Smoothing

Mikis Stasinopoulos[1] Bob Rigby[1]

[1]STORM, London Metropolitan University

GAMLSS: Short course Campina Grande, Brazil, Jul 2013

gamlss

gamlss

Health and Lifestyle Survey Data

5191 observations
Adults age 18 upwards, in England, Wales and Scotland, 1984-5.
Height: was measured without shoes using a stadiometer reading to 1 mm.
Income: Household Equivalent Income obtained using the McClemets scale which divides the net household income by a factor which is dependent on the presence of spouse, number of dependent children and the number of other adults in the household.

gamlss

Health and Lifestyle Survey Data: plot

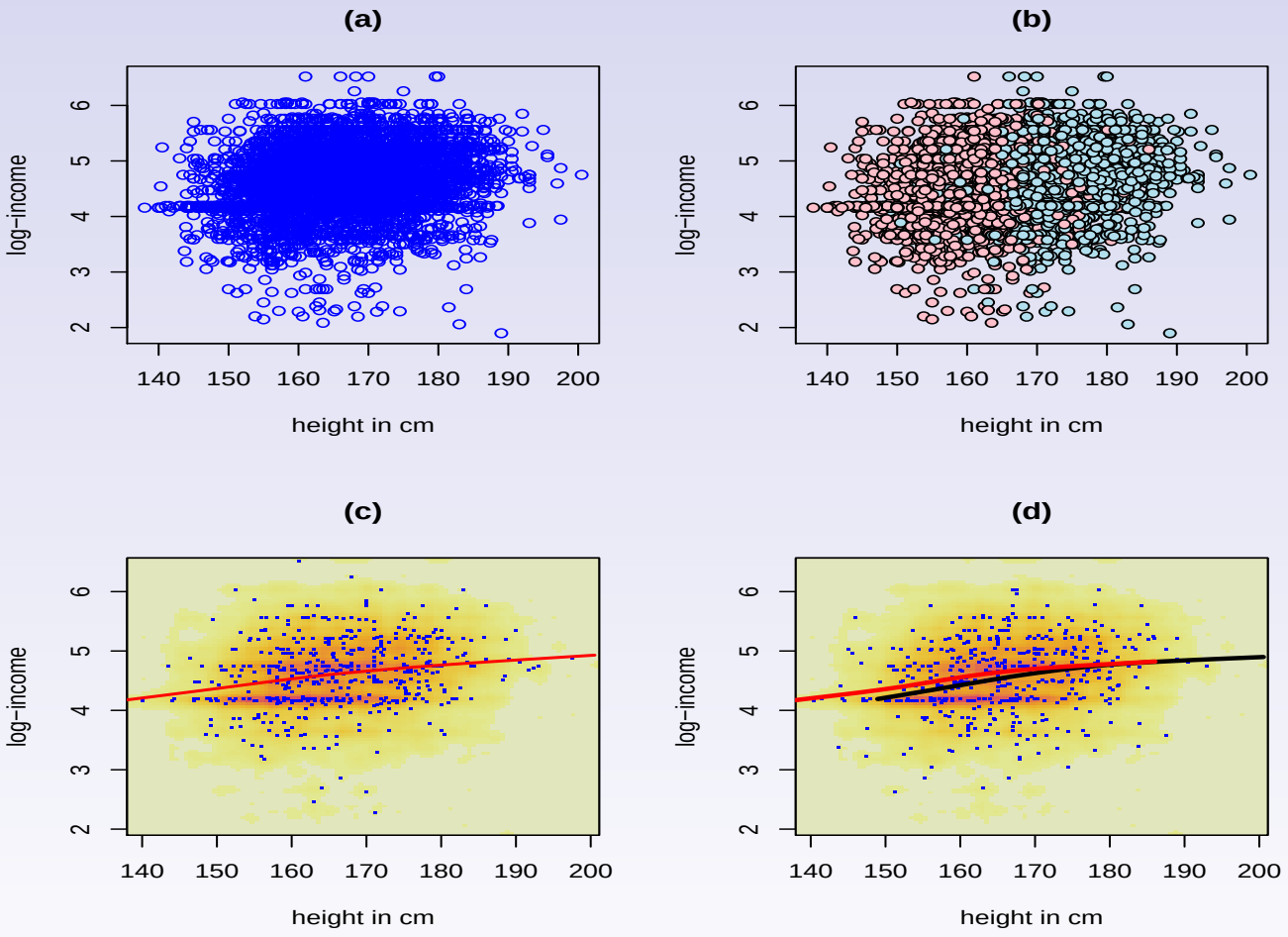


Figure : Height against log-income together with a lowess smoother

gamlss

Reporting crime in the local media: the data

10590 observations

media: whether crime reported ($y = 1$) or not ($y = 0$) in the local media

age: the age of the victim of crime

gamlss

Reporting crime in the local media: plot

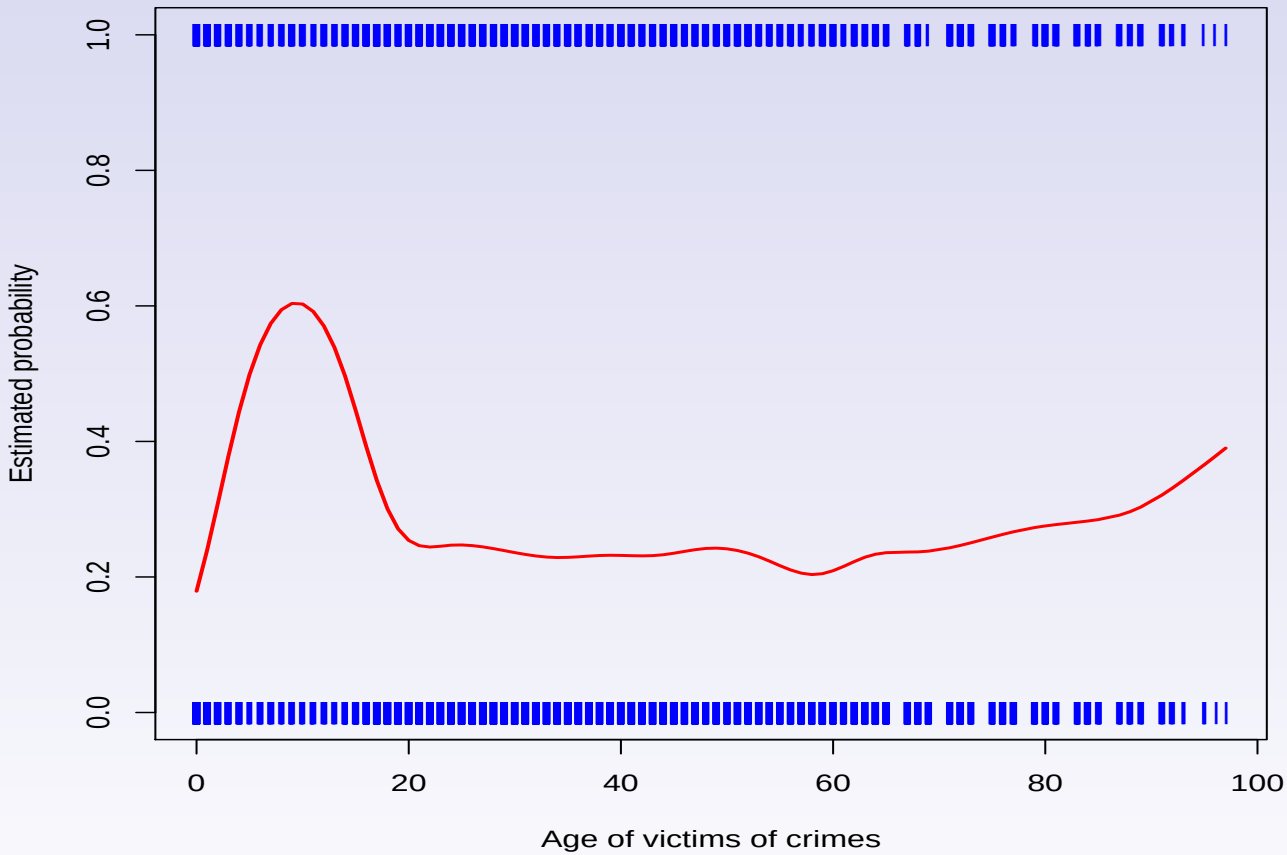


Figure : Reported cases by media against the age of the victim.

gamlss

The Munich rent data

`rent` : the monthly net rent for flats in the city of Munich.

`area` : the floor space area in square meters

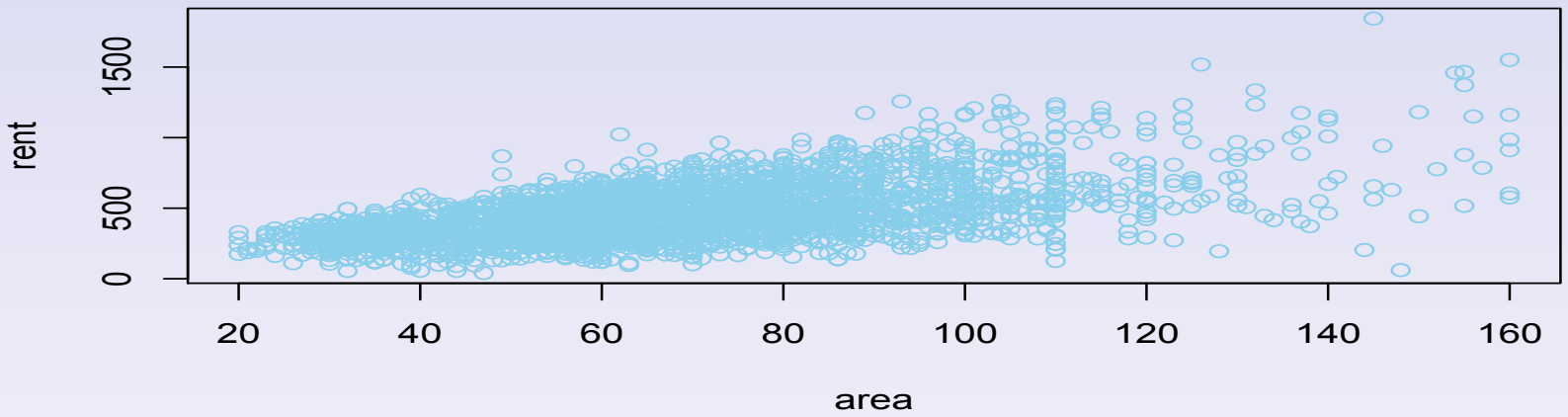
`yearc` : year of construction

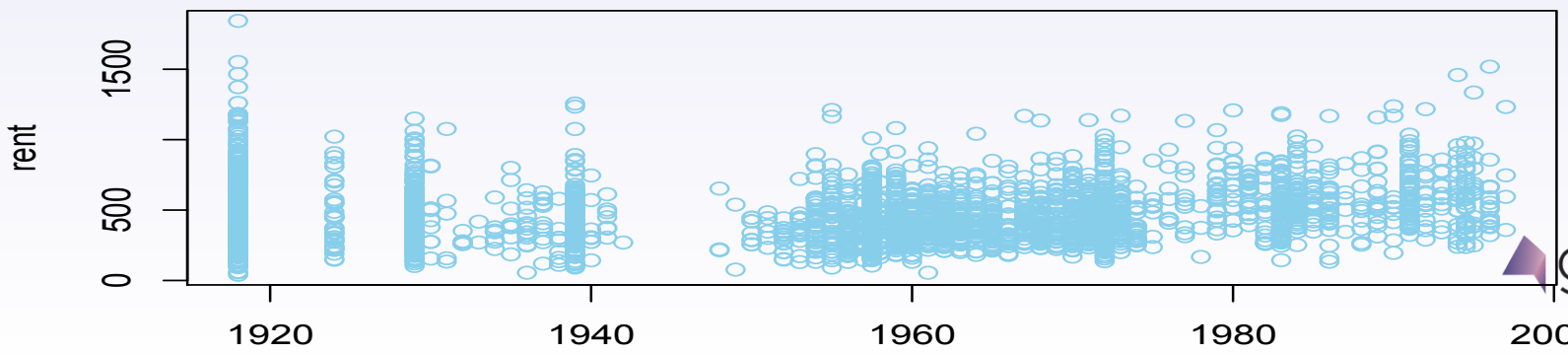
`district` : the district of the flat

Source: Munich rental guide 1999, (3493 observations)

gamlss

The Munich rent data: area and year of construction





gamlss

Introduction

Scatter plot smoothers:

- local polynomials
- Penalised piecewise polynomials

gamlss

Local polynomials

- Unweighted
  - Running means smoother
  - Local polynomial smoother
- Weighted
  - weighed running means smoother or kernel smoother
  - weighed local polynomials smoother
    - loess

gamlss

Running means smoothers

demo.Locmean(): for symmetric local window mean fitting

gamlss

Local polynomials smoother

demo.Locpoly(): for symmetric local window polynomial fitting

gamlss

Weighed running means smoothers

demo.WLocmean(): for Gaussian kernel fitting

gamlss

Weighed running means smoothers

demo.WLocpoly(): for weighed local polynomial fitting using a Gaussian kernel window

gamlss

Weighed running means smoothers: summary

- weighted smoother better in the eye than unweighted
- smoothing parameter: span or λ.
- important how the x-values are taken into the account i.e. local symmetric window, weighted window etc.
- linear smoothers $\hat{\mathbf{y}} = \mathbf{S}\mathbf{y}$.
- smoother are effected differently with the sparseness of the x-values. i.e. kernel smoother can misbehave at the end of the data
- smoothers are robust for extreme values at x-axis since only local observations taking part on the fit.
- influential observations in the y-axis do effect the smoothers.

gamlss

Penalised piecewise polynomial smoothers

- Piecewise Polynomials
- Regressions Splines (without penalty)
- Penalised Regression Splines

gamlss

Piecewise Polynomials: linear

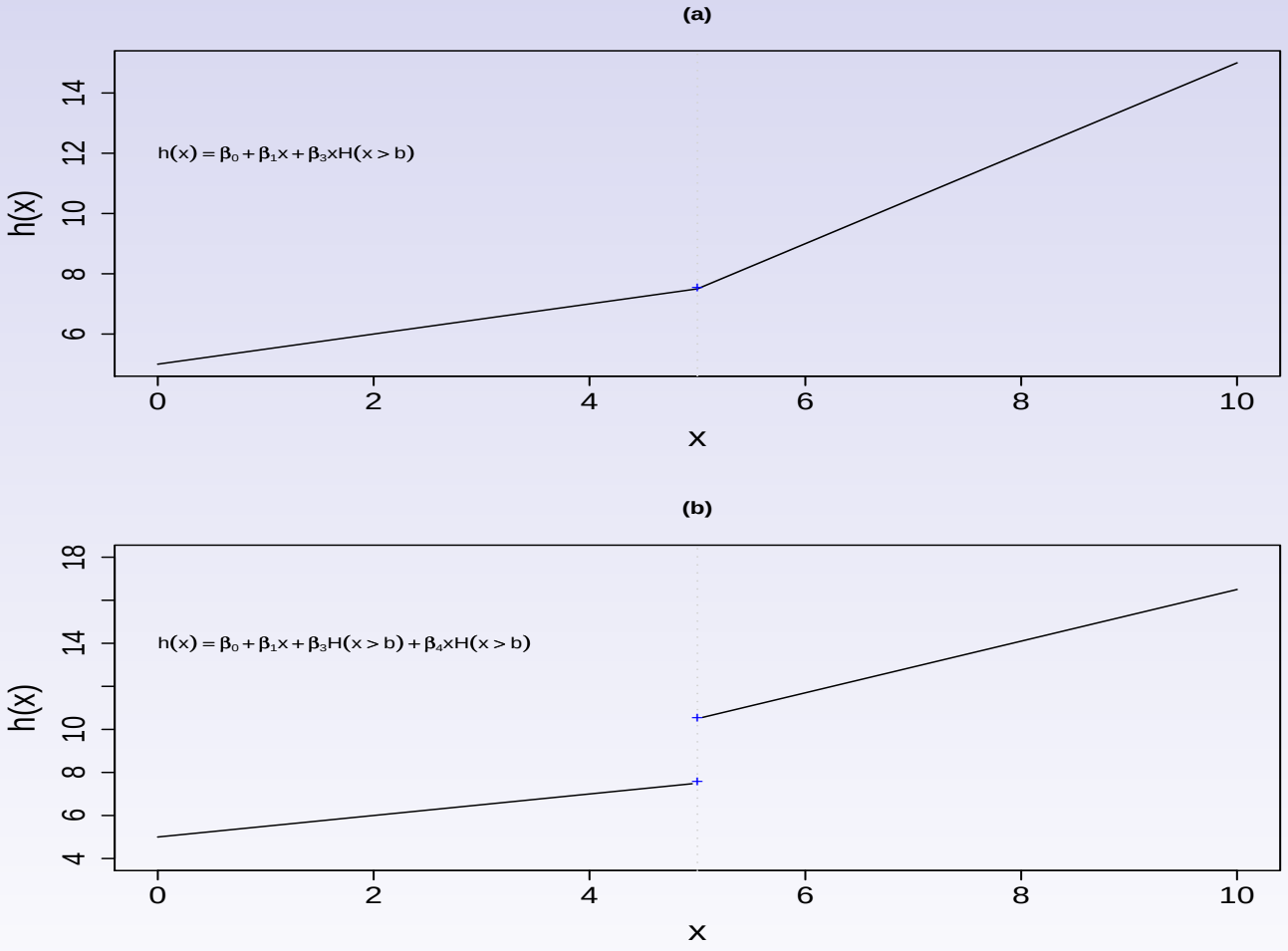


Figure : Piecewise linear, continuous(a) and discontinuous (b) lines.

gamlss

Piecewise Polynomials: linear

$$h(x) = \beta_{00} + \beta_{01}x + \beta_{11}(x - b)H(x > b) \tag{1}$$

$$h(x) = \beta_{00} + \beta_{01}x + [\beta_{10} + \beta_{11}(x - b)]\, H(x > b) \tag{2}$$

gamlss

Piecewise Polynomials: quadratic

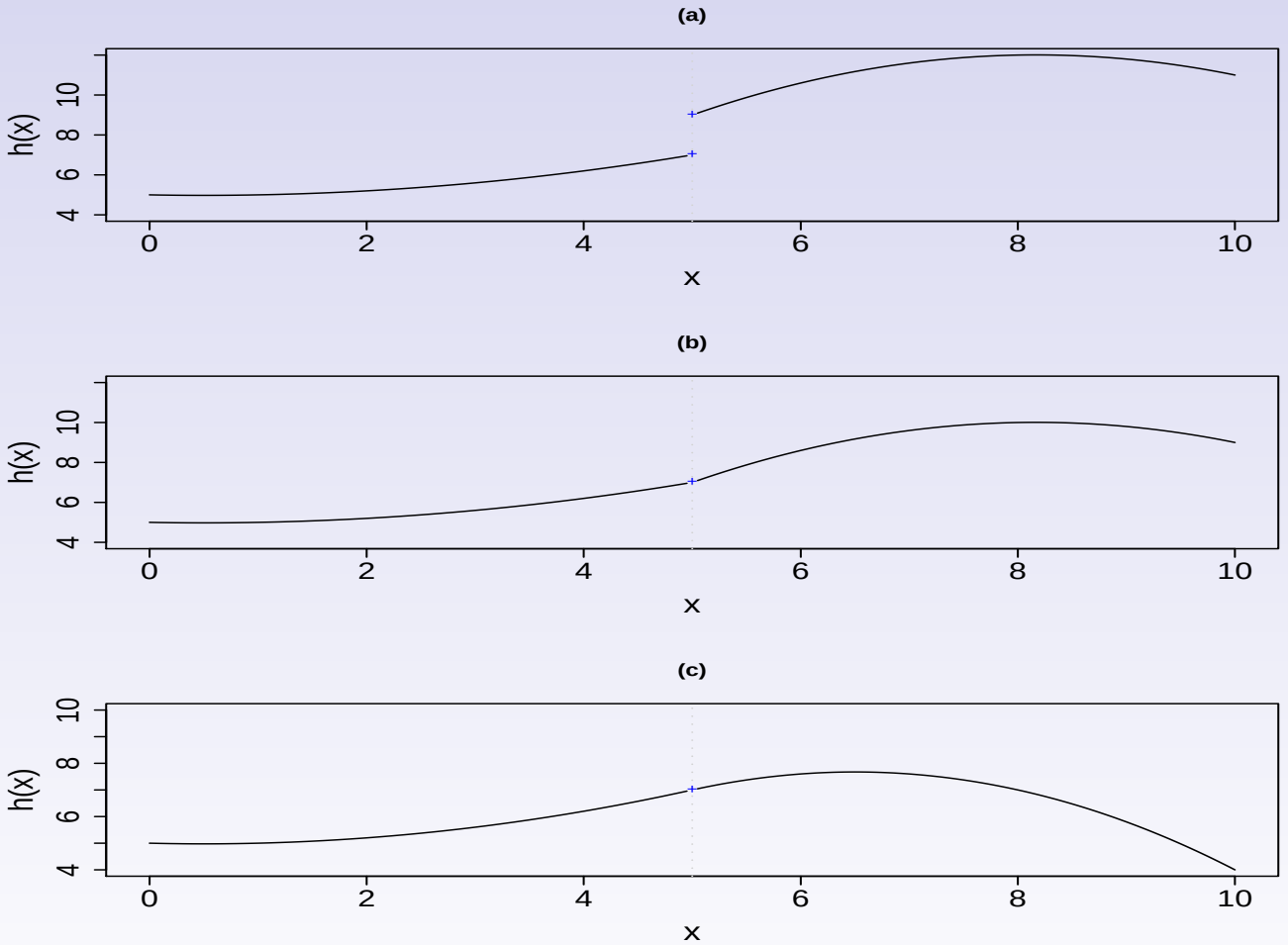


Figure : Piecewise quadratic.

gamlss

Piecewise Polynomials: quadratic

$$
\begin{aligned}
h(x) &= \beta_{00} + \beta_{01}x + \beta_{02}x^2 + \\
&\quad \left[\beta_{10} + \beta_{11}(x-b) + \beta_{12}(x-b)^2\right] H(x > b)
\end{aligned}
$$

gamlss

Piecewise Polynomials: general

Piecewise Polynomials:

$$h(x) = \sum_{j=0}^{D} \beta_{0j} x^j + \sum_{k=1}^{K} \sum_{j=0}^{D} \beta_{kj} (x - b_k)^j H(x > b_k) \quad (3)$$

Splines

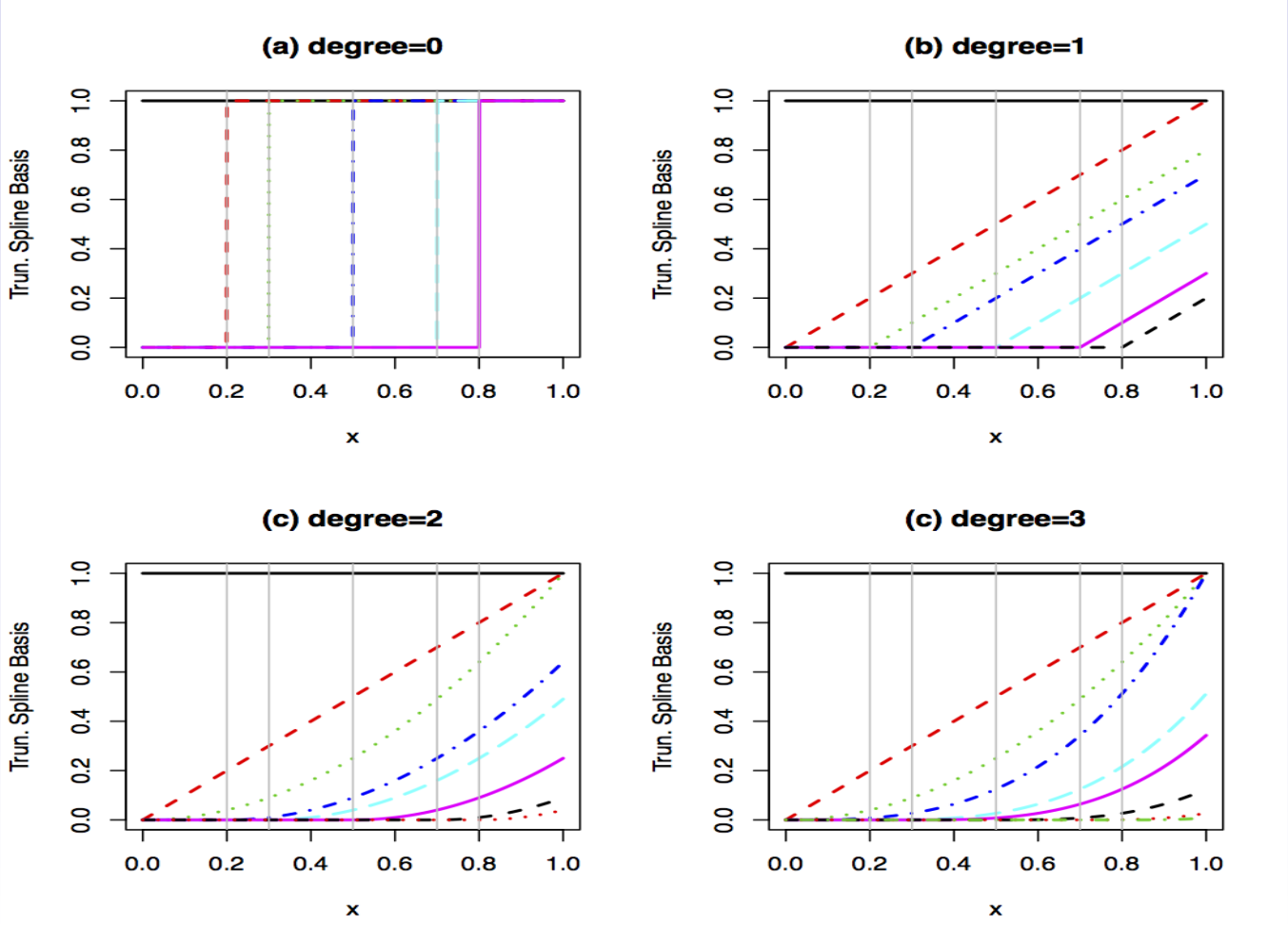
$$h(x) = \sum_{j=0}^{D} \beta_{0j} x^j + \sum_{k=1}^{K} \beta_k (x - b_k)^D H(x > b_k) \quad (4)$$

Cubic Splines

$$h(x) = \beta_{00} + \beta_{01} x + \beta_{02} x^2 + \beta_{03} x^3 + \sum_{k=1}^{K} \beta_k (x - b_k)^3 H(x > b_k) \quad (5)$$

gamlss

Piecewise Polynomials: different degrees a) constant, b) linear, c) quadratic and d) cubic, break points at $(0.2, 0.4, 0.5, 0.6, 0.8)$



gamlss

B-splines

demo.BetaSplines(): for B-spline

gamlss

B splines

- B-splines are local functions, look like Gaussian (for degree=3)
- B-splines are columns of basis matrix $\mathbf{B}$
- scaling and summing gives fitted values: $\boldsymbol{\mu} = \mathbf{B}\boldsymbol{\gamma}$
- the knots determine the B-spline basis
- Polynomial pieces make up B-splines, joint at knots
- General patterns of knots are possible

Copied from Paul Eilers notes

gamlss

Regressions Splines (without penalty)

- Fixed knots (or break points):
  - **gamlss**: `bs()` and `ns()`
  - **gamlss.add** `fitFixedKnots()`
- Free knots (or break points):
  - **gamlss.add**: `fitFreeKnots()`
  - **gamlss.add**: `fk()` (additive terms in gamlss)

gamlss

Penalised Regression Splines

- Penalised discrete smoothing (smoothing the mean)
- Penalised B-spline smoothing (smoothing a function)

gamlss

Penalised discrete smoothing

- data series y_i for $i = 1, 2, \ldots, n$
- Wanted: a smooth series μ
- Two (conflicting) goals: fidelity to y and smoothness
- Fidelity, sum of squares: $S = \sum_i (y_i - \mu_i)^2$
- How to quantify smoothness?
- Use roughness: $R = \sum_i (\mu_i - \mu_{i-1})^2$
- Simplification of Whittaker (1923) "graduation"

Copied from Paul Eilers's notes

gamlss

Penalised least squares

- Combine fidelity and roughness

$$Q = S + \lambda R = \sum_i (y_i - \mu_i)^2 + \lambda \sum_i (\mu_i - \mu_{i-1})^2$$

- Parameter λ sets the balance
- Operator notation $\Delta\mu_i = \mu_i - \mu_{i-1}$

$$Q = \sum_i (y_i - \mu_i)^2 + \lambda \sum_i (\Delta\mu_i)^2$$

Copied from Paul Eilers's notes

gamlss

Matrix notation

- Penalised least squares objective function

$$Q = ||\mathbf{y} - \boldsymbol{\mu}||^2 + \lambda||\mathbf{D}\boldsymbol{\mu}||^2$$

- Differencing matrix $\mathbf{D}$, such that $\mathbf{D}\boldsymbol{\mu} = \Delta\boldsymbol{\mu}$

$$\mathbf{D} = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

- Explicit solution

$$\hat{\boldsymbol{\mu}} = \left(I + \lambda\mathbf{D}^\top\mathbf{D}\right)^{-1}\mathbf{y}$$

gamlss

Copied from Paul Eilers's notes

Higher order penalties

- Higher order differences are easily defined
- Second order: $\Delta^2\mu_i = \Delta(\Delta\mu_i) = (\mu_i - \mu_{i-1}) - (\mu_{i-1} - \mu_{i-2})$
- Second order differencing matrix $\mathbf{D}$,

$$\mathbf{D} = \begin{bmatrix} -1 & -2 & 1 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 1 & -2 & 1 \end{bmatrix}$$

- Higher orders are straightforward item in R:
  `D=diff(diag(M), diff=d)`

Copied from Paul Eilers's notes

gamlss

Penalised discrete smoothing

demo.discreteSmo(): for penalised discrete smoothing
demo.interpolateSmo(): for showing the effect of interpolation and extrapolation

gamlss

Interpolation and extrapolation

- Interpolation is by polynomial of degree $2d - 1$
- Extrapolation: introduce "missing" data in the end(s)
- Extrapolation is by polynomial of degree $d - 1$

Copied from Paul Eilers's notes

gamlss

Optimal smoothing

- How much we should smooth
- Let data decide
- Cross-validation, Generalized cross validation, AIC, SBC, (BIC), local maximum likelihood
- Essentially we measure prediction performance

Copied from Paul Eilers notes

gamlss

Penalised B-spline smoothing: P-splines

- do regression on (cubic) B-splines
- take a large number of them
- put a difference penalty (order 2 or 3) on the coefficients
- tune smoothness with λ
- don't try tp optimise the number of B-spline knots
- relatively small system of equations $(10, 20, 50)$

Copied from Paul Eilers notes

gamlss

Technical details of P-splines

- minimise (with basis $\mathbf{B}$)

$$Q = ||\mathbf{y} - \mathbf{B}\boldsymbol{\gamma}||^2 + \lambda||\mathbf{D}\boldsymbol{\gamma}||^2$$

- explicit solution $\hat{\boldsymbol{\gamma}} = \left(\mathbf{B}^\top\mathbf{B} + \lambda\mathbf{D}^\top\mathbf{D}\right)^{-1}\mathbf{B}^\top\mathbf{y}$
- hat matrix $\mathbf{H} = \mathbf{B}\left(\mathbf{B}^\top\mathbf{B} + \lambda\mathbf{D}^\top\mathbf{D}\right)^{-1}\mathbf{B}^\top$

Copied from Paul Eilers's notes

gamlss

P-splines

demo.PenSplines(): for penalised beta splines

gamlss

P-splines weighted

$$\hat{\gamma} = \left(\mathbf{B}^\top \mathbf{W} \mathbf{B} + \lambda \mathbf{D}^\top \mathbf{D}\right)^{-1} \mathbf{B}^\top \mathbf{W} \mathbf{y}$$

where $\mathbf{W}$ is diagonal matrix

It is amazing what you can do by changing parts of the above equation

ridge regression

$$\hat{\gamma} = \left(\mathbf{X}^\top \mathbf{W} \mathbf{X} + \lambda \mathbf{I}\right)^{-1} \mathbf{X}^\top \mathbf{W} \mathbf{y}$$

random effects

$$\hat{\gamma} = \left(\mathbf{W} + \lambda \mathbf{I}\right)^{-1} \mathbf{W} \mathbf{y}$$

gamlss

The smoothers in gamlss

`pb()`	Penalised beta splines
`cy()`	Cycle Penalised beta splines
`pvc()`	Penalised Varying coefficient
`random()`	random effects
`ridge()`	ridge regression
`rmf()`	Markov random fields
`tp()`	Tensor Product (see below)
`la()`	Penaliesed Lags
`ga()`	Simon Wood's smoothers from package `mgcv`

gamlss

Non P-smoothers in gamlss

`cs()` Cubic spline

`lo()` `loess`

`nn()` neural networks

`tr()` decision (or regression) trees

`gamboostLSS` Boosting

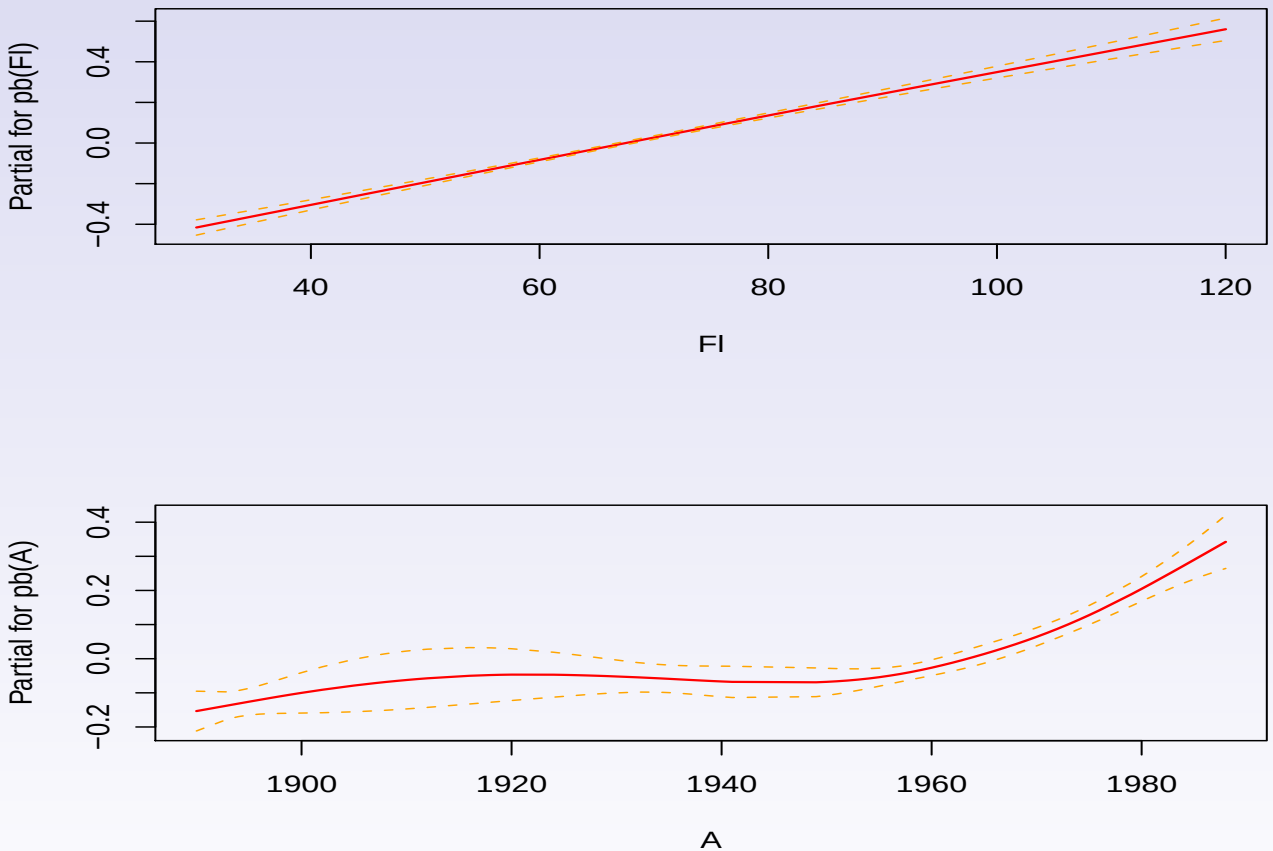
gamlss

P-splines, the pb() function: additive fit

```
 r1<-gamlss(R~pb(Fl)+pb(A),data=rent, family=GA)
GAMLSS-RS iteration 1: Global Deviance = 27923.61
GAMLSS-RS iteration 2: Global Deviance = 27923.61
op <- par(mfrow = c(2, 1))
term.plot(r1, se = TRUE, partial = TRUE)
par(op)
```

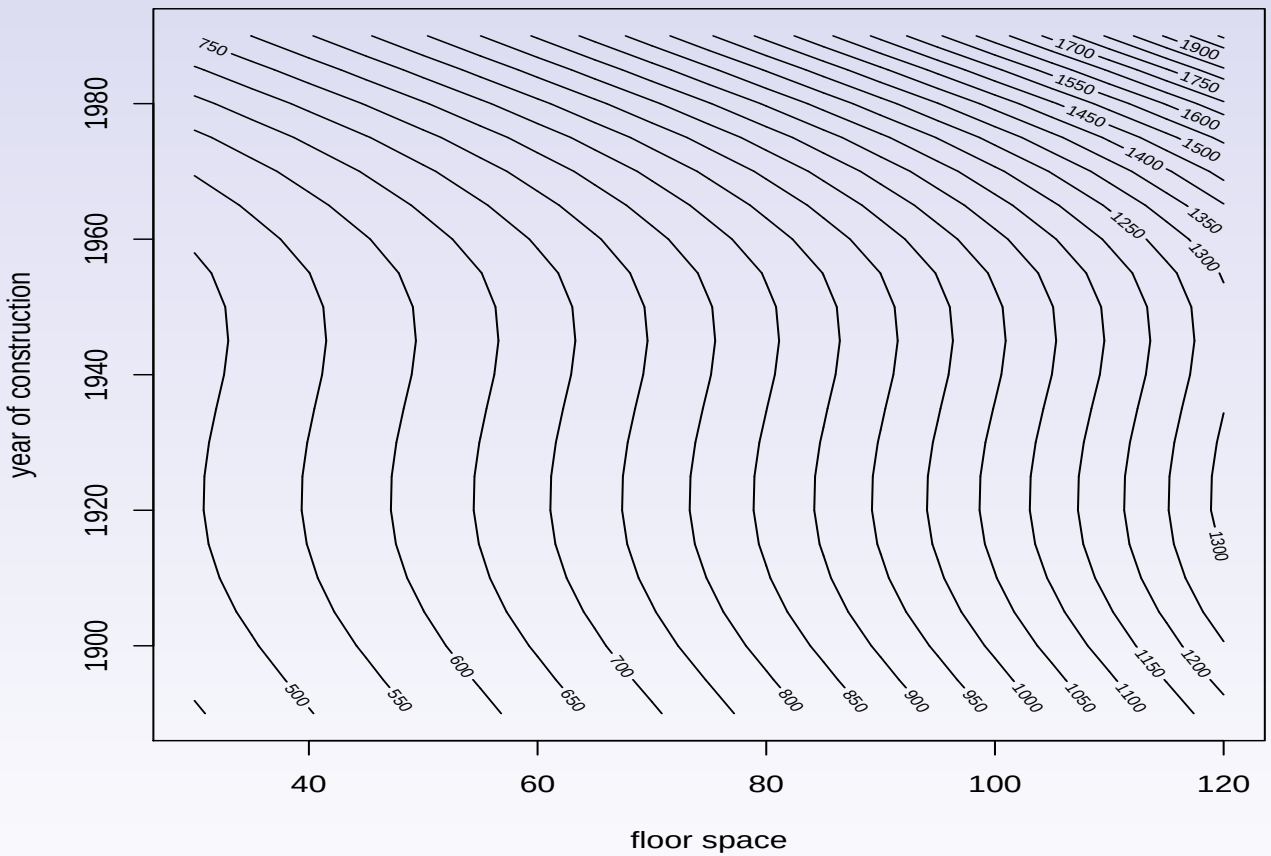
gamlss

P-splines, the pb() function: term.plot()



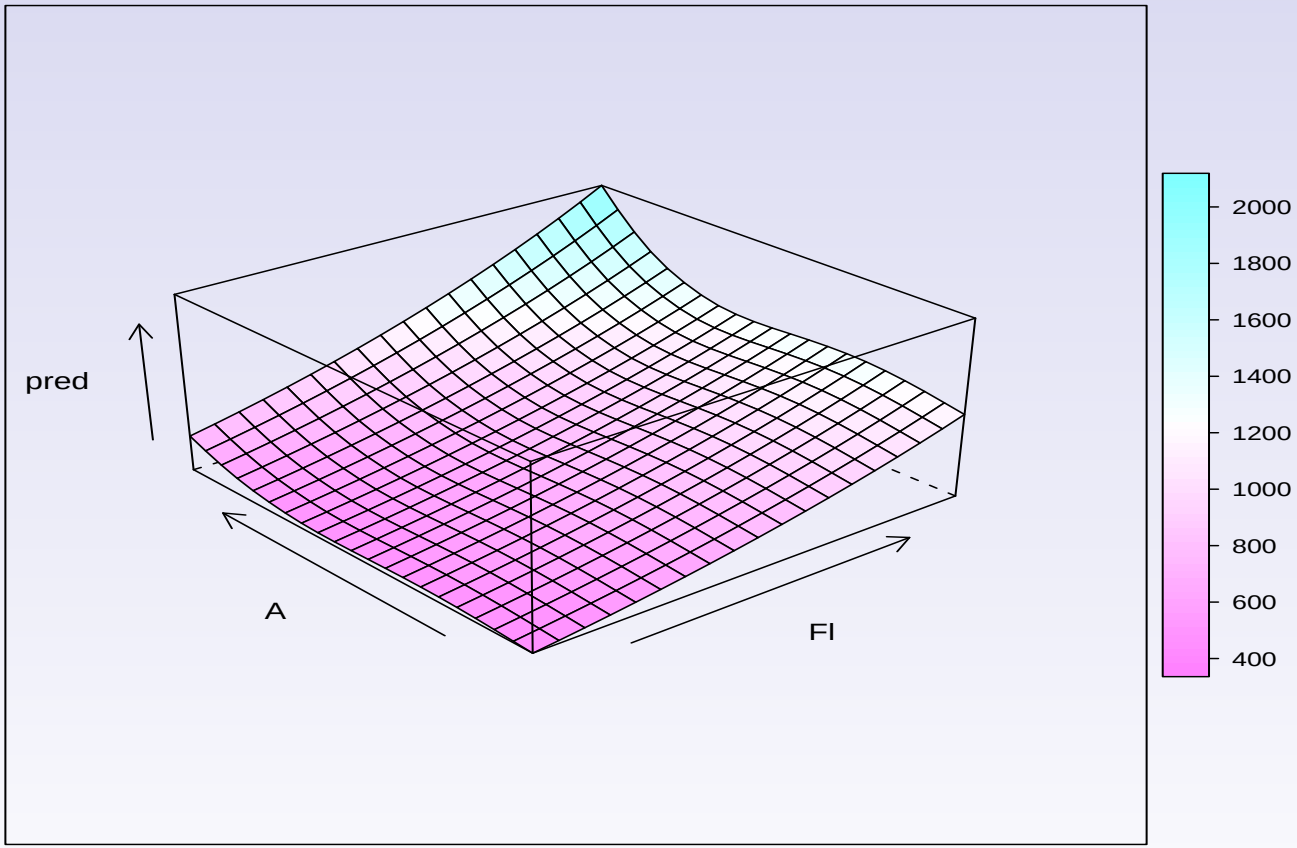
gamlss

P-splines, the pb() function: contour



gamlss

P-splines, the pb() wireframe()



gamlss

Varying coefficient, the pvc() function

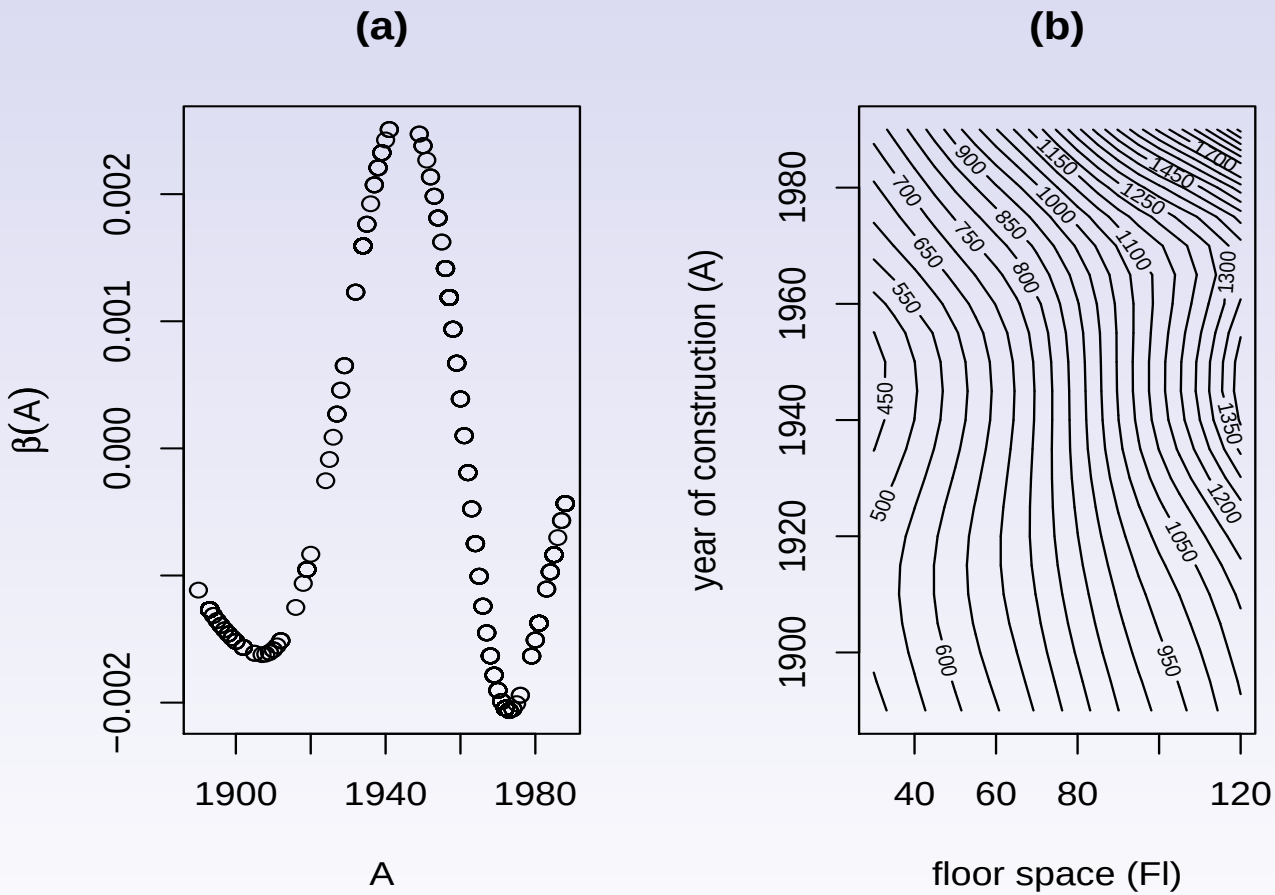
Varying coefficient is a special type of interaction between explanatory variables.
This interaction takes the form of $\beta(R)X$,

Varying coefficient, the pvc() function

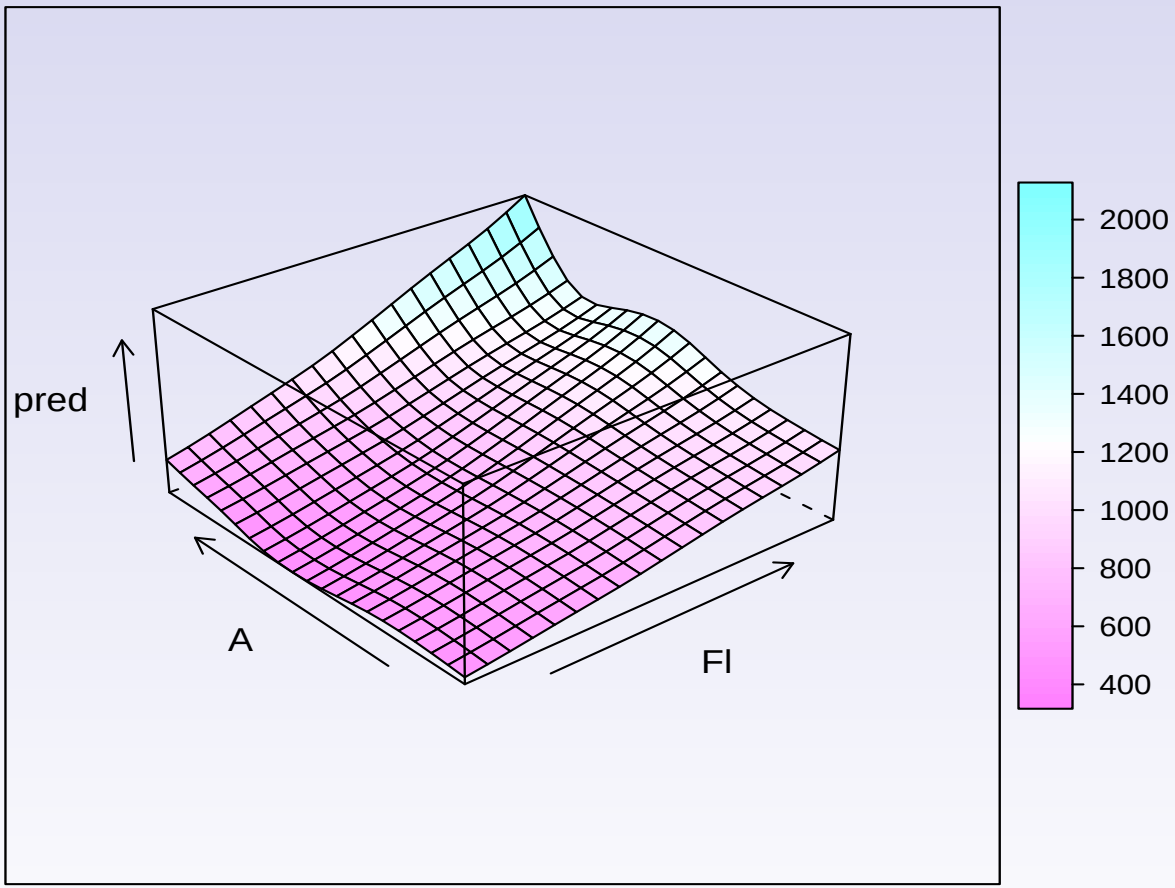
```
 r2 <- gamlss(R ~ pb(Fl) + pb(A) + pvc(A, by = Fl),
                data = rent,family = GA)
GAMLSS-RS iteration 1: Global Deviance = 27905.56
GAMLSS-RS iteration 2: Global Deviance = 27905.55
GAMLSS-RS iteration 3: Global Deviance = 27905.56
GAMLSS-RS iteration 4: Global Deviance = 27905.56
GAMLSS-RS iteration 5: Global Deviance = 27905.56
 AIC(r1, r2)
           df      AIC
r2 10.903716 27927.37
r1  7.371373 27938.35
```

gamlss

The pvc() function



The pvc() function



gamlss

The loess function lo()

span : how big is the neighbourhood
degree : the degree of the local polynomial

gamlss

loess, the rent data analysis

```
r11 <- gamlss(R ~ lo(Fl) + lo(A), data = rent, family = GA)
r12 <- gamlss(R ~ lo(Fl, degree = 2) + lo(A, degree = 2),
         data = rent, family = GA)
r21 <- gamlss(R ~ lo(Fl, A, degree = 1), data = rent,
          family = GA)
r22 <- gamlss(R ~ lo(Fl, A, degree = 2), data = rent,
           family = GA)
GAIC(r11, r12, r21, r22, k = 2.5)
           df      AIC
r22 19.08488 27936.78
r21 11.33833 27937.33
r11 10.00128 27946.69
r12 14.12759 27952.95
```

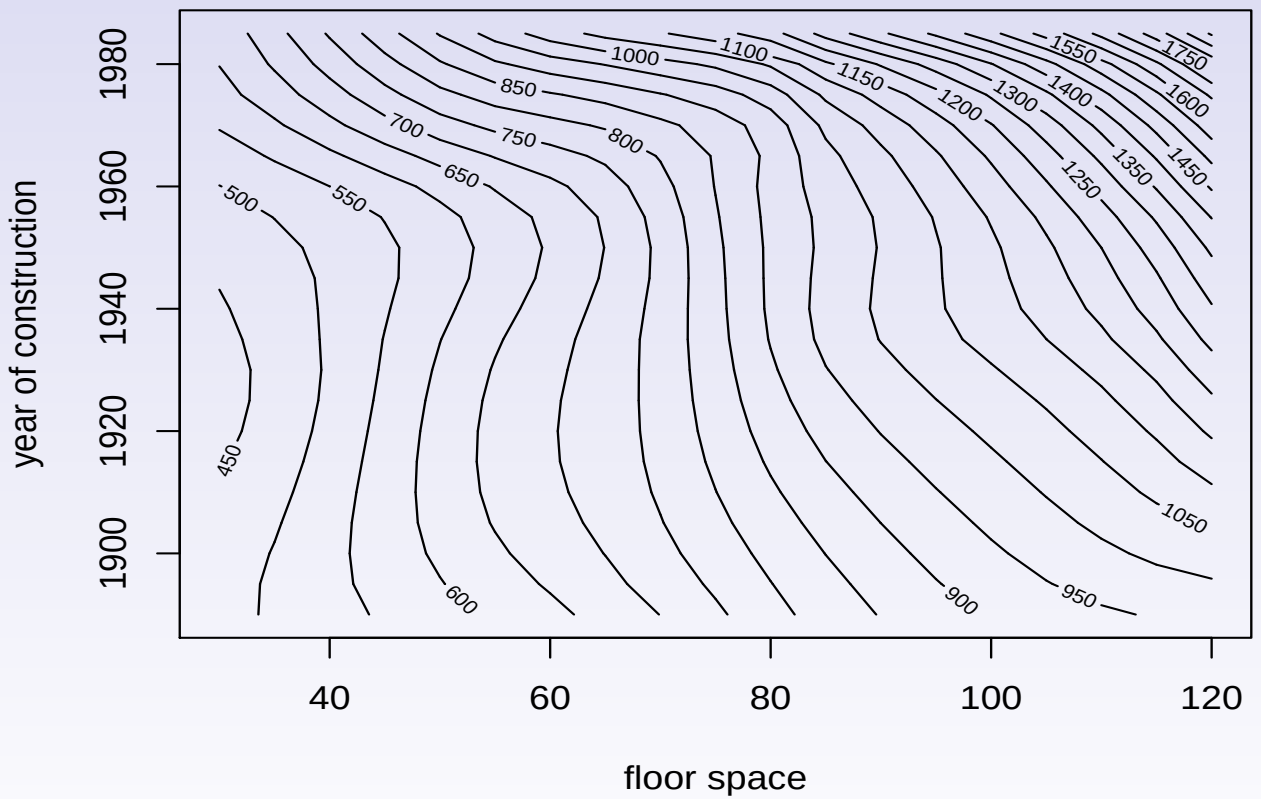
gamlss

loess, the rent data analysis

```
r223 <- gamlss(R ~ lo(Fl, A, degree = 2, span = 0.3),
   data = rent, family = GA)
r224 <- gamlss(R ~ lo(Fl, A, degree = 2, span = 0.4),
   data = rent, family = GA)
r226 <- gamlss(R ~ lo(Fl, A, degree = 2, span = 0.6),
   data = rent, family = GA)
AIC(r22, r223, r224, r226, k = 2.5)
            df       AIC
r22   19.08488 27936.78
r226  16.29977 27938.16
r224  22.46753 27939.15
r223  28.89809 27946.83
```

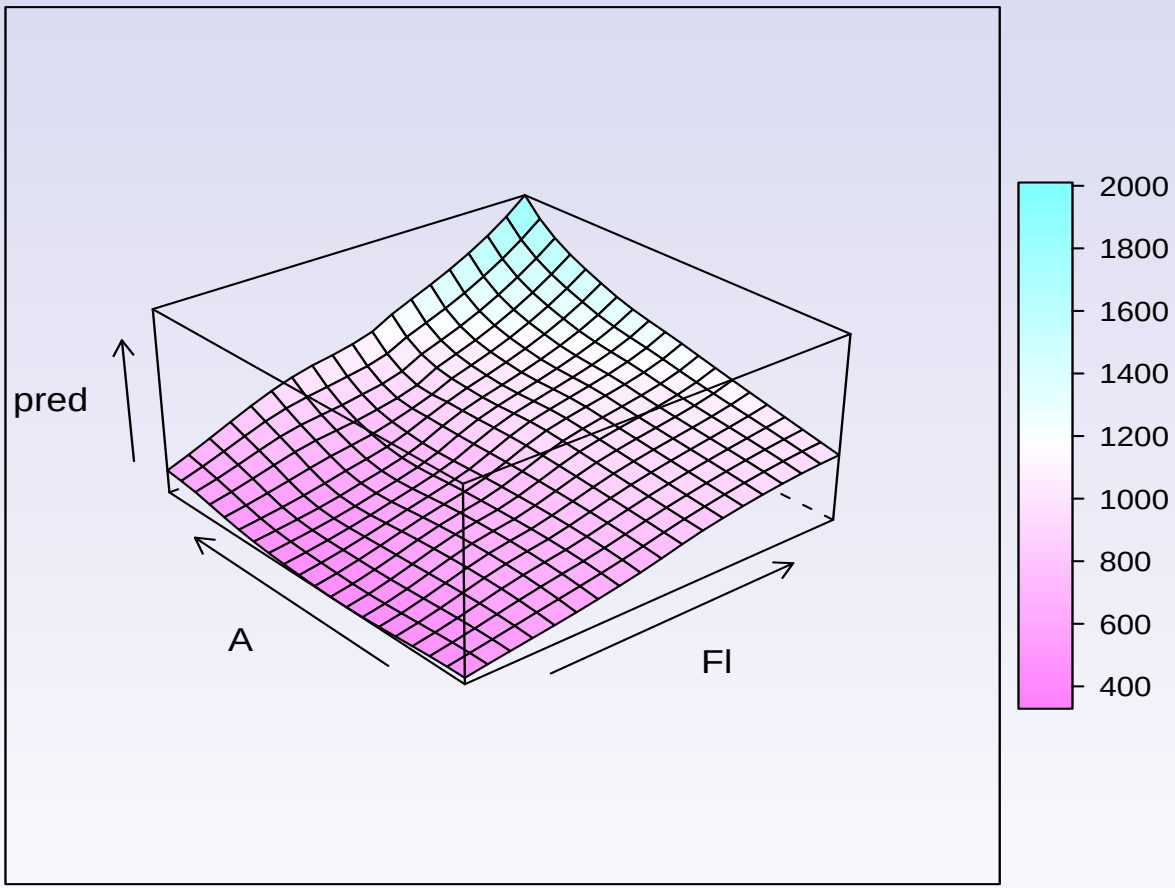
gamlss

loess, the rent data analysis



gamlss

loess, the rent data analysis



gamlss

The ga() function

Interface with the `gam()` function of package `mgcv`

For example `ga(~(x1,x2)+ s(x3))`

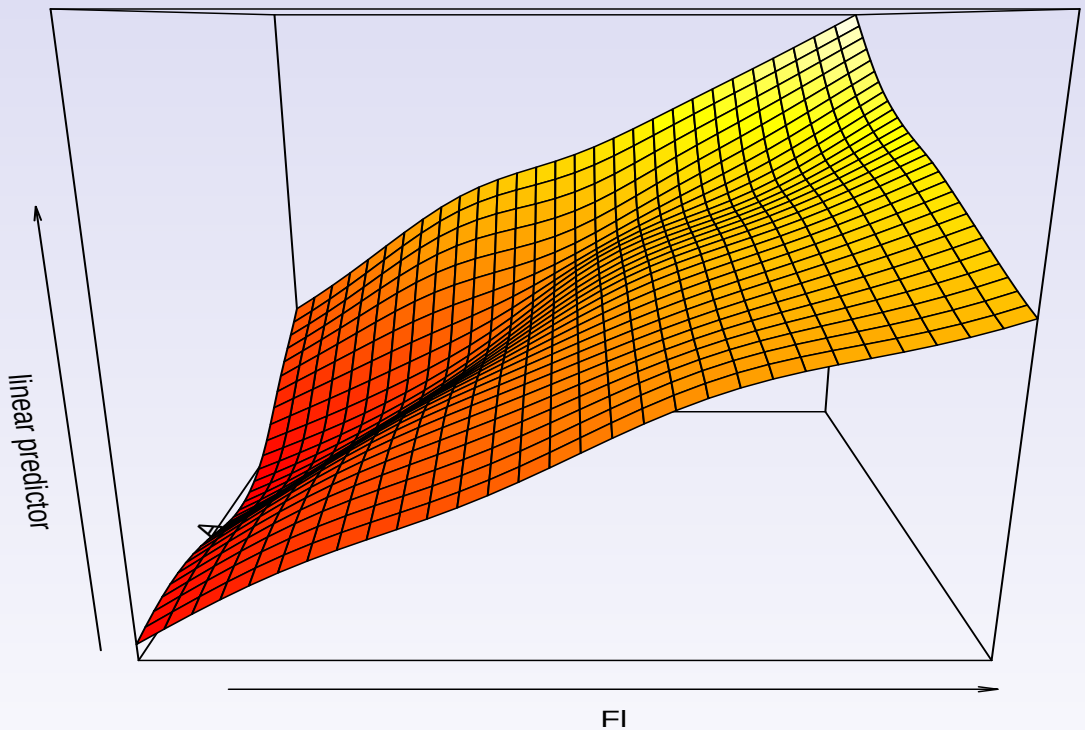
gamlss

The ga() function

```
library(gamlss.add)
g1 <- gamlss(R ~ ga(~s(Fl, A) ), data = rent, family = GA)
vis.gam(g1$mu.coefSmo[[1]])
```

gamlss

ga(), the rent data analysis



gamlss

END

for more information see

www.gamlss.org

gamlss