlled?
- What advantage(s)/inconvenient(s) do these methods have?

Question:

1. Apply lowess() to both the USA 2000 log-mortality rates and the (non-log) crude mortality rates from the Life dataset, and visualize the outputs. Adjust the smoother span to successively 2/3, 1/3 and 0.1, and compare outputs.
2. Compute the MSE for each smooth, for both datasets.
3. Based on the above, describe the main steps for a grid-search approach for selecting an appropriate value for the span.

Assessment of graduation

- Typical issues:
  - Lack of goodness-of-fit, either overall or at specific locations
  - Consistent bias
  - Inconsistency in overall shape
- Comparing smoothing procedures: use tests, e.g. χ^2-test

Assessment of graduation / comparisons

χ^2-test

- $\mathcal{H}_0$: graduated estimates $\{\mu_x\}$ are true
- Assumes independent numbers of deaths at different ages
- Asusming $D_x \sim \mathcal{N}(E_x^c \mu_x^0, E_x^c \mu_x^0)$ (Poisson-Normal approx.),

$$\text{Actual deaths - Expected deaths} = D_x - E_x^c \mu_x^0$$

(E_x^c is the central exposed-to-risk at age x nearest birthday)

$$z_x = \frac{D_x - E_x^c \mu_x^0}{\sqrt{E_x^c \mu_x^0}} \sim \mathcal{N}(0,1) \text{ (under } \mathcal{H}_0)$$

and

$$\xi = \sum_{i=1}^{m} z_{x_i}^2 \sim \chi_m^2, \quad m = \text{number of age groups}$$

Plenty methods yielding smooth curves...

- Choice of method should be data-driven (and user-driven?)
- Parametric modelling is a form of smoothing
- Ex: fit Makeham model to a rough mortality curve...
- Exponential smoothing (cf. time series) is another adaptive procedure
- Weighted functional polynomials
- Etc.

Mortality rates projections

- IoFA: in-house method for mortality projection
- Cf. implementation of 2009 model for mortality rates proj.
- Office Nat. Stat. population mortality data
- Use of age-cohort P-spline early on in processing step
- Recent CMI documents [7] illustrate concerns that current projection methods are not statistical in nature
- Lee-Carter typically used for projection of smoothed data