Linear and Generalized Linear Models (4433LGLM6Y)

Checking assumptions in Linear Model

Meeting 7

Vahe Avagyan

Biometris, Wageningen University and Research

Overview

- Checking assumptions in Linear Model (Fox, 12.1 -12.2)
- Transformations (Fox, 12.3 -12.4)
- Polynomials and splines (Faraway text, 8.2-8.4)

Overview

- Checking assumptions in Linear Model
- Transformations
- Polynomials and splines

Example: Survey of Labour and Income Dynamics

- SLID data
  - wages: Composite hourly wage rate ($/hour).
  - age: in years.
  - sex: dummy variable, (1=male or 0=female).
  - education: Completed years of education.

```
> head(SLID)
  age    sex wages education
1  40   Male 10.56        15
2  19   Male 11.00        13
3  46   Male 17.76        14
4  50 Female 14.00        16
5  31   Male  8.20        15
6  30 Female 16.97        13
```

```
> SLID <- read.table("SLID-Ontario.txt", header=T)
> slidreg <- lm(wages ~ sex + age + education, data=SLID)
> coef(summary(slidreg))
              Estimate  Std. Error   t value      Pr(>|t|)
(Intercept) -8.1242314 0.598977251 -13.56351  5.267939e-41
sexMale      3.4736704 0.207009203  16.78027  4.038267e-61
age          0.2612932 0.008663968  30.15861 3.424031e-180
education    0.9296491 0.034256733  27.13771 5.472869e-149
> summary(slidreg)$r.squared
[1] 0.3073726
```

Model assumptions

- For a linear model to be a good model, there are four conditions that need to be fulfilled.
  - **Independence**: The residuals are independent of each other.
  - **Linearity:** The relationship between the variables can be described by a linear equation (also called additivity)
  - **Equal variance:** The residuals have equal variance (also called homoskedasticity)
  - **Normality:** The distribution of the residuals is normal

Graphical check of normality

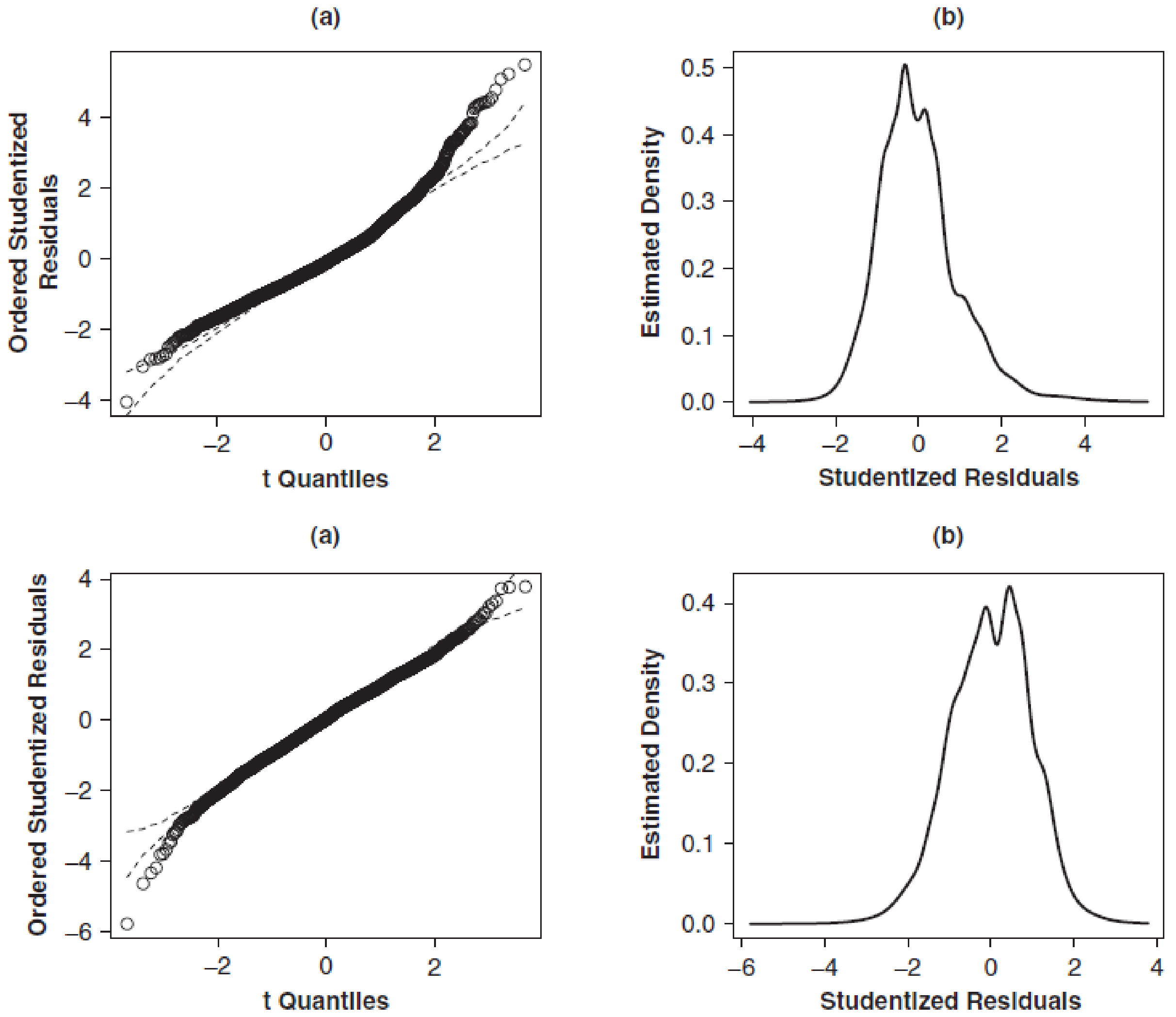
- A quantile comparison plot can give us a sense of which observations depart from normality.

- QQ-plot of residuals
  - Plot studentized residual E_i^* versus normal or t_{n-k-2} distribution.
  - The difference between two is important for small samples.
  - In larger samples, internally studentized residuals or raw residuals will give same impression.
  - QQ-plot effective in displaying tail behavior.

- Histogram or smoothed histogram.
  - The skew may help to chose a transformation.

Graphical check of normality: Examples: SLID regression

- (a) QQ-plot and (b) smoothed histogram of studentized residuals E_i^*.
- First row, not transformed.
- Second row, after the log-transformation.



Nonconstant Error Variance

- Error variance:

$$V(\epsilon) = V(Y|x_1, \ldots, x_k) = \sigma_\epsilon^2$$

Heteroscedasticity = nonconstant error variance.

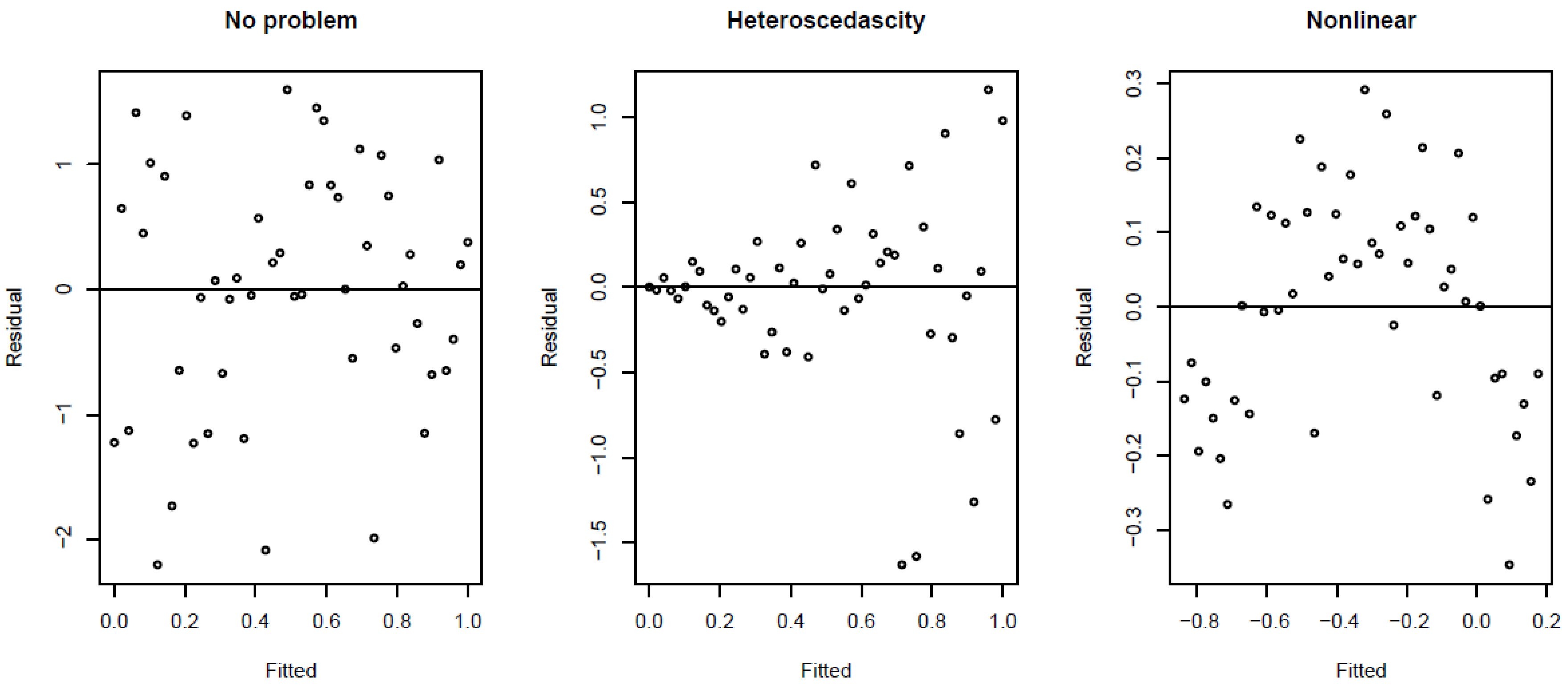
Homoscedasticity = constant error variance.

- **Note**: LS estimator $\mathbf{b}$ remains unbiased and consistent even with nonconstant variance.
- Its efficiency is impaired (we can do better) and usual formulas for standard errors are inaccurate.

- Harm produced by heteroscedasticity is relatively mild. Worry if the largest variance is 4 times the smallest variance (i.e., sd of the errors varies by more than a factor 2).

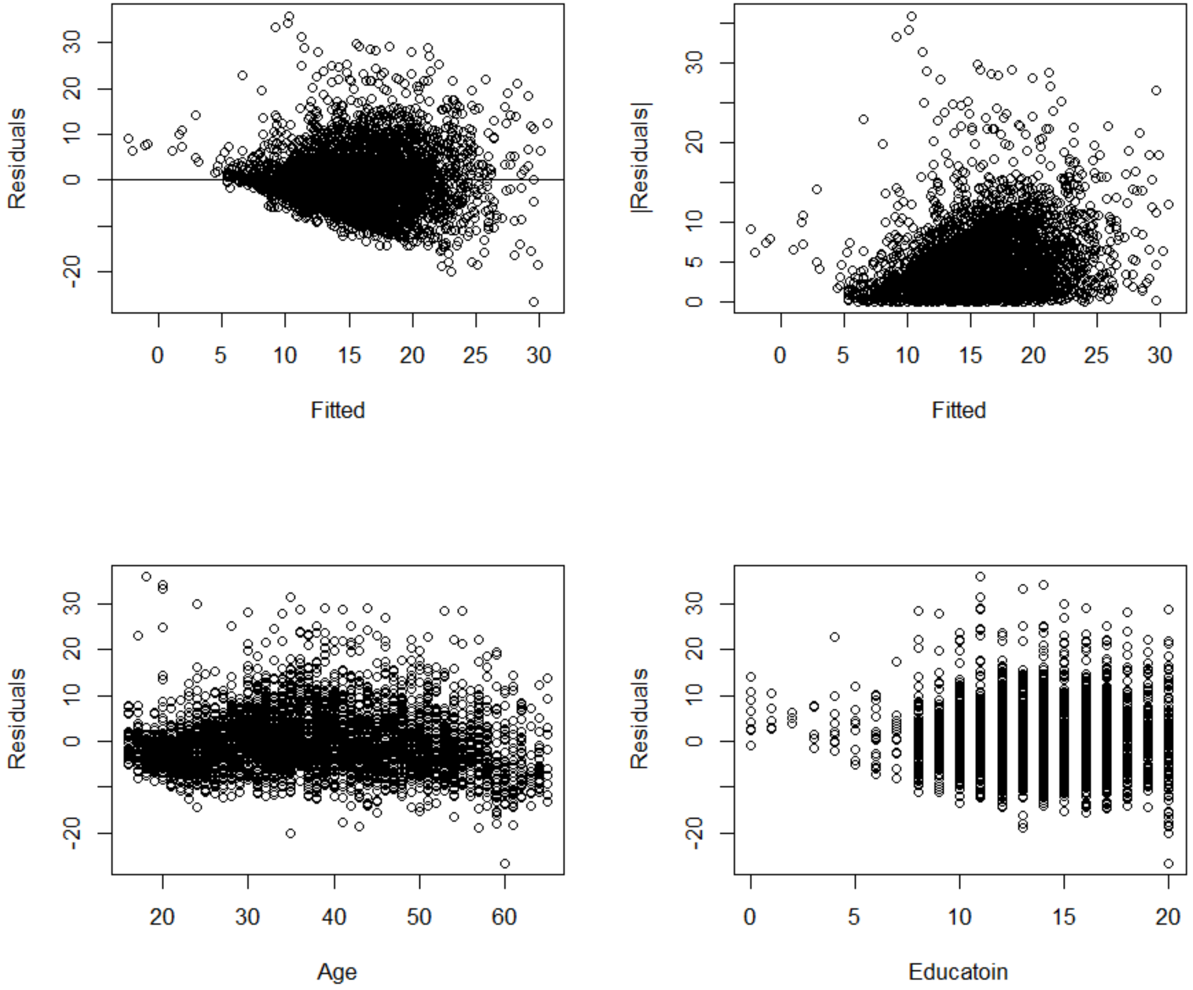
Graphical check of constant variance

- Most important plot: Residual plots - $\mathbf{e}$ vs $\hat{y}$



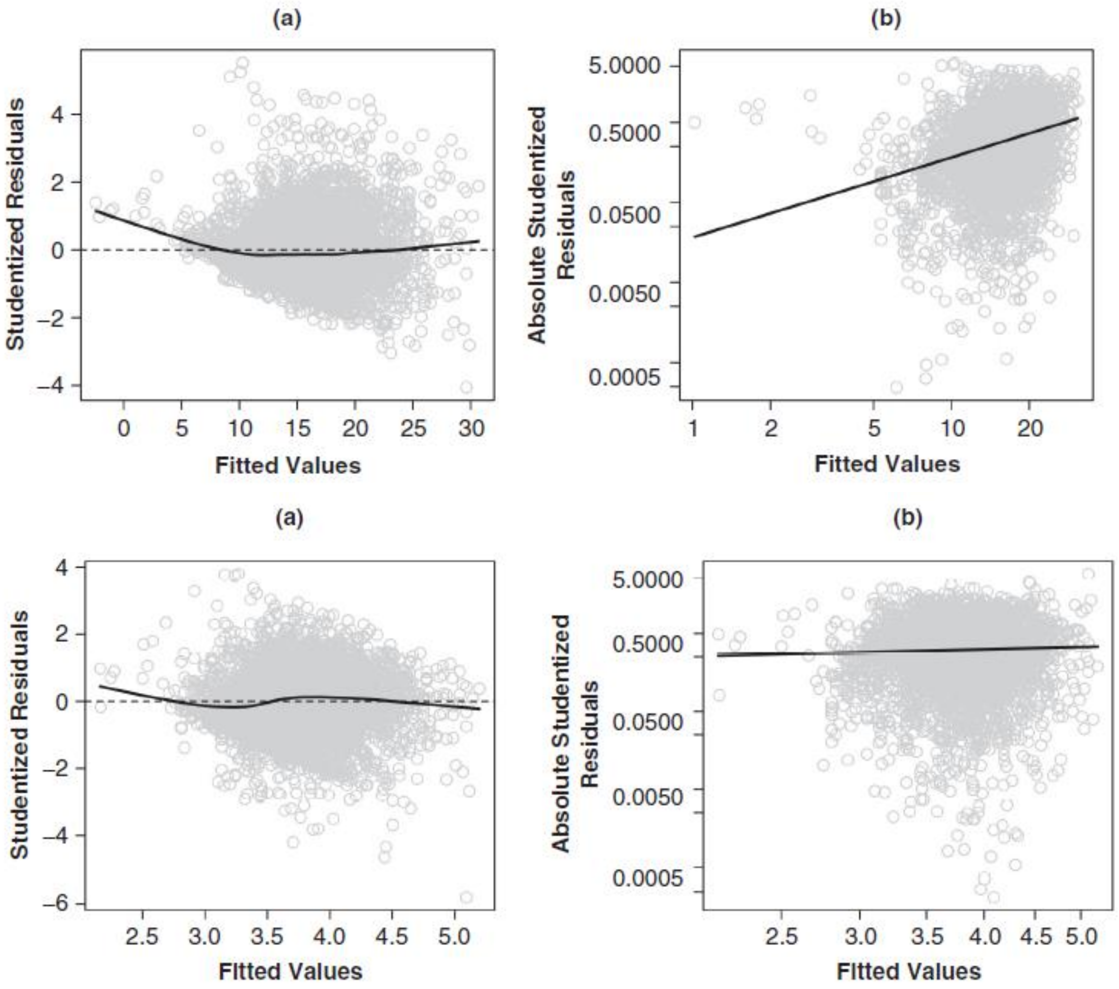
Graphical check of constant variance: Example

- Plot residuals E_i against fitted values $\hat{Y}_i$ (not Y).
- Check for constant variance in vertical direction, and scatter should be symmetric vertically about 0.
- Plot residuals against each X (included or excluded).



Graphical check of constant variance: Example

- Plot studentized residuals E_i^*
- Ordinary residuals have unequal variances, **even with constant error variance**.
- Pattern of changing spread more easily seen by plotting $|E_i^*|$ or E_i^{*2} against $\hat{Y}$.
- First row: not transformed.
- Second row: log-transformed.

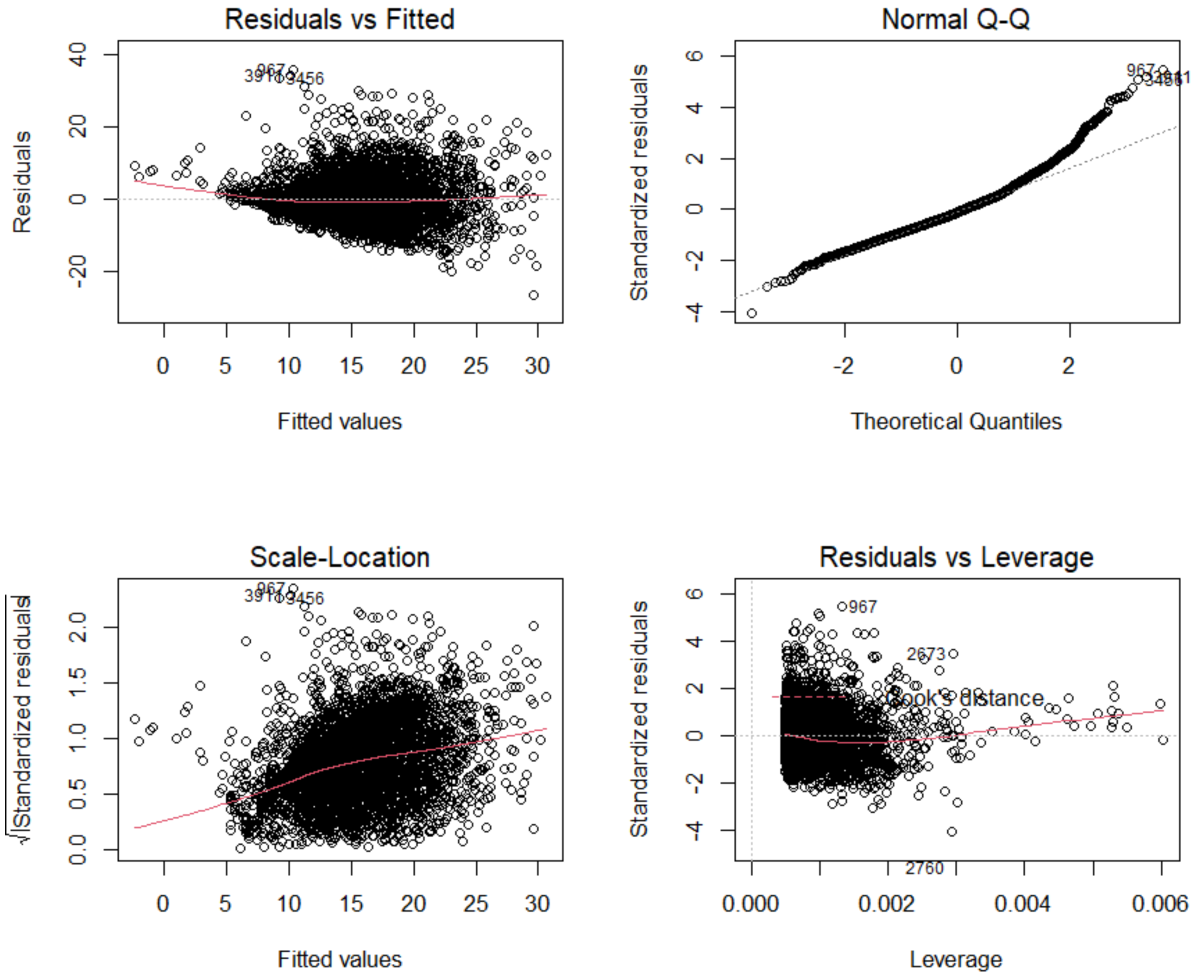


Graphical check of constant variance: Example

- **R** provides default residual diagnostics with plot() function.

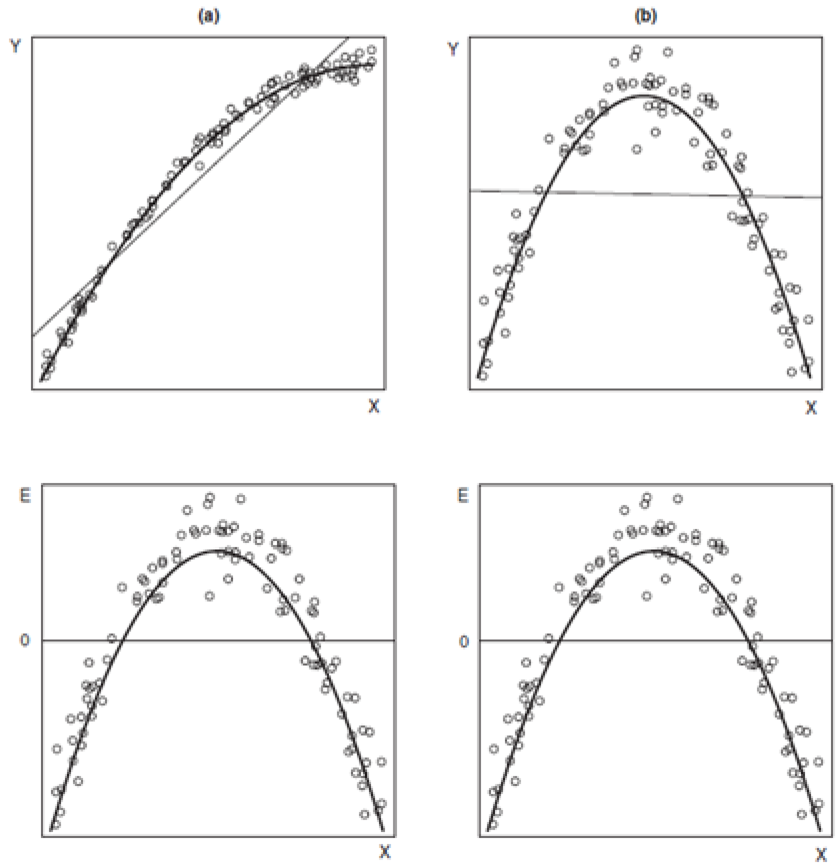
```
> plot(slidreg)
```

1. E_i versus $\hat{\mathbf{y}}_i$
2. Normal QQ-plot for E_i'
3. $\sqrt{|E_i'|}$ versus $\hat{\mathbf{y}}_i$
4. E_i' versus h_i



Nonlinearity

- $E(\epsilon) = 0$ implies that regression surface accurately reflects the dependency $E(Y|X_i)$.
- Regression surface is generally high dimensional.
- Focus on certain patterns of departure from linearity.
- Graphical diagnostics: cloud plotting or Y vs X_i.



- The residual based plots maybe more informative.
- Monotone and non-monotone nonlinearity.

Component-Plus-Residual Plots (partial residual plots)

Residual of the main regression model

- Partial residual for j-th regressor

$$E_i^{(j)} = E_i + B_j X_{ij} :$$

i.e., add back linear component of partial relationship between Y and X_j to E_i.

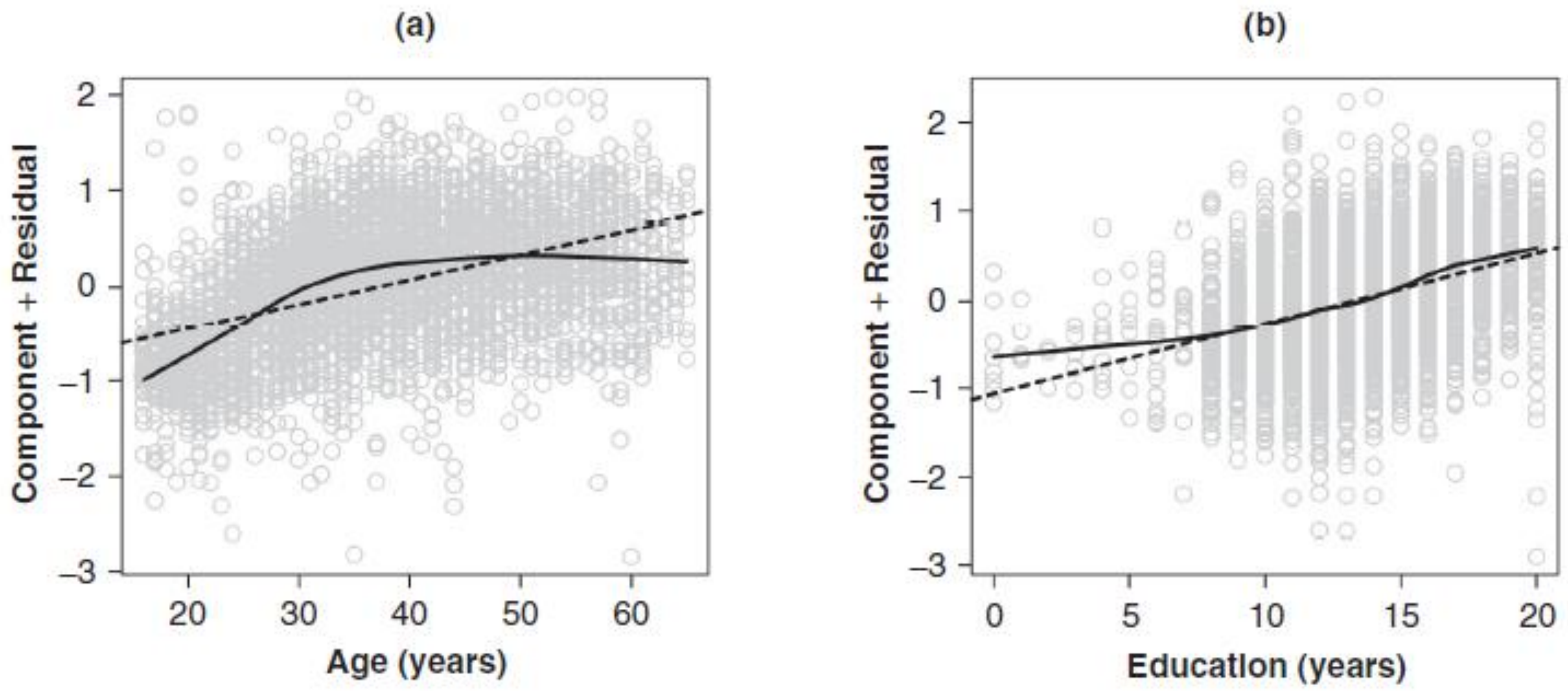
- Plot $E^{(j)}$ versus X_j .

- Multiple regression coefficient B_j is the slope of simple regression of $E^{(j)}$ on X_j.

- Nonlinearity may be apparent in the plot.

Component-Plus-Residual Plots: Example

- SLID regression: the solid lines show the lowess smooths, the broken lines are least-squares fits.



Overview

- Checking assumptions in Linear Model
- Transformations
- Polynomials and splines

Transformations (response variable)

- Variable transformation may help to address possible violations of the assumptions:
  - Log transformation $Y \to \ln Y$
  - Power transformation: $Y \to Y^{\lambda}$ (parameter of transformation)
- We can obtain $\hat{\lambda}_{MLE}$ using the Maximum Likelihood Estimation.
- Check the hypothesis

Which λ_0 means no transformation?

$$H_0 : \lambda = \lambda_0.$$

- Likelihood Ratio Tests, Wald test, Score test.

Transformations: Box-Cox

- The aim of the Box-Cox transformations is to ensure the usual assumptions for Linear Model hold.

$$Y_i^{(\lambda)} = \beta_0 + \beta_1 X_{i1} + \cdots + \beta_k X_{ik} + \epsilon_i,$$

where

$$Y_i^{(\lambda)} = \begin{cases} \dfrac{Y_i^{\lambda} - 1}{\lambda}, & \text{for } \lambda \neq 0 \\ \ln Y_i, & \text{for } \lambda = 0 \end{cases}$$

- Log-transformation is a particular case of Box-Cox (i.e., $\lambda = 0$).

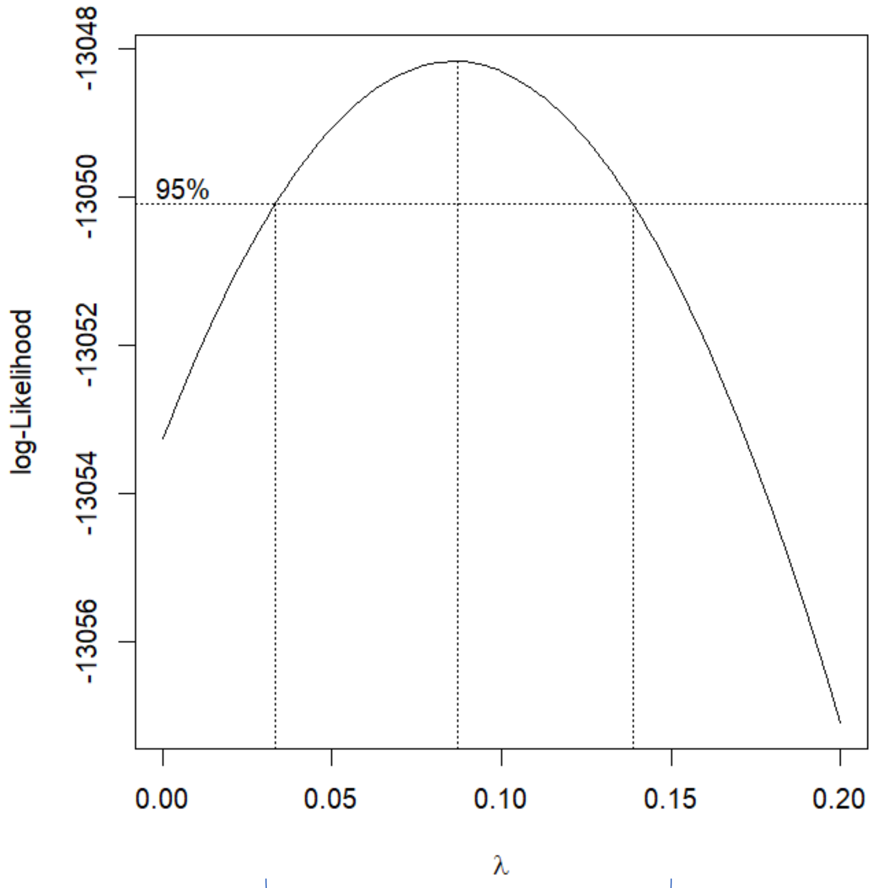
- Which λ means no transformation?

Maximum Likelihood Estimation

- SLID regression:

```
> library(MASS)
> bc <- boxcox(slidreg,plotit=T,lambda=seq(0,0.2,by=0.01))
> # Exact lambda
> lambda <- bc$x[which.max(bc$y)]
> lambda
[1] 0.08686869
> bc_wages <- (SLID$wages^lambda - 1) / lambda
```

- We can see a strong reason to transform (why?).



$\mathrm{CI}_{0.95}(\lambda)$

Overview

- Checking assumptions in Linear Model
- Transformations
- Polynomials and splines

Transformations (predictors)

- Generalizing the $\mathbf{X}\boldsymbol{\beta}$ part of the model by adding polynomial terms (e.g., one-predictor case):

$$y = \beta_0 + \beta_1 X + \cdots + \beta_d X^d + \epsilon$$

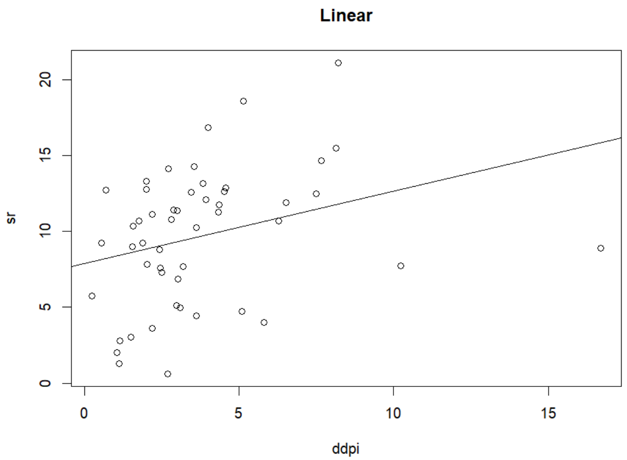
- Selection of d.
    1. Keep adding terms until the added term is not statistically significant.
    2. Start with a large d , eliminate not significant terms starting with the highest order term.

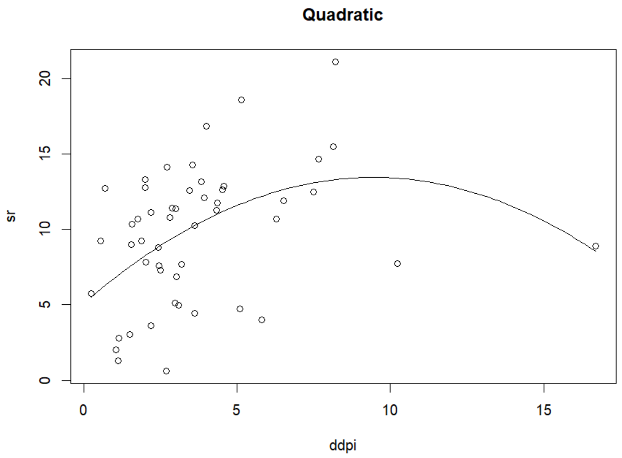
- Polynomial regression allows for more flexible relationship
- Principle of marginality: do not remove lower order terms from model, even if they are not statistically significant.

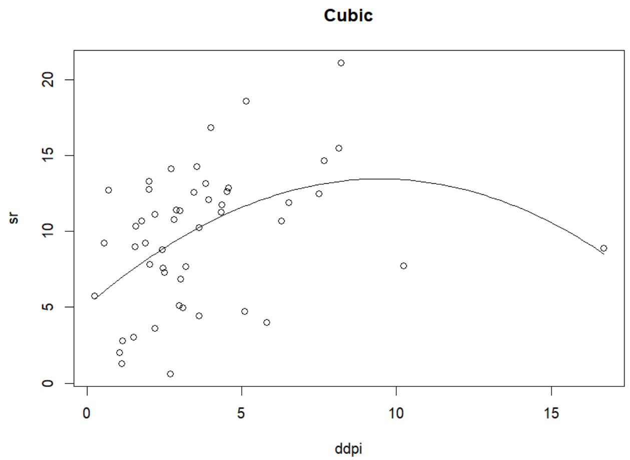
Polynomial regression

```
> g1<-lm(sr ~ ddpi,savings)
>  plot(sr ~ ddpi, savings); abline(g1); title("Linear"); summary(g1)$coef
             Estimate Std. Error  t value     Pr(>|t|)
(Intercept) 7.883021  1.0110011 7.797243 4.464697e-10
ddpi        0.475830  0.2146166 2.217117 3.138509e-02
>  g2<-lm(sr ~ ddpi + I(ddpi^2),savings)
>  summary(g2)$coef
                Estimate Std. Error   t value     Pr(>|t|)
(Intercept)  5.13038069 1.43471517  3.575888 0.0008211413
ddpi         1.75751897 0.53772368  3.268443 0.0020258542
I(ddpi^2)   -0.09298521 0.03612318 -2.574115 0.0132617330
>  x <- seq(min(savings$ddpi), max(savings$ddpi), length.out=100)
>  predy2 <- coef(g2)[1]+coef(g2)[2]*x + coef(g2)[3]*x^2
>  plot(sr ~ ddpi, savings); lines(x, predy2); title("Quadratic")
```

```
>  g3<-lm(sr ~ ddpi + I(ddpi^2) + I(ddpi^3),savings)
>  summary(g3)$coef
                 Estimate Std. Error      t value   Pr(>|t|)
(Intercept)  5.145360e+00 2.19860644  2.340282237 0.02366212
ddpi         1.746017e+00 1.38045499  1.264812459 0.21230898
I(ddpi^2)   -9.096724e-02 0.22559835 -0.403226554 0.68864973
I(ddpi^3)   -8.496955e-05 0.00937393 -0.009064453 0.99280691
>  predy3 <- coef(g3)[1]+coef(g3)[2]*x + coef(g3)[3]*x^2 + coef(g3)[4]*x^3
>  plot(sr ~ ddpi, savings); lines(x, predy3); title("Cubic")
```







Orthogonal polynomials

- When a term is removed from or added to the model, coefficients change and model needs to be refitted.
- High order polynomial models may be numerically unstable.
- Orthogonal polynomials may help:
  - replace old set of predictors X, X^2, X^3 ... by new, orthogonal, set of predictors Z_1, Z_2, Z_3 ...

$$Z_1 = a_1 + b_1 X$$

$$Z_2 = a_2 + b_2 X + c_2 X^2$$

$$Z_3 = a_3 + b_3 X + c_3 X^2 + d_3 X^3$$

Such that $Z_i' Z_j = 0$ and $Z_i' Z_i = 1$

Orthogonal polynomials: Example

```
>  g2 <- lm(sr ~ poly(ddpi,2),savings)
>   summary(g2)$coef
                 Estimate Std. Error   t value     Pr(>|t|)
(Intercept)      9.671000  0.5768611 16.764868 1.841030e-21
poly(ddpi, 2)1   9.558993  4.0790239  2.343451 2.338794e-02
poly(ddpi, 2)2 -10.499876  4.0790239 -2.574115 1.326173e-02
>   g4<- lm(sr ~ poly(ddpi,4),savings)
>   summary(g4)$coef
                   Estimate Std. Error      t value     Pr(>|t|)
(Intercept)      9.67100000   0.584602 16.542879686 9.477039e-21
poly(ddpi, 4)1   9.55899338   4.133760  2.312420904 2.538538e-02
poly(ddpi, 4)2 -10.49987612   4.133760 -2.540030321 1.460646e-02
poly(ddpi, 4)3  -0.03737382   4.133760 -0.009041119 9.928263e-01
poly(ddpi, 4)4   3.61196847   4.133760  0.873773113 3.868811e-01
```

The `poly()` function constructs Orthogonal polynomials

We come to the same coefficients

```
>   x <- model.matrix(g4)
>   dimnames(x) <- list(NULL,c("Int","power1","power2","power3","power4"))
>   round(t(x) %*% x,3)
       Int power1 power2 power3 power4
Int     50      0      0      0      0
power1   0      1      0      0      0
power2   0      0      1      0      0
power3   0      0      0      1      0
power4   0      0      0      0      1
```

Orthogonal polynomials Z_is are indeed orthogonal.

Regression splines

- A spline is a piecewise polynomial with a certain level of smoothness. Spline fixes the disadvantages of Polynomial regression by combining it with Segmented regression (see more on practical session).
- Splines use B-spline basis functions.
- Define cubic B-spline basis $S(X)$ (defined over $[a;\ b]$) using knots at $t_1, \ldots, t_k$
  - $S(X),\ S'(X), S''(X)$ are continuous on $[a;\ b]$
  - Partition $a = t_0 < t_1 < \cdots < t_k = b$, function $S(\mathrm{X})$ is cubic on each subinterval $[t_i, t_{i+1}]$, i.e.,

$$S_i(X) = a_{0,i} + a_{1,i}X + a_{2,i}X^2 + a_{3,i}X^3$$

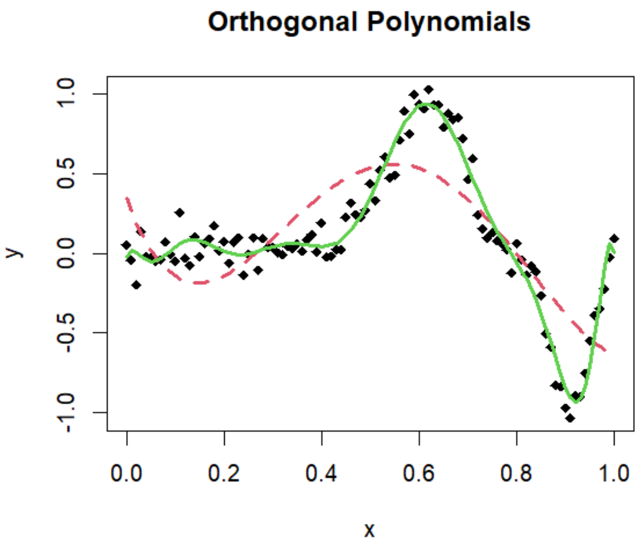
  - How many unknowns are there?

Regression splines: Example

- Suppose we know the true model is:

$$y = \sin^3(2\pi x^3) + \epsilon, \text{ with } \epsilon \sim N(0; 0.1^2)$$

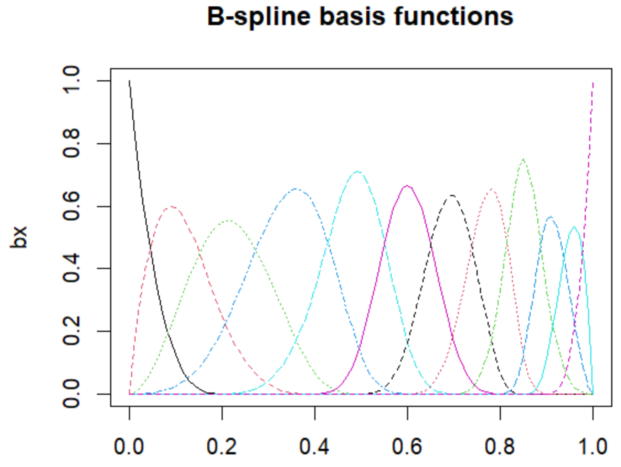
```
>   funky <- function(x) sin(2*pi*x^3)^3
>   x <- seq(0,1,by=0.01)
>   y <- funky(x) + 0.1*rnorm(101)
>   matplot(x,cbind(y,funky(x)),type="pl",ylab="y",
+             pch=18,lty=1,main="True Model")

>   g4 <- lm(y ~ poly(x,4))
>   g12 <- lm(y ~ poly(x,12))
>   matplot(x,cbind(y,g4$fit,g12$fit),type="pll",ylab="y",
+     lwd=3, pch=18,lty=c(1,2),main="Orthogonal Polynomials")
```

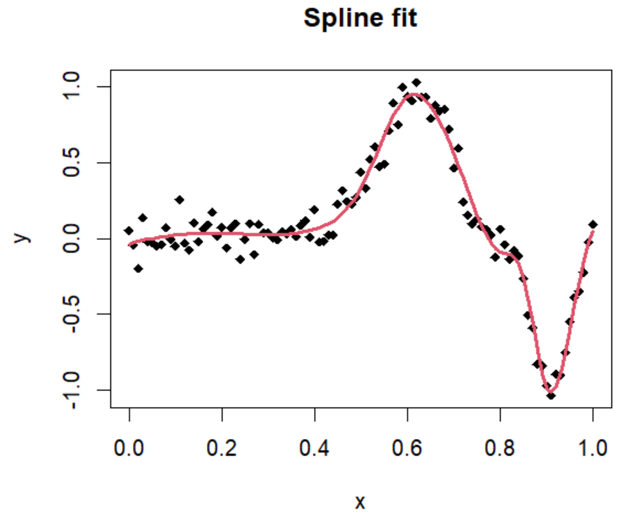


Regression splines: Example

- Now, let’s use splines with 12 basis functions.

```
>   library(splines)
>   knots <- c(0,0,0,0,0.2,0.4,0.5,0.6,0.7,0.8,0.85,0.9,1,1,1,1)
>   # a cubic spline has order 4 (4 coefficients in each segment)
>   # 16 - 4 = 12 basis functions
>   bx <- splineDesign(knots,x)
>   gs <- lm(y ~ bx)
>   matplot(x,bx,type="l",main="B-spline basis functions")
>   matplot(x,cbind(y,gs$fit),type="pl",ylab="y",pch=18,lty=1,
+           lwd=3,  main="Spline fit")
```





- **Hint**: Place more knots in places where the function might vary rapidly and fewer knows where it seems more stable.