Applied Statistical Regression
AS 2015 – Extending the Linear Model

Marcel Dettling

Institute for Data Analysis and Process Design

Zurich University of Applied Sciences

marcel.dettling@zhaw.ch

http://stat.ethz.ch/~dettling

ETH Zürich, 30. November 2015

Generalized Additive Modelling (GAM)

Motivation:

We require a flexible regression method, similar to 1-dimensional smoothing, that also works in multiple regression setting.

Background:

The generic multiple regression formula is:

$$y_i = f(x_{i1}, x_{i2}, \ldots, x_{ip}) + E_i$$

As we have argued before, this is a too challenging problem, as there are just too many functions $f(\cdot)$. While in simple regression, visualization of the function is feasible, this is no longer the case in a multiple regression where $p > 2$ *("curse of dimensionality")*.

Solution 1: Linear Modelling with OLS

The canoncical approach for solving the multiple regression problem lies in using parametric linear models such as:

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_p x_{ip} + E_i$$

As we know, the predictors x_{ij} may be transformed in any arbitrary way. However, there is no way around exactly specifying these transformations.

Since these models are linear in the parameters $\beta_0, \beta_1, ..., \beta_p$, there is (under some mild conditions) an analytical and unique solution if the OLS algorithm is used.

Solution 2: GAM

A ***Generalized Linear Model*** is based on the following:

$$\begin{aligned} y_i &= \beta_0 + f_1(x_{i1}) + f_2(x_{i2}) + \ldots + f_p(x_{ip}) + E_i \\ &= \beta_0 + \sum_{j=1}^{p} f_j(x_{ij}) + E_i \end{aligned}$$

Here, $f_j(\cdot)$ are smooth, flexible, 1-dimensional functions that don't need to be explicitly defined by the user, but can be determined from the data in an explorative fashion.

There are several approaches to determine the $f_j(\cdot)$. Some are better, some are worse. The most popular approach is based on cubic splines, as explained on the next few slides…

Simple (1-dimensional) GAM

We first explain the concept in 1-dimension, i.e. we only require to fit $f_j(\cdot)$. This is somewhat similar to smooting, but here we actually require a formula and not just visualization.

A very powerful approach is to express $f_j(\cdot)$ using some simple basis functions (i.e. transformations of x_j):

$$f_j(x_j) = \sum_{m=1}^{M} \gamma_m h_m(x_j)$$

Here, γ_m are some unknown coefficients that are to be estimated from data. Moreover, $h_m(\cdot)$ are arbitrary but explicitly specified basis functions. The choice of M and the complexity of $h_m(\cdot)$ controls the fit to the data.

Polynomial Basis Functions

A simple, yet intuitive choice for the basis functions $h_m(\cdot)$ is given by powers of x_j, i.e. fitting a polynomial. In particular:

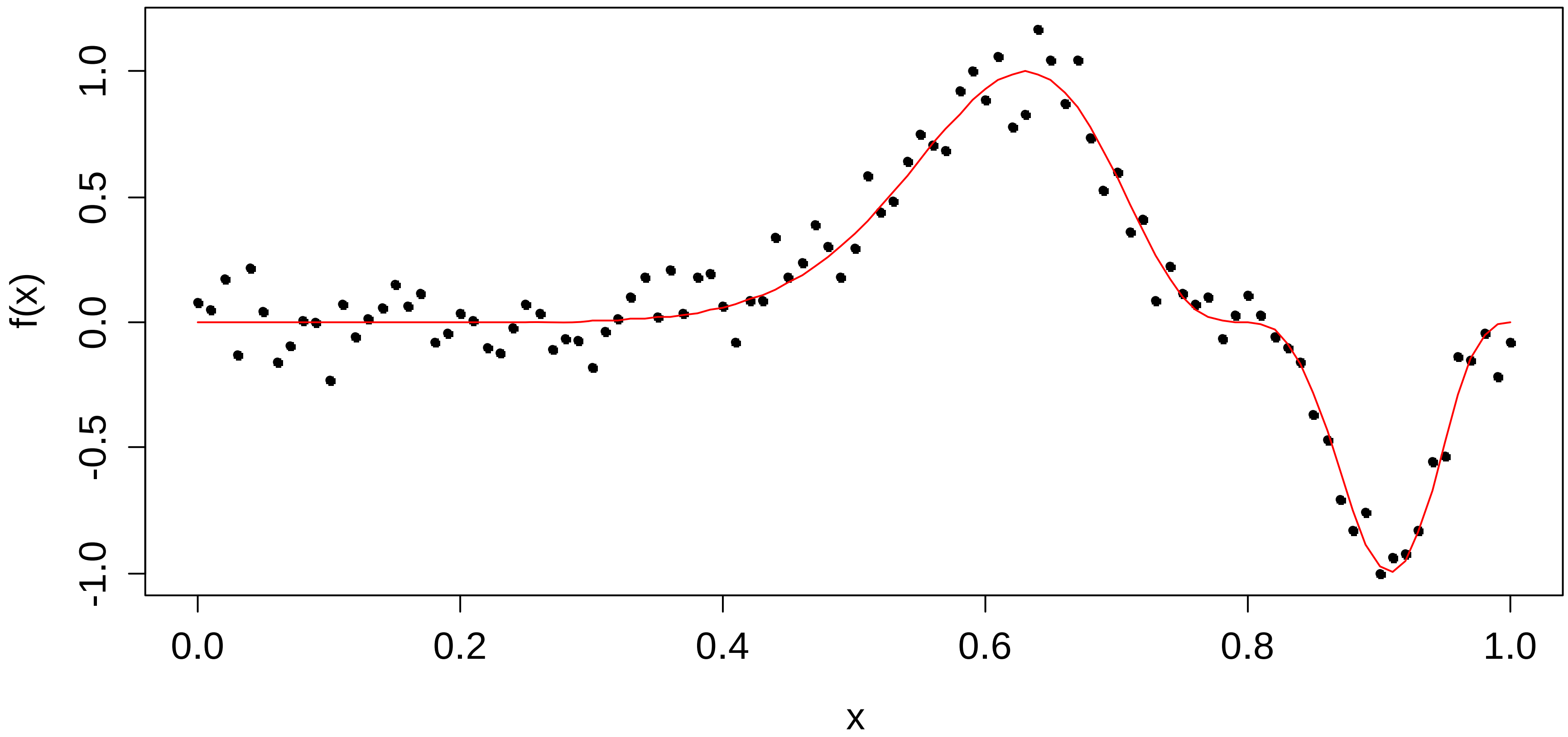
$$h_m(x_j) = x_j^m \text{ , resp. } f_j(x_j) = \sum_{m=1}^{M} \gamma_m x_j^m = \gamma_1 x_j + \ldots + \gamma_M x_j^M$$

Polynomial basis functions have the following properties:

- They allow for a flexible, data-adaptive fit!!!
- Since each of the basis functions $h_m(x_j) = x_j^m$ extends over the entire range of predictor x_j, we may observe some erratic behavior, especially at the boundaries.
- Some simulations results illustrate these drawbacks…

Example 1

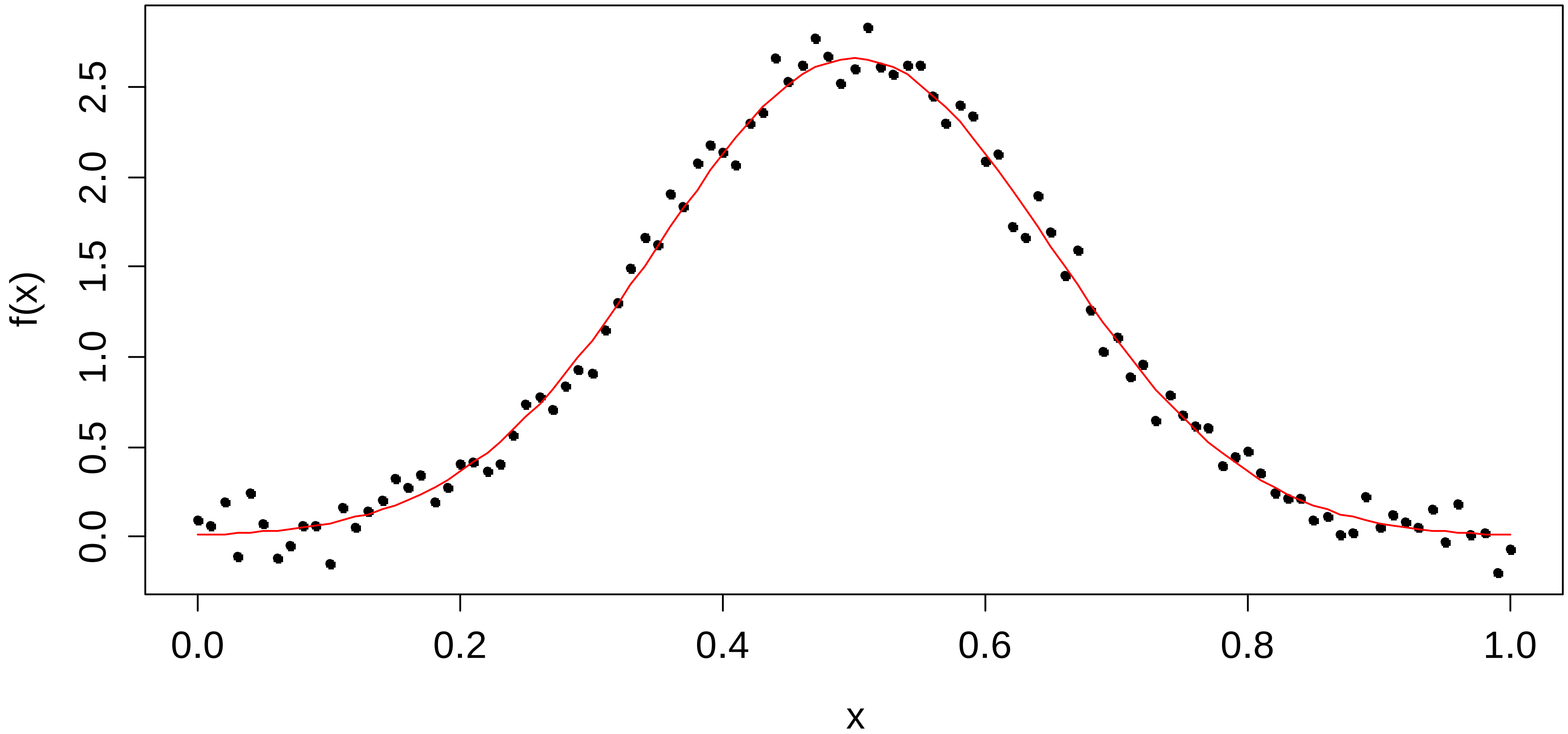
True functional relation: $y = \left(\sin(2\pi x^3)\right)^3 + E$



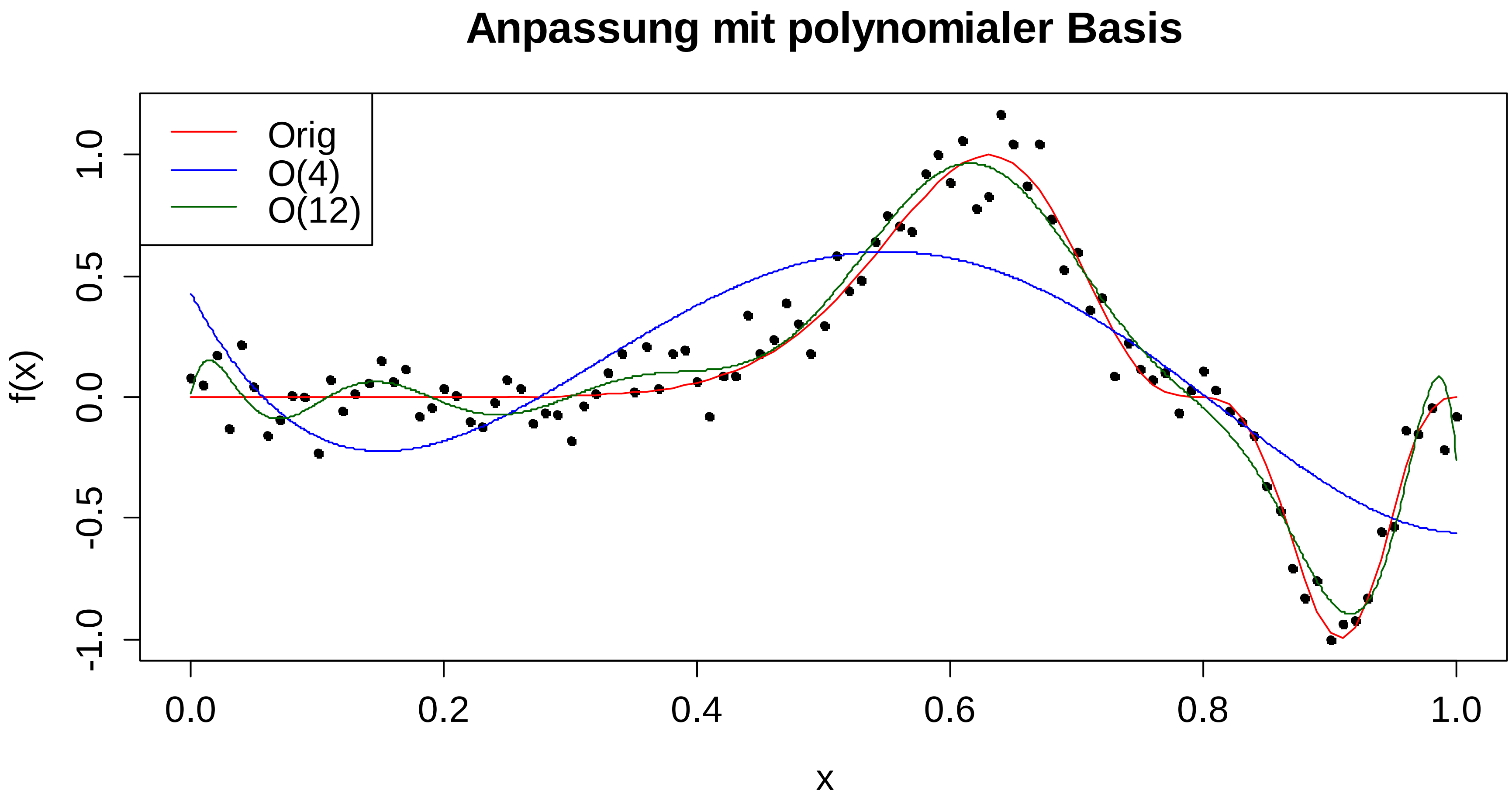
Applied Statistical Regression
AS 2015 – Generalized Additive Modelling

Example 2

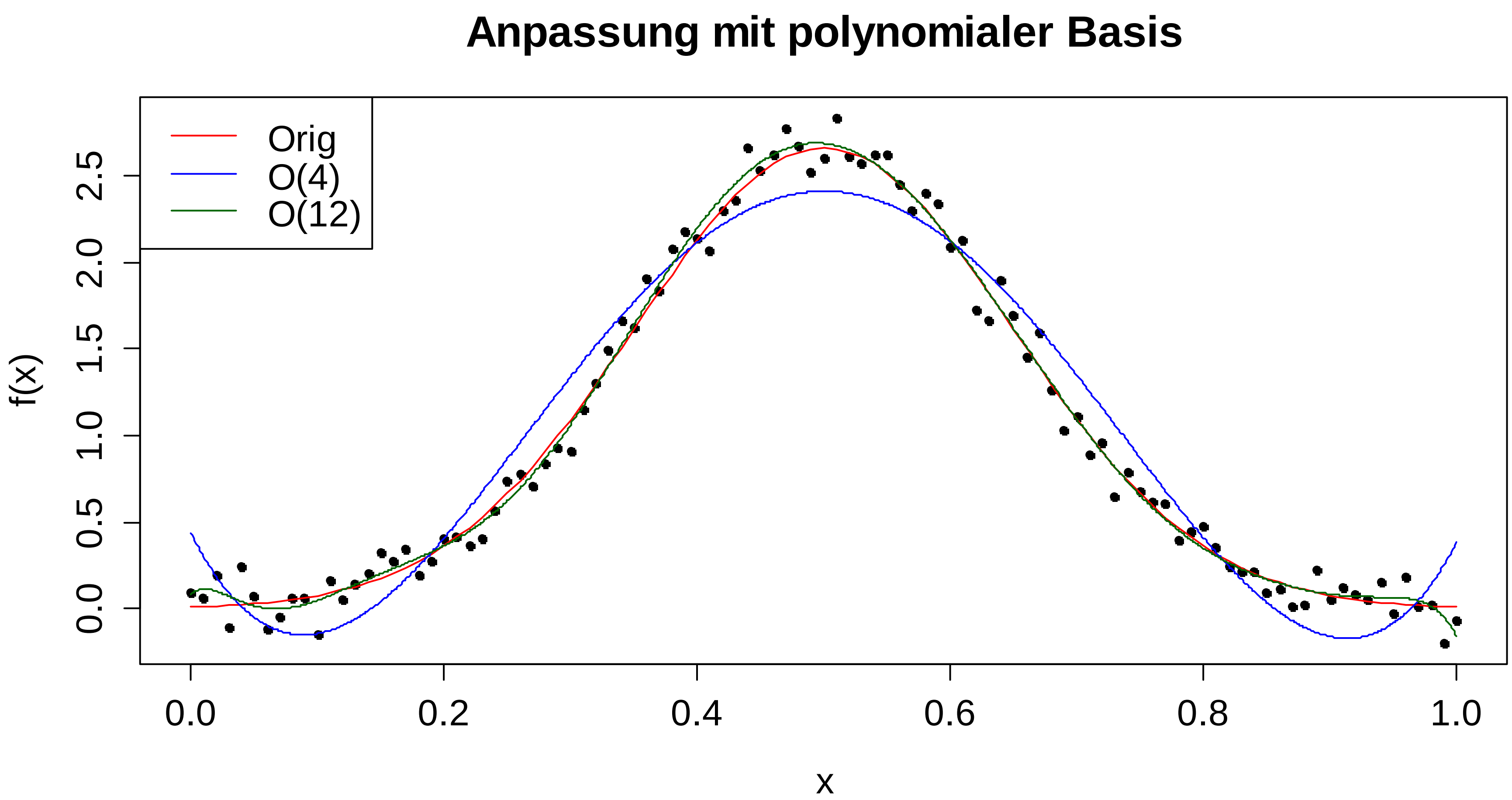
True function relation: Density function of $N(0.5, 0.15^2)$



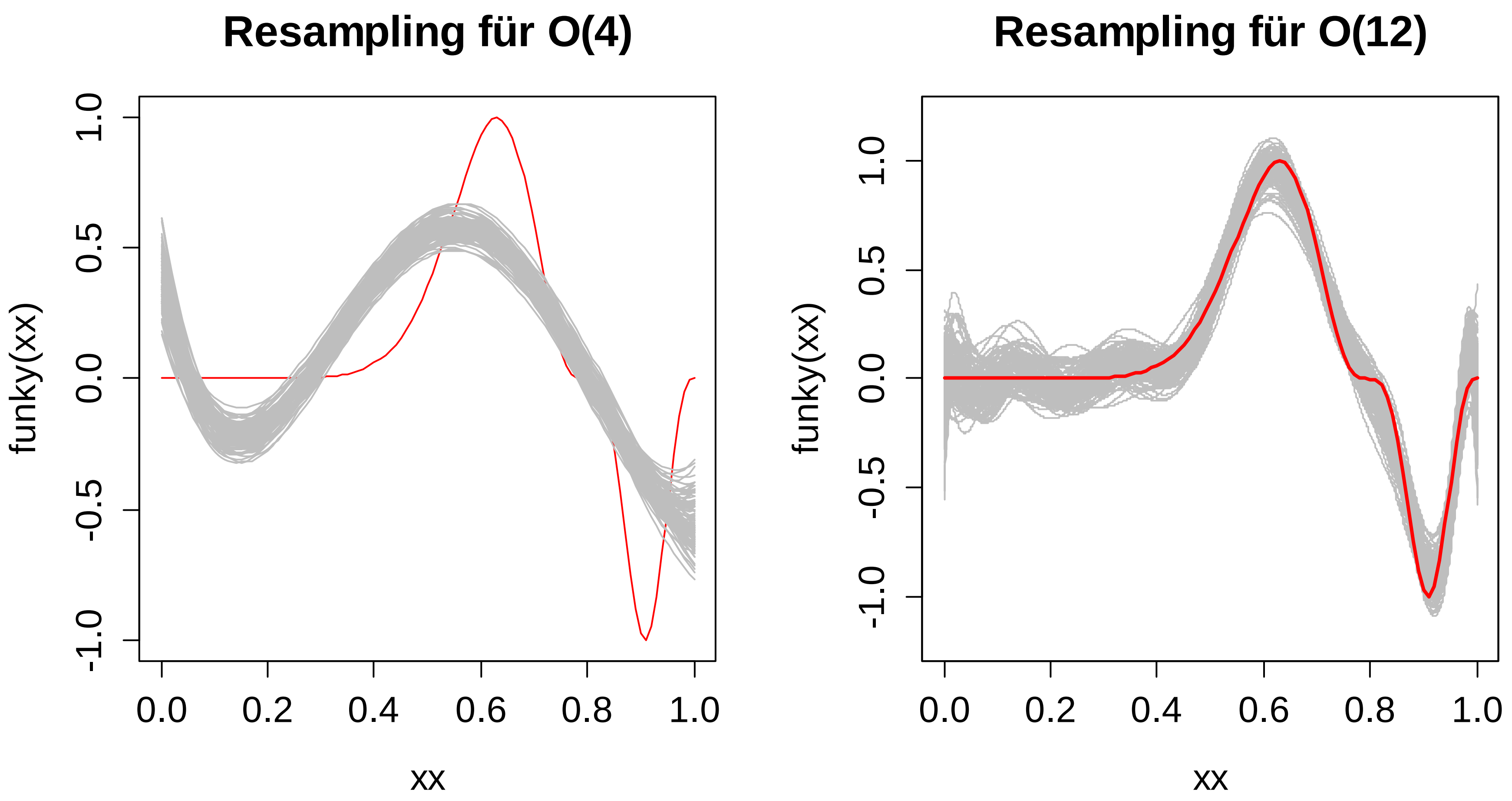
Smoothing with Polynomial Basis Functions



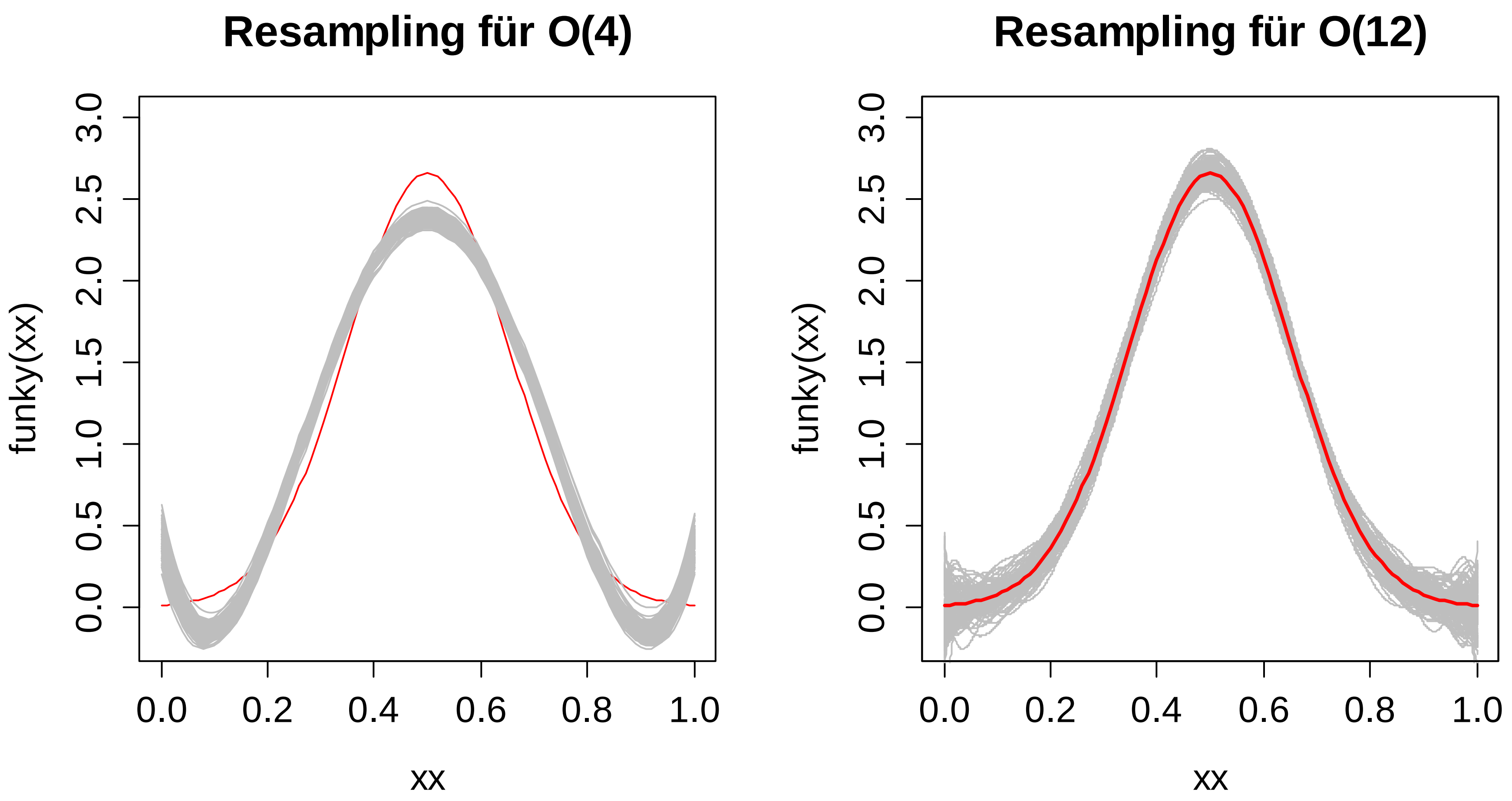
Smoothing with Polynomial Basis Functions



Resampling on Example 1



Resampling on Example 2



What is a Better Alternative?

As the simulation results have shown us, using polynomial basis functions has some severe drawbacks and will not results in a fruitful generalized multiple regression approach.

Idea: why not using basis functions that minimize

$$\sum_{i=1}^{n}(y_i - f_j(x_{ij}))^2 + \lambda \int_{-\infty}^{+\infty} f''(u)du$$

This criterion implements a trade-off between goodness-of-fit and smoothness of the function. Attractive, but how can we find a solution?

→ The solution will always be a cubic B-spline!!!

Regression Splines

We define a basis consisting of cubic B-splines on the interval $[a,b]$ by imposing the following conditions on the knots, which are fixed at the observations $x_{1j},...,x_{nj}$:

1) Each of the basis functions must be different from zero only over a range of 4 knots, so that its influence remains local.

2) The basic form of $h_m(\cdot)$ is a local polynomial of third order.

3) These basis functions are twice continuously differentiable at each of the knots. This implies smoothness of the fit consisting of numerous local functions.

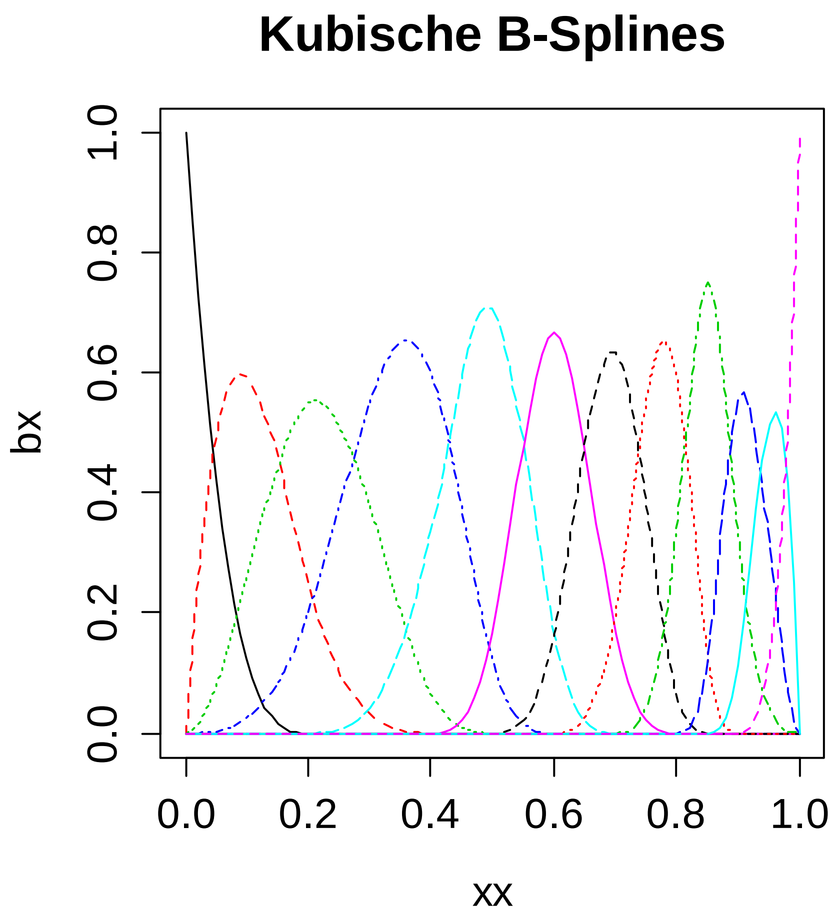
4) The integral over all basis functions shall be equal to 1.

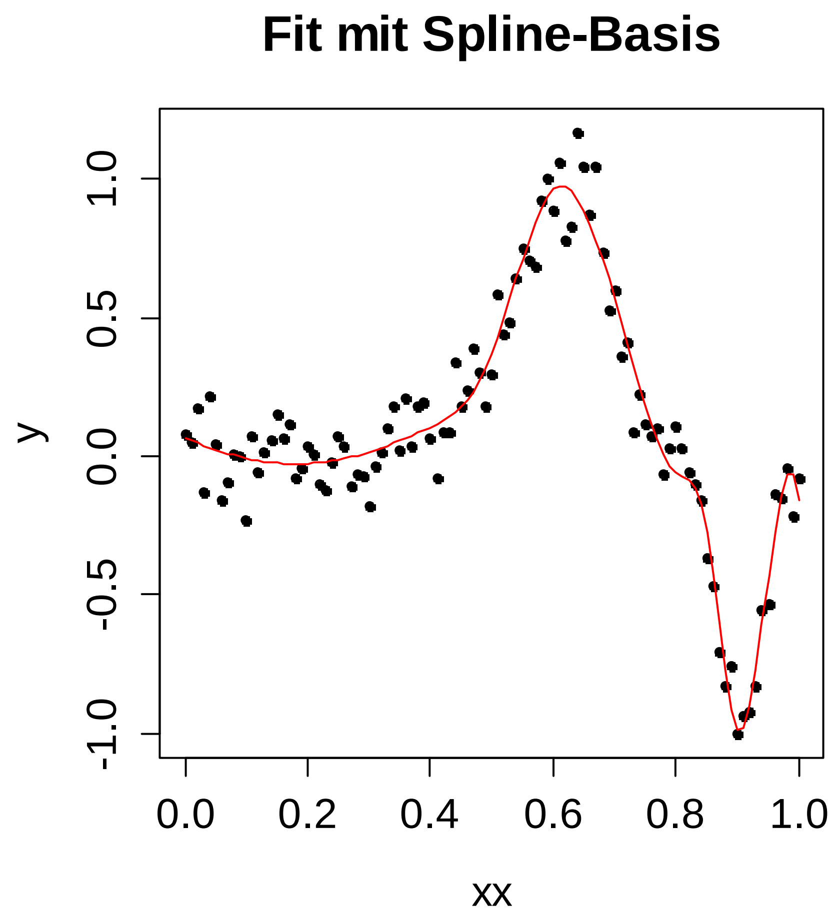
Generating a Spline Basis

In R, such a regression spline basis can be generated conveniently:

```
> set.seed(21)
> library (splines)
> funky <- function(x) sin(2*pi*x^3)^3
> xx <- seq (0, 1, by=0.01)
> yy <- funky(xx) + 0.1*rnorm (101)
> kn <-c(0,0,0,0,.2,.4,.5,.6,.7,.8,.85,.9,1,1,1,1)
> bx <- splineDesign (kn, xx)
> gs <- lm (yy ~ bx)
> matplot(xx, bx, type="l")
> matplot(xx, cbind (yy, gs$fit), type="pl")
```

Spline Basis and Resulting Fit





GAM Using a Spline Basis

In practice, we will rarely be satisfied with simple models, but require fitting multiple predictor GAMs. The idea is as follows:

$$y_i = \beta_0 + f_1(x_{i1}) + f_2(x_{i2}) + \ldots + f_p(x_{ip}) + E_i$$

The principle is that for each predictor x_j, we will have a flexible and exploratively determined contribution $f_j(\cdot)$ that is rooted on a basis consisting of cubic B-splines with correct complexity. There is an excellent implementation in R…

→ How can this model be estimated?

→ How can one determine the correct smoothness of $f_j(\cdot)$?

Backfitting-Algorithm

There is no single step solution to a multiple predictor GAM. We pursue an iterative approach that is based on stepwise solution of 1-dimensional problems:

1) Initialize $\hat{\beta}_0 = \bar{y}$ and $f_j(\cdot) = 0$ for all $j = 1, \ldots, p$

2) Repeat for all $j = 1, \ldots, p$ until convergence:
   - Compute $\tilde{y}_i = y_i - \hat{\beta}_0 - \sum_{k \neq j} f_k(x_{ik})$
   - Solve the 1-dimensional problem for $\hat{f}_j(\cdot)$ on $(x_{ij}, \tilde{y}_i)$
   - Center $\hat{f}_j(\cdot) \leftarrow \hat{f}_j(\cdot) - n^{-1} \sum_i \hat{f}_j(x_{ij})$

Note: the solution will only be identifiable if $\sum_i \hat{f}_j(x_{ij}) = 0$

Implementation in `library(mgcv)`

- The backfitting-algorithm and in particular R function `gam()` also allows for having parametric terms in the model.

- The estimation in R package `mgcv` is not based on the backfitting algorithm specified above, but on the more sophisticated Lanczos approach (w/o details here…).

- Syntax: `fit <- gam(resp ~ s(p1) + s(p2) + p3, data=ex)`

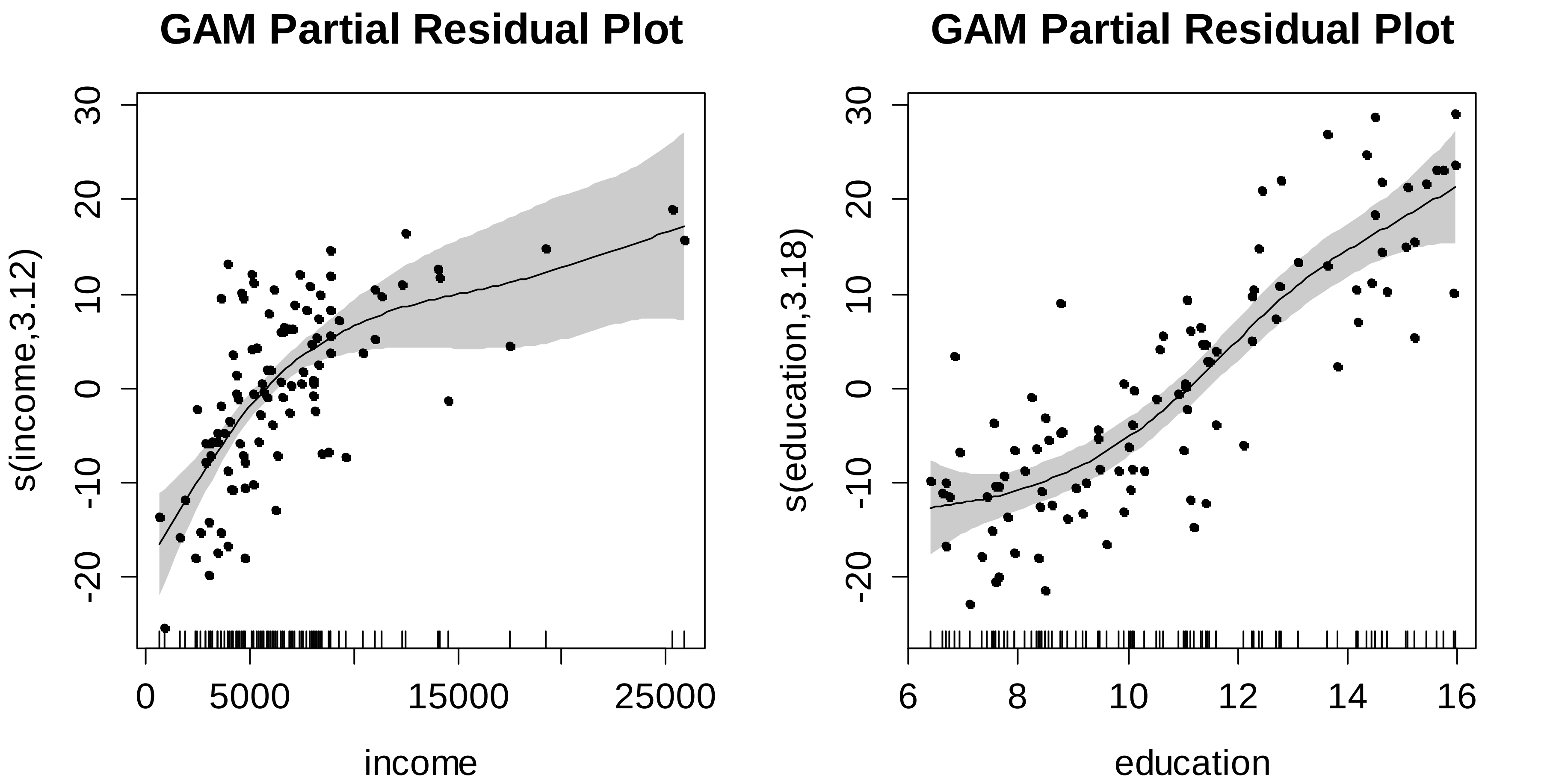
- The complexity of the spline basis for each component will be estimated exploratively using cross validation. It may be overruled by typing `s(p1, df=…)`.

Example: Prestige Data

```
> fit <- gam(prestige ~ s(income) + s(education), data=…)
> summary(fit)
Family: gaussian; Link function: identity
Formula: prestige ~ s(income) + s(education)
---
Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  46.8333     0.6889   67.98   <2e-16 ***
---
Approximate significance of smooth terms:
               edf Ref.df     F  p-value
s(income)    3.118  3.877 15.29 8.94e-10 ***
s(education) 3.177  3.952 38.34  < 2e-16 ***
---
R-sq.(adj) =  0.836   Deviance explained = 84.7%
GCV = 52.143  Scale est. = 48.414    n = 102
```

Example: Partial Residual Plots

```
> plot(fit, shade=TRUE, residuals=TRUE, pch=20, main=…)
```

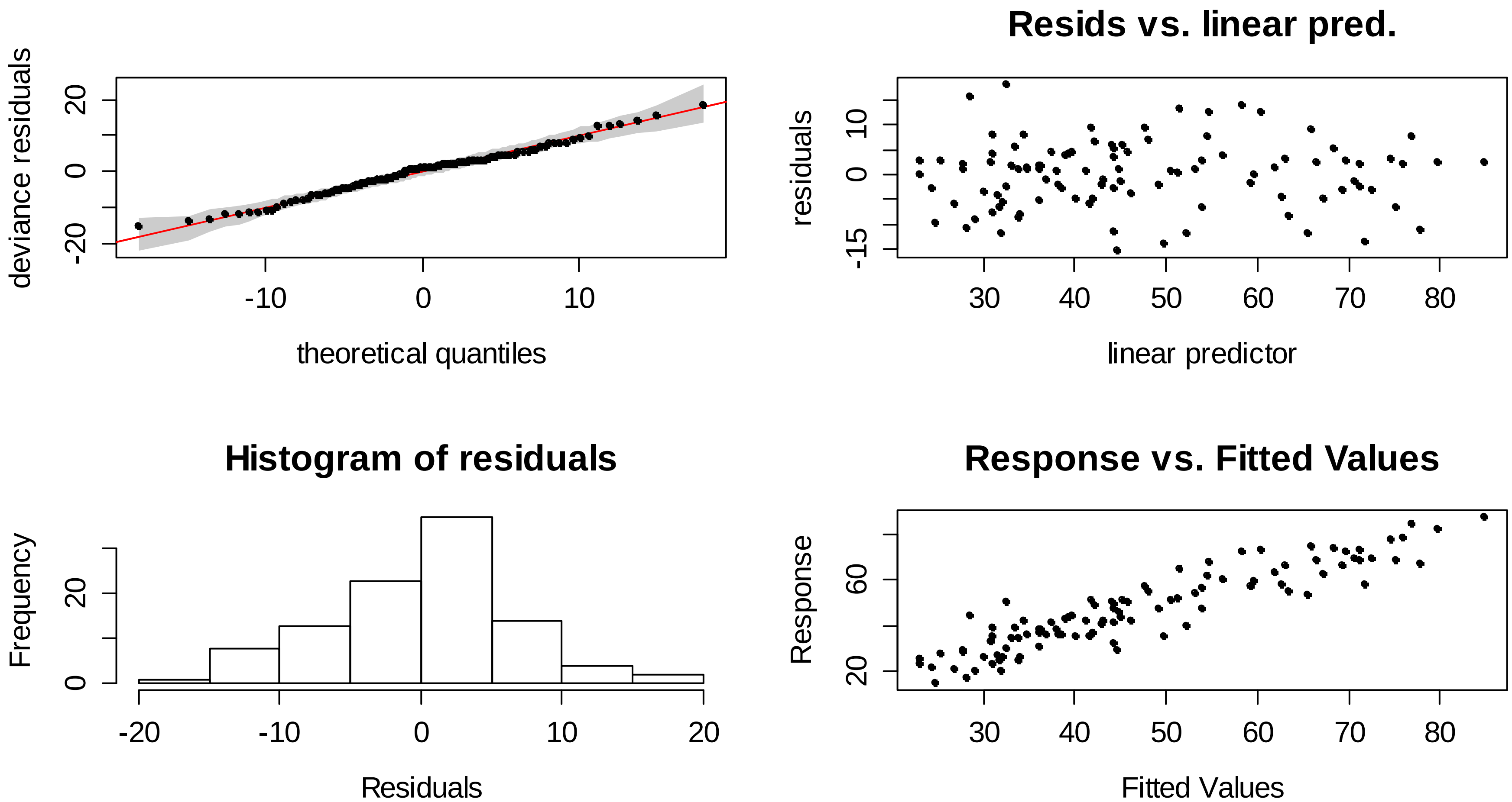


Example: Residual Analysis

```
> gam.check(fit, pch=20, rep=100)

Method: GCV   Optimizer: magic

Smoothing parameter selection converged after 4 iterations
The RMS GCV score gradiant at convergence was 9.783945e-05
The Hessian was positive definite.

The estimated model rank was 19 (maximum possible: 19)
Model rank =  19 / 19

Basis dimension (k) checking results.
Low p-value (k-index<1) may indicate that k is too low,
especially if edf is close to k'.

                k'   edf k-index p-value
s(income)    9.000 3.118   0.981    0.36
s(education) 9.000 3.177   1.025    0.61
```

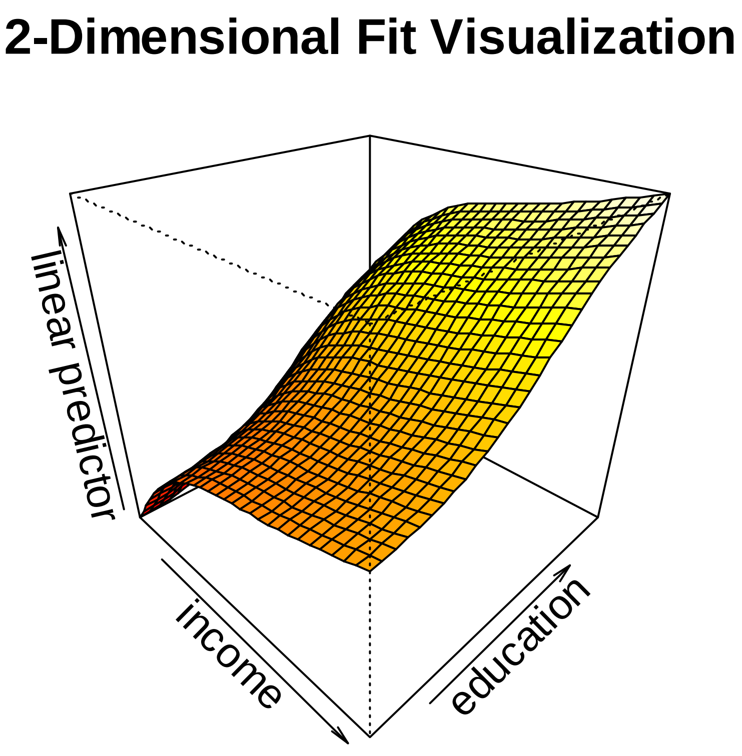
Example: Residual Analysis

```
> gam.check(fit, pch=20, rep=100)
```



Example: Visualizing the Fit

```
> vis.gam(fit, theta=45, phi=30)
```



Note: both predictors contribute in a non-linear fashion, but model is additive!

Testen for Linearity

Function `gam()` determines the degrees of freedom for each of the predictors data-adaptively. If no flexibility is required, we can obtain `df=1`. In that case, the predictor contributes linearly.

However, in many situations one may be interested in formally testing whether a GAM yields a better fit than using OLS. This can be done on the basis of a test that gauges RSS versus the degrees of freedom of the respective models.

```
> fit
Estimated degrees of freedom:
3.12 3.18  total = 7.3
```

The GAM for the Prestige data spends 7.3 degrees of freedom. The competing OLS model only takes 3 of them!!!

Testen for Linearity

```
> fit.ols <- gam(prestige ~ income + education, data=…)

Family: gaussian; Link function: identity
Formula: prestige ~ income + education
Total model degrees of freedom 3
GCV score: 62.84693

> deviance(fit.ols)
[1] 6038.851

> dd <- deviance(fit.ols)-deviance(fit); dd
[1] 1453.856

> 1-pchisq(dd, 7.3-3)
[1] 0
```

The GAM has a highly significant edge on OLS. However, we need to use variable transformations in the OLS model.

Testen for Linearity

There is some alternative (better) functionality that carries out the test for linearity as a one-line-command:

```
> anova(fit.ols, fit, test="Chisq")
Analysis of Deviance Table

Model 1: prestige ~ income + education
Model 2: prestige ~ s(income) + s(education)
  Resid. Df Resid. Dev     Df Deviance  Pr(>Chi)
1    99.000     6038.9
2    94.705     4585.0 4.2951   1453.9 6.783e-06 ***
```

As we can see, the computed value for the test statistic is identical to the one one the previous slide. There is some rounding-based difference in the p-value, though.

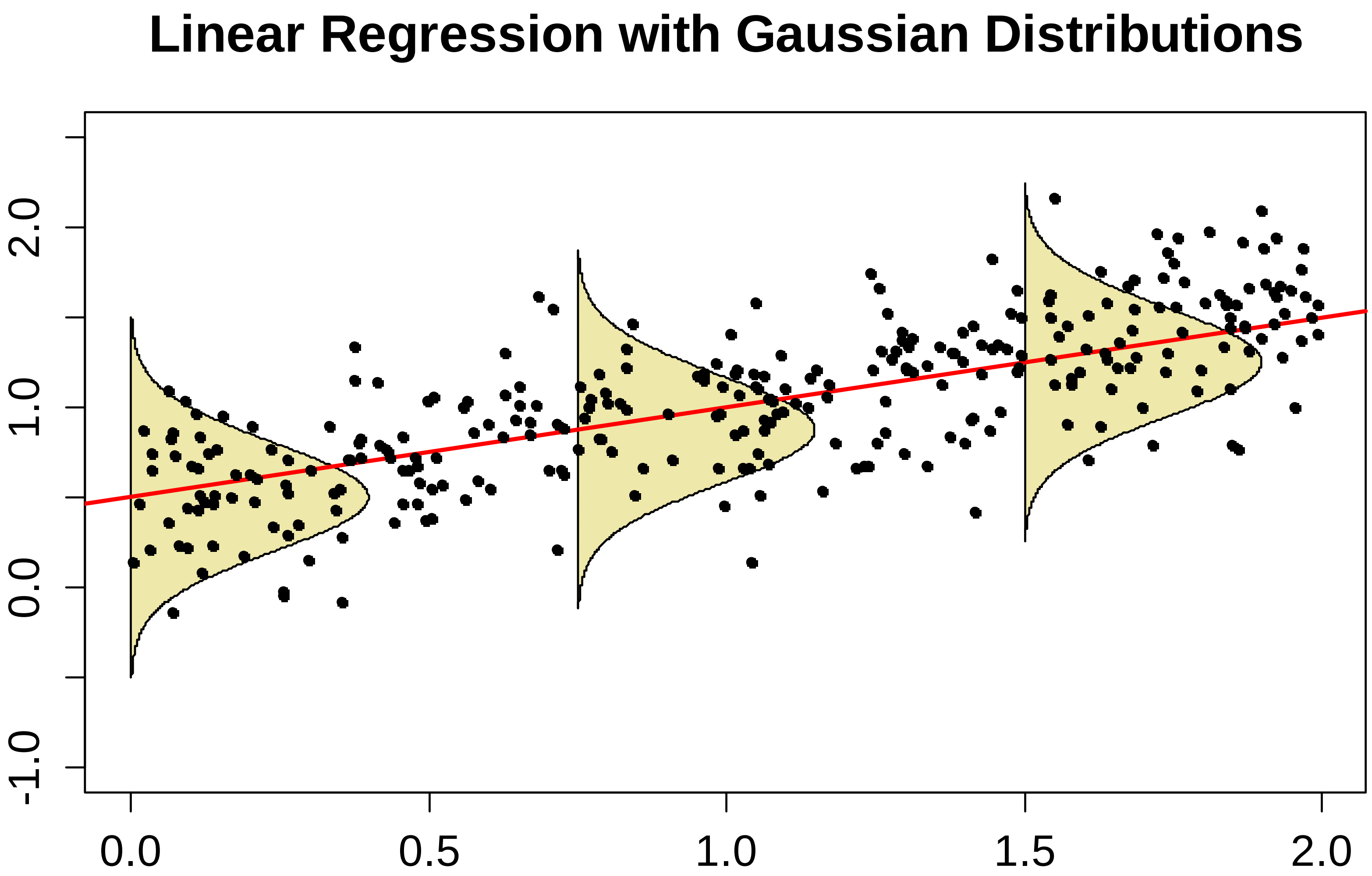
Non-Numerical Response Variable

→ **So far, the response y_i was a continuous random variable with infinite range, where the conditional distribution was a Gaussian, i.e. $y_i \mid X_i \sim N(\hat{y}_i, \sigma_E^2)$, see next slide.**

→ If the task is modeling binary, binary or multinomial response (i.e. probabilities or proportions) or a count, this is not doable correctly with the methods that were discussed yet.

→ We will present some additional techniques which implement linear modeling for these different types of responses. As we will see, there is a generic framework that incorporates all of these, as well as multiple linear regression.

Conditional Gaussian Distribution



Binary Response / Logistic Regression

What is the question?

In toxicological studies, one tries to infer wheter a lab mouse survives when it is given a particular dose of poisonous matter.

In human medicine, one is often interested in the question how much of a drug is required to see an effect, i.e. pain reduction.

Mathematics:

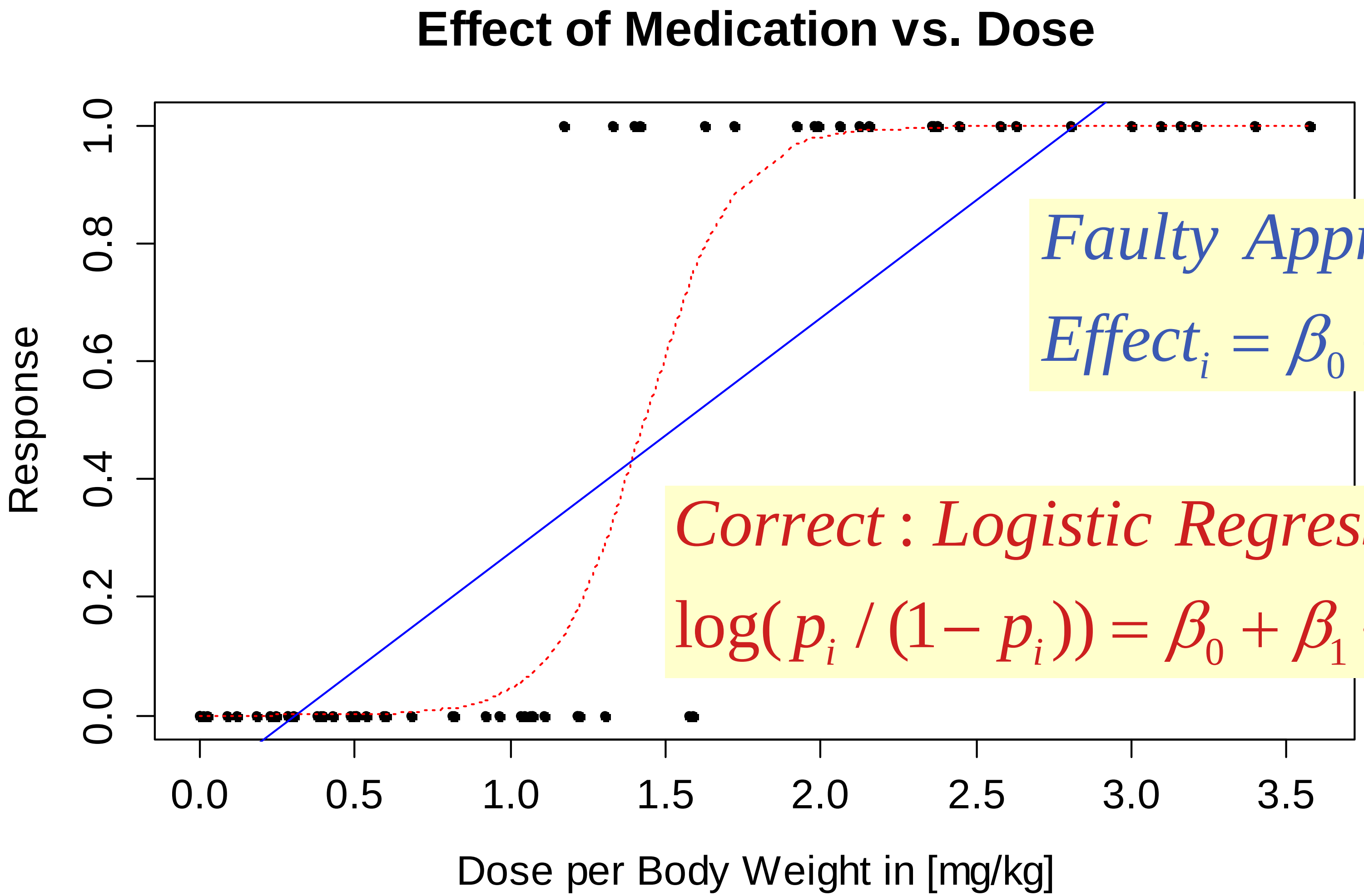
→ The response variable $y_i \in \{0,1\}$ is binary

→ The conditional distribution $y_i \mid X_i \sim Bernoulli(p_i)$

→ The fitted value is the expectation of the above conditional distribution, and hence the probability of death/survival p_i.

Applied Statistical Regression
AS 2015 – Generalized Linear Modeling

Binary Response / Logistic Regression



Faulty Approach:

$$Effect_i = \beta_0 + \beta_1 \cdot Dose_i + E_i$$

Correct: Logistic Regression

$$\log(p_i / (1 - p_i)) = \beta_0 + \beta_1 \cdot Dose_i$$

Count Response / Poisson Regression

What are predictors for the locations of starfish?

→ analyze the number of starfish at several locations, for which we also have some covariates such as water temperature, ...

→ the response variable is a count. The simplest model for this assumes a Poisson as the conditional distribution.

We assume that the logged parameter λ_i at location *i* depends in a linear way on the covariates:

$y_i \mid X_i \sim Pois(\lambda_i)$, where $\log(\lambda_i) = \beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}$

Generalized Linear Models

What is it?

- General framework for regression type modeling
- Many different response types are allowed
- Notion: the responses' conditional expectation has a monotone relation to a linear combination of the predictors.

$$g(E[y_i \mid X_i]) = \beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}$$

- Some further requirements on variance and density of y

→ **may seem complicated, but is very powerful!**