1 Transformation

Variance-stabilizing transformation

Transform y to $h(y)$ to stabilize the variance. Applying Taylor expansion: let $\mu = E(y)$

$$h(y) \approx h(\mu) + h'(\mu)(y - \mu)$$

$\Rightarrow \mathbf{E}[h(y)] = h(\mu) \Rightarrow \text{Var}[h(y)] = \text{Var}\{h(y) - \mathbf{E}[h(y)]\}^2 \approx \text{Var}(y)[h'(\mu)]^2 \Rightarrow h'(\mu) \propto [\text{Var}(y)]^{-1/2} \Rightarrow h(y) \approx c \int [\text{Var}(y)]^{-1/2} dy$ for some constant c

- if $\text{Var}(y) \propto y^2$, $\Rightarrow h(y) = \log(y)$ for $y_i > 0$
- if $\text{Var}(y) \propto y$, $\Rightarrow h(y) = \sqrt{y}$ for $y_i \geq 0$
- if y is a Poisson RV in a SLR model: mean of y is related to the regressor variable x and the variance of y will be proportional to x, then we could regression $y' = \sqrt{y}$ against x.
- if y is a proportion and $\text{Var}(y) \propto \mathbf{E}(y)[1 - \mathbf{E}(y)]$, then the arcsin transformation $y = sin^{-1}(\sqrt{y})$ is appropriate.

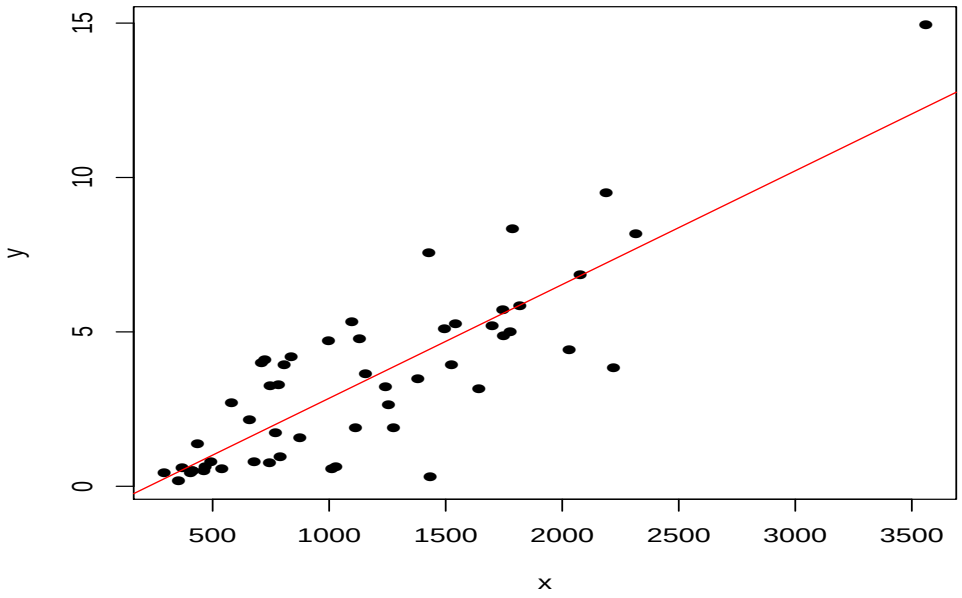
Nonconstant error variance: the LS estimator is still unbiased but is not minimum-variance.

Engergy Demand during the peak hour to the total enerage useage during the month.

```
> dat1=read.table("ex-5-1.csv", header=T, sep=",")
> str(dat1)

'data.frame':        53 obs. of  3 variables:
 $ Customer: int  1 2 3 4 5 6 7 8 9 10 ...
 $ x..kWh. : int  679 292 1012 493 582 1156 997 2189 1097 2078 ...
 $ y..kW.  : num  0.79 0.44 0.56 0.79 2.7 3.64 4.73 9.5 5.34 6.85 ...

> x=dat1[,2]
> y=dat1[,3]
> plot(x,y, pch=16)
> f=lm(y~x)
> (b=coef(f))

 (Intercept)           x
-0.831303660  0.003682843

> summary(f)
```

```
Call:
lm(formula = y ~ x)

Residuals:
    Min      1Q  Median      3Q     Max
-4.1399 -0.8275 -0.1934  1.2376  3.1522

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.8313037  0.4416121  -1.882   0.0655
x            0.0036828  0.0003339  11.030 4.11e-15

Residual standard error: 1.577 on 51 degrees of freedom
Multiple R-squared: 0.7046,        Adjusted R-squared: 0.6988
F-statistic: 121.7 on 1 and 51 DF,  p-value: 4.106e-15
```

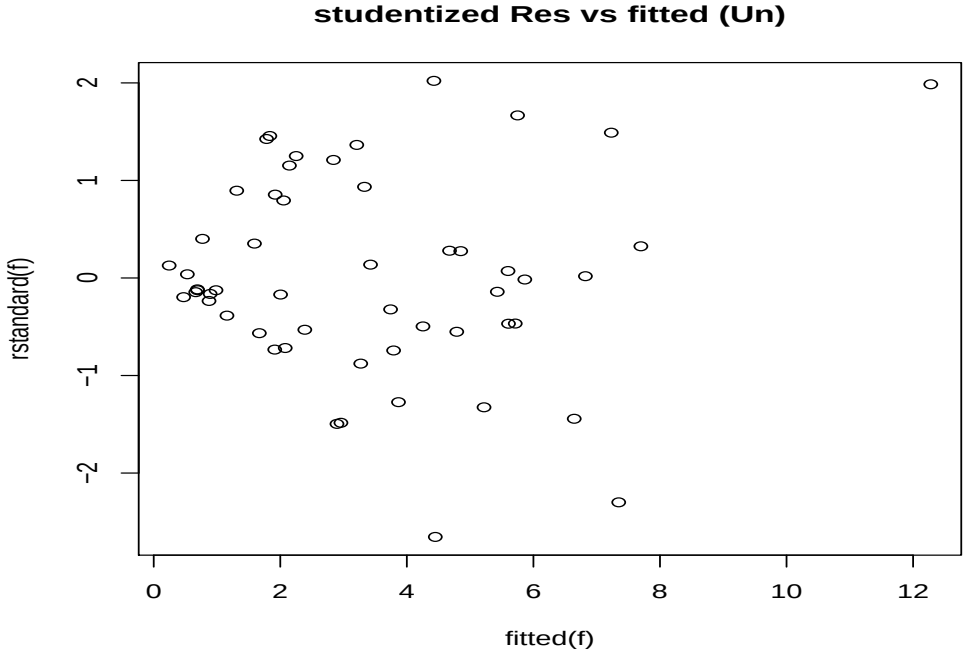
```
> abline(coef(f), col=2)
```




```
> library(xtable)
> xtable(anova(f))
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
x	1	302.63	302.63	121.66	0.0000
Residuals	51	126.87	2.49		

```
> plot(fitted(f), rstandard(f), main="studentized Res vs fitted (Un)")
```



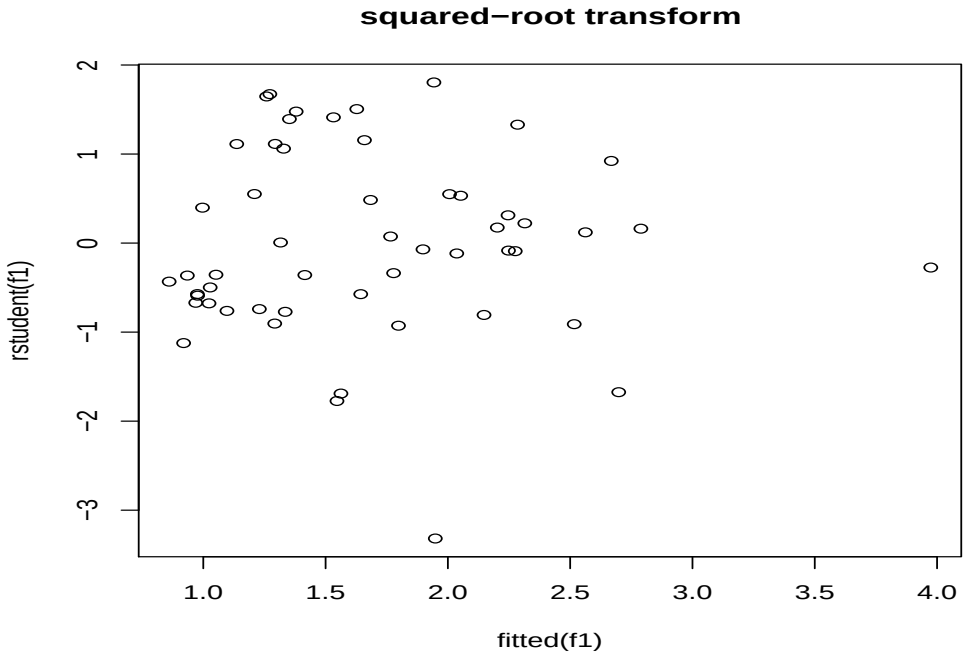
■fig=T■ plot(fitted(f), rstudent(f), main="external studentized Res vs fitted") Error variance is increasing as the energy consumption increases.

Try a squared-root transformation: regression $y' = sqrty$ on x as a variance-stabilizing transformation.

```
> f1=lm(sqrt(y)~x)
> summary(f1)

Call:
lm(formula = sqrt(y) ~ x)

Residuals:
     Min       1Q   Median       3Q      Max
-1.39185 -0.30576 -0.03875  0.25378  0.81027

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 5.822e-01  1.299e-01   4.481 4.22e-05
x           9.529e-04  9.824e-05   9.699 3.61e-13

Residual standard error: 0.464 on 51 degrees of freedom
Multiple R-squared: 0.6485,        Adjusted R-squared: 0.6416
F-statistic: 94.08 on 1 and 51 DF,  p-value: 3.614e-13

> (b.sy=coef(f1))
```

```
(Intercept)           x
0.582225917 0.000952859
```

```
> plot(fitted(f1), rstudent(f1), main="squared-root transform")
```

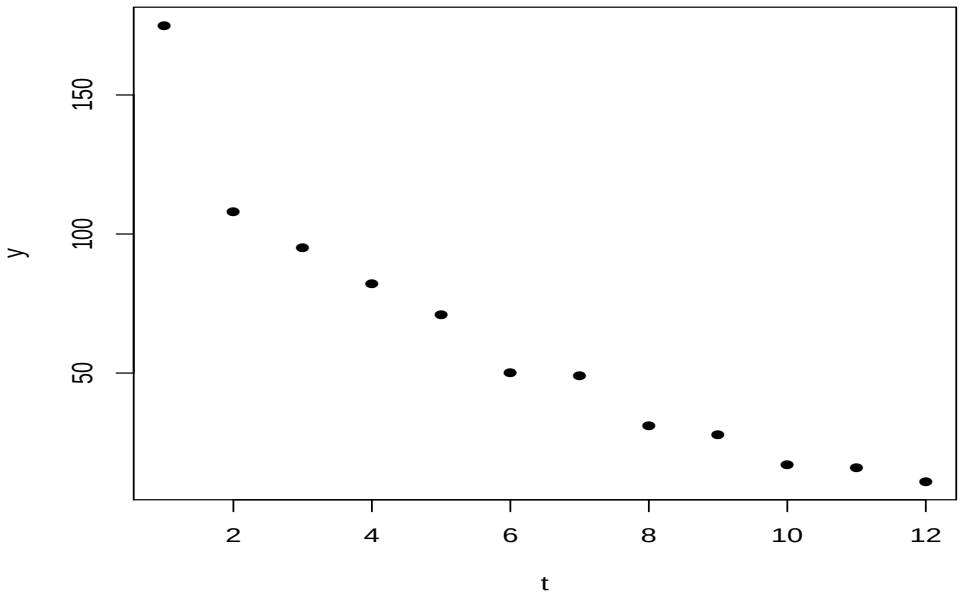


We might want to check data 26 and 50.
Another example

```
> #bacteria <- read.table("bacteria.txt", header=T)
>
> bacteria=structure(list(y = c(175L, 108L, 95L, 82L, 71L, 50L, 49L, 31L,
+ 28L, 17L, 16L, 11L), t = 1:12), .Names = c("y", "t"), class = "data.frame", row.name
+ -12L))
> str(bacteria)

'data.frame':        12 obs. of  2 variables:
 $ y: int  175 108 95 82 71 50 49 31 28 17 ...
 $ t: int  1 2 3 4 5 6 7 8 9 10 ...

> y=bacteria$y
> t=bacteria$t
```

bacteria vs. time – exponential decay?

```
> plot(y~t, pch=16)
```



Fit model with untransformed data

```
> g <- lm(y ~ t)  # untransformed model
> summary(g)

Call:
lm(formula = y ~ t)

Residuals:
    Min      1Q  Median      3Q     Max
-17.323  -9.890  -7.323   2.463  45.282

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   142.20      11.26  12.627 1.81e-07
t             -12.48       1.53  -8.155 9.94e-06

Residual standard error: 18.3 on 10 degrees of freedom
Multiple R-squared: 0.8693,        Adjusted R-squared: 0.8562
F-statistic: 66.51 on 1 and 10 DF,  p-value: 9.944e-06
```

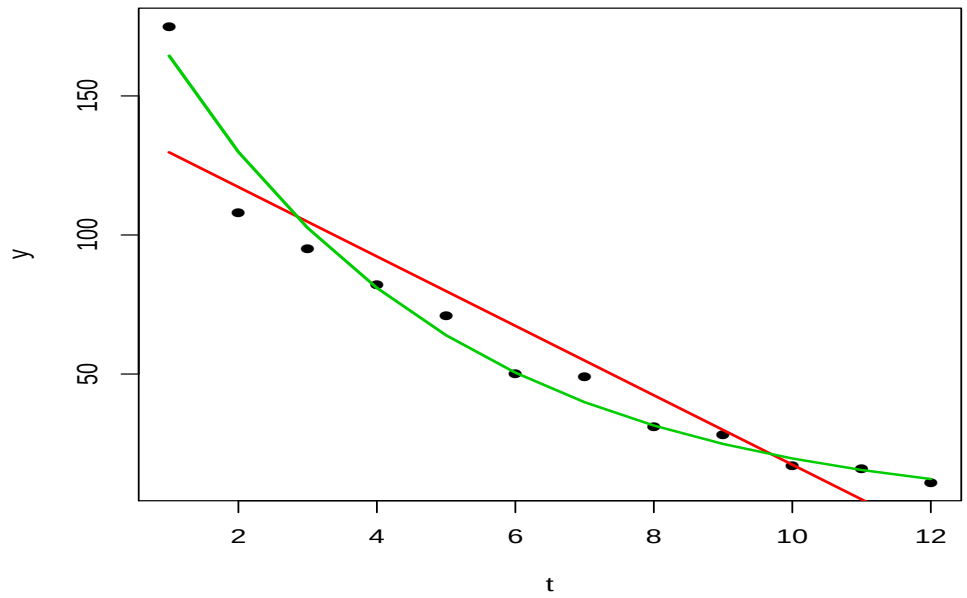
Fit model with log-transformed data

```
> # transformed model
> g.log <- lm(log(y) ~ t)
> summary(g.log)
```

```
Call:
lm(formula = log(y) ~ t)

Residuals:
      Min        1Q    Median        3Q       Max
-0.184303 -0.083994  0.001453  0.072825  0.206246

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  5.33878    0.07409   72.05 6.47e-15
t           -0.23617    0.01007  -23.46 4.49e-10

Residual standard error: 0.1204 on 10 degrees of freedom
Multiple R-squared: 0.9822,        Adjusted R-squared: 0.9804
F-statistic: 550.3 on 1 and 10 DF,  p-value: 4.489e-10
```

bacteria vs. time – fitted values

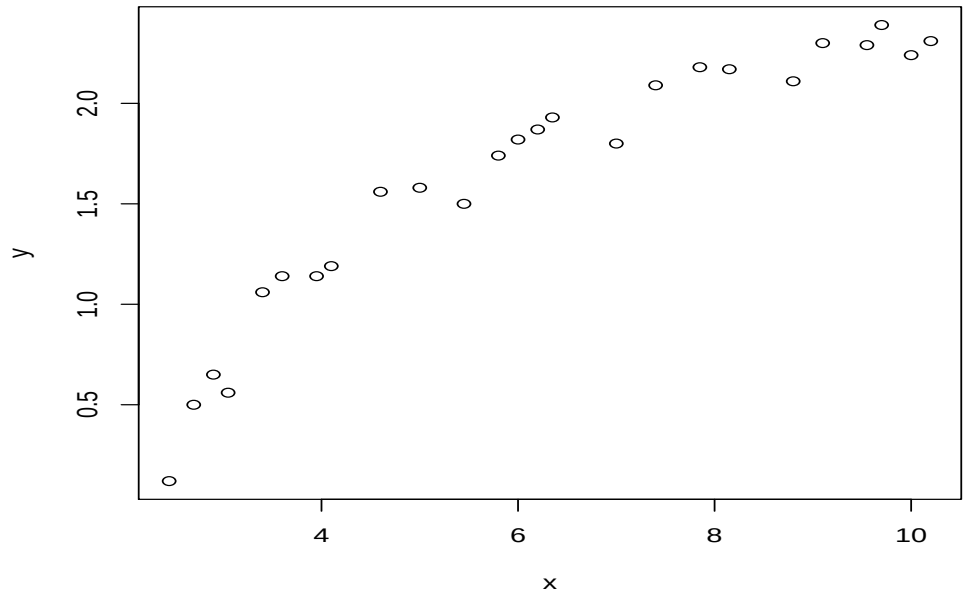
```
> par(mfrow=c(1,1))
> plot(y~t, pch=16)
> lines(t, fitted(g), lwd=2, col=2)
> lines(t, exp(fitted(g.log)), lwd=2, col=3)
```



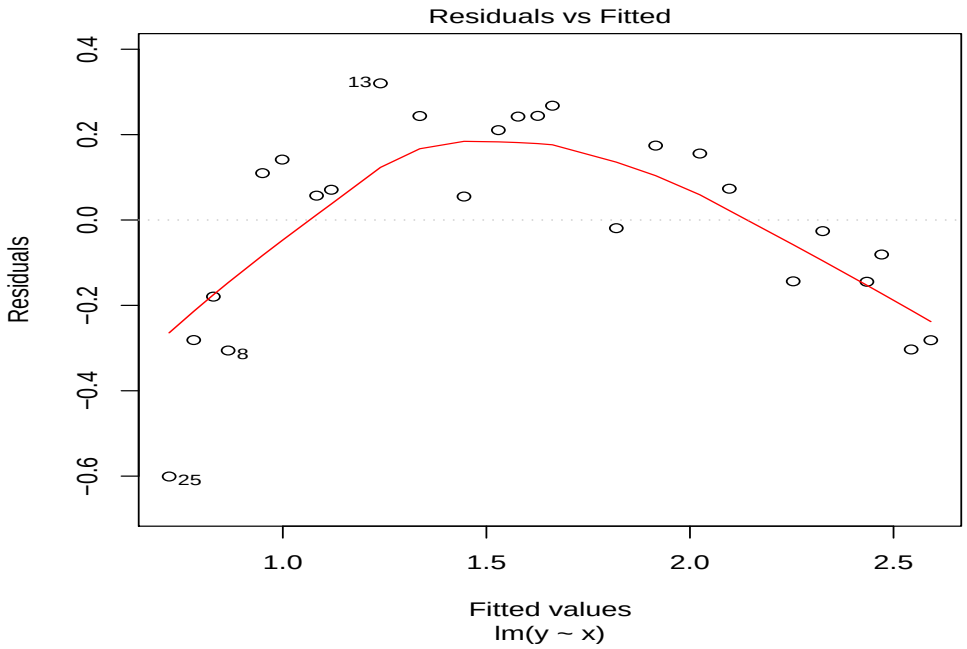
Intrinsically or transformably linear

```
> windmill=read.table("ex-5-2.csv", header=T, sep=",")
> str(windmill)
```

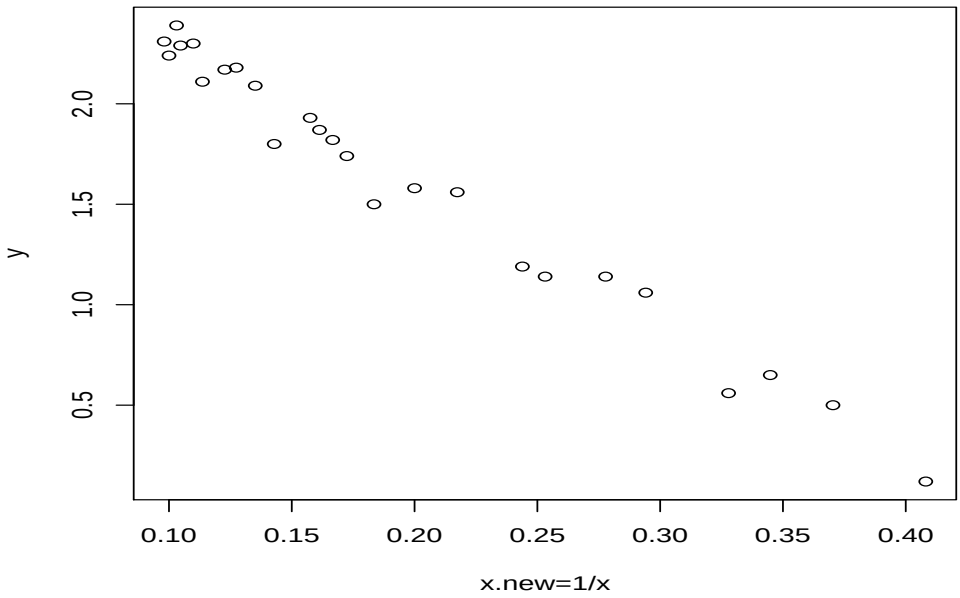
```
'data.frame':        25 obs. of  3 variables:
 $ Observation.Number..i  : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Wind.Velocity..xi..mph.: num  5 6 3.4 2.7 10 9.7 9.55 3.05 8.15 6.2 ...
 $ DC.Output..yi          : num  1.58 1.82 1.06 0.5 2.24 2.39 2.29 0.56 2.17 1.87 ...

> x=windmill[,2]
> y=windmill[,3]
> plot(y~x)
> g1=lm(y~x)
> summary(g1)

Call:
lm(formula = y ~ x)

Residuals:
     Min       1Q   Median       3Q      Max
-0.60084 -0.14463  0.05709  0.17433  0.32019

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.12946    0.12619   1.026    0.316
x            0.24138    0.01908  12.651 7.64e-12

Residual standard error: 0.2364 on 23 degrees of freedom
Multiple R-squared: 0.8744,        Adjusted R-squared: 0.8689
F-statistic: 160.1 on 1 and 23 DF,  p-value: 7.641e-12
```



```
> plot(g1, which=1)
```




```
> x.new=1/x
> plot(y~x.new, xlab="x.new=1/x")
> g2=lm(y~x.new)
> summary(g2)

Call:
lm(formula = y ~ x.new)

Residuals:
     Min       1Q   Median       3Q      Max
-0.20656 -0.04603  0.01036  0.08261  0.12543

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.98013    0.04518   65.96   <2e-16
x.new       -6.94099    0.20770  -33.42   <2e-16

Residual standard error: 0.09475 on 23 degrees of freedom
Multiple R-squared: 0.9798,        Adjusted R-squared: 0.9789
F-statistic:  1117 on 1 and 23 DF,  p-value: < 2.2e-16
```



Analytical methods for selecting a ransformation

transformation on y

Box-Cox transformation: choose λ using maximum likelihood which the residual sum of squares from the fitted model $\mathrm{SS}_{\mathrm{Res}}(\lambda)$ is a minimum.

Transforming the response can make the model harder to interpret so we don't want to do it unless it's really necessary. One way to check is to form a confidence interval for λ. A $100(1-\alpha)\%$ confidence interval for λ is

$$\{\lambda : L(\lambda) > L(\hat{\lambda} - \frac{1}{2}\xi^2_{1-\alpha})\}$$

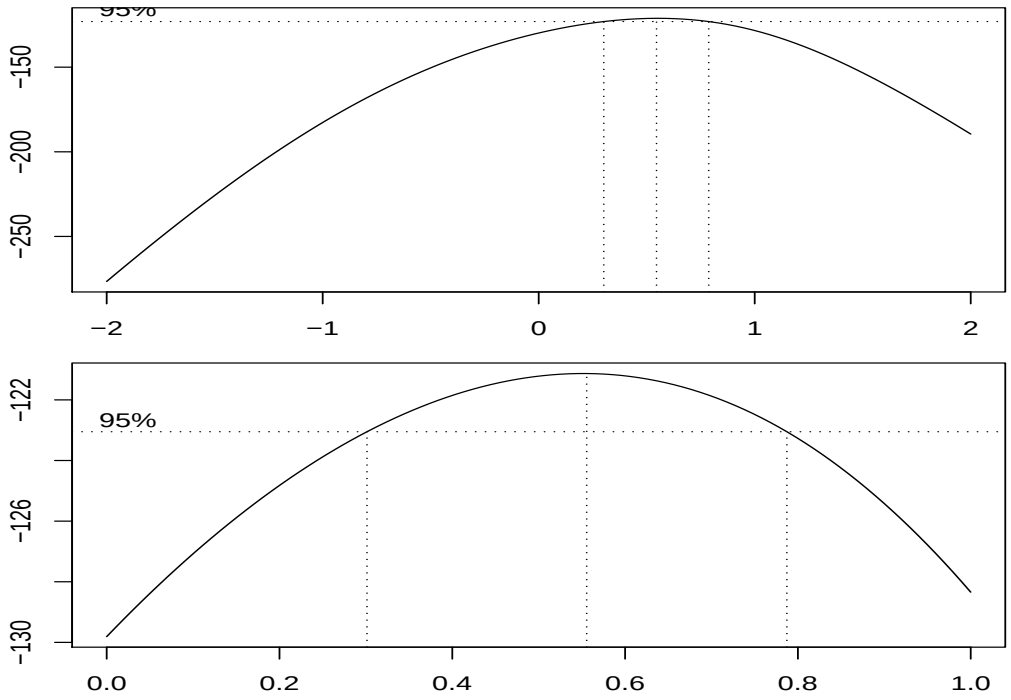
This is derived by inverting the likelihood ratio test of the hypothesis that $H_0 : \lambda = \lambda_0$ which uses the statistics

$$-2\log\frac{L(f_0)}{L(f_1)} = 2[L(\hat{\lambda}) - L(\lambda_0)] \rightarrow \Xi^2(1)$$

example: Use the Box and Cox procedure to select a variance-stabilizing trasformation for the electric utility data. Recall $f = \mathrm{lm}(y \sim x)$

"MASS" library is needed for the boxcox procedure

```
> library(MASS)
> y=dat1[,3]
> x=dat1[,2]
> g1=lm(y~x)
> par(mfrow=c(2,1), mar=c(2,2,1,1))
> boxcox(g1, plotit=T)
> boxcox(g1, plotit=T, lambda=seq(0,1,by=0.1))
```



the second one gives a finer "grid" on λ in (0,1).

transformations on the regressor variables

Box and Tidwell (1962) Piecewise regression (Ch7) Spline regression

2 Generalized and Weighted Least Squares

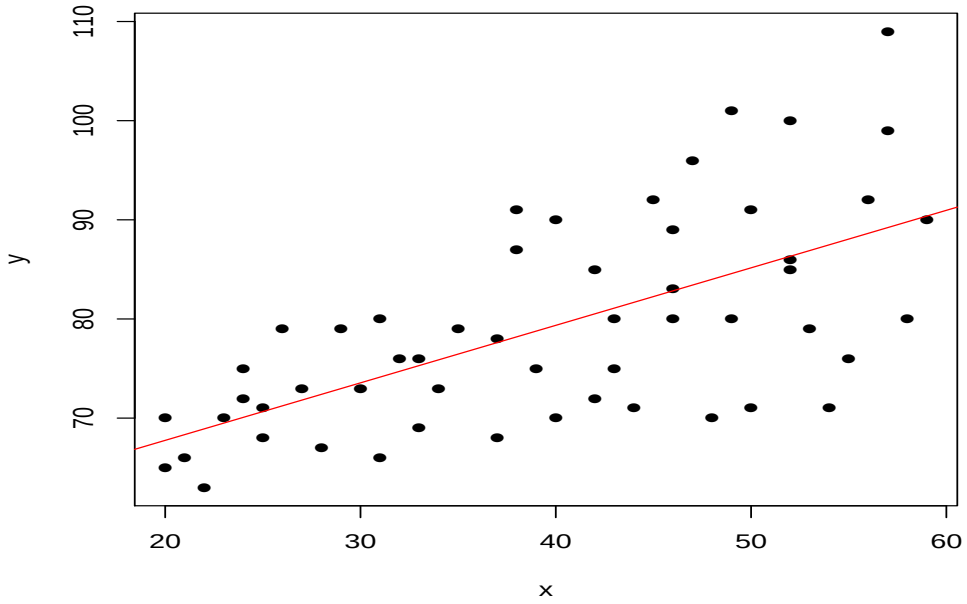
1. WLS: the deviation betweem the observed and expected values of y_i is multiplied by a **weight** w_i **chosen inversely proportional to the variance of** y_i.

$$S(\beta_0, \beta_1) = \sum_{i=1}^{n} w_i (y_i - \beta_0 - \beta_1 x_i)^2$$

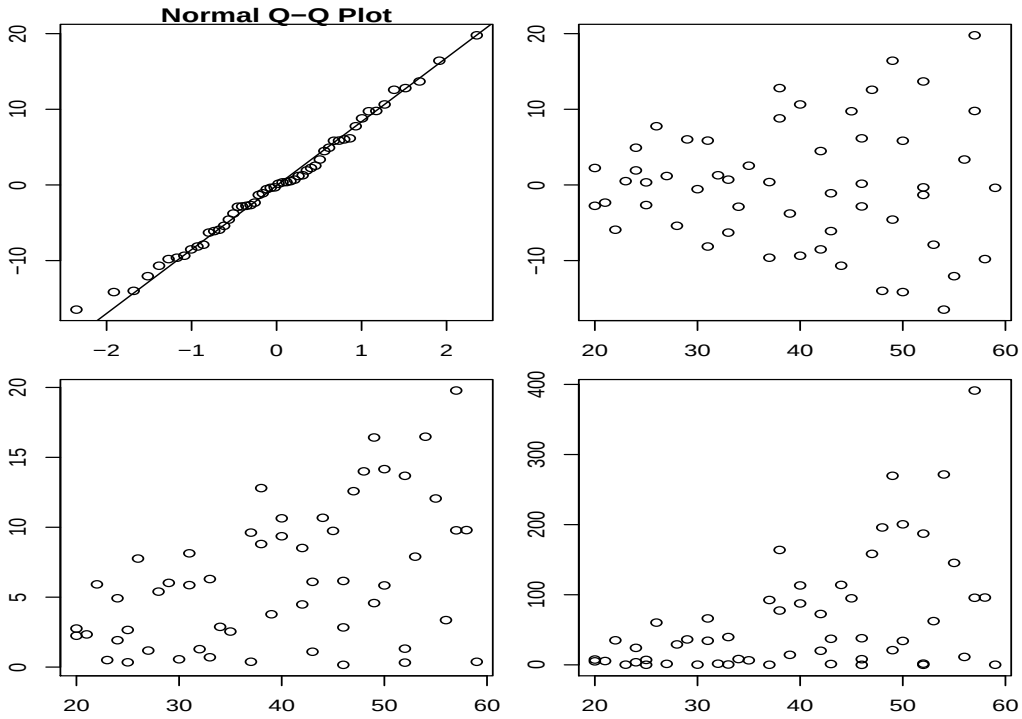
nonconstant error variance but the form is known $\varepsilon \sim N(0, \sigma^2 \mathbf{I}) \rightarrow \varepsilon \sim N(0, \sigma^2 \mathbf{V}$ where σ^2 is unknown but $\mathbf{V}$ is known.

```
> blood=read.table("blood.txt", header=T)
> str(blood)

'data.frame':        54 obs. of  2 variables:
 $ x: int  27 21 22 24 25 23 20 20 29 24 ...
 $ y: int  73 66 63 75 71 70 65 70 79 72 ...

> y=blood$y
> x=blood$x
> fm1=lm(y~x)
> summary(fm1)
```

```
Call:
lm(formula = y ~ x)

Residuals:
      Min        1Q    Median        3Q       Max
-16.47859  -5.78765  -0.07844   5.61173  19.78132

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 56.15693    3.99367  14.061  < 2e-16
x            0.58003    0.09695   5.983 2.05e-07

Residual standard error: 8.146 on 52 degrees of freedom
Multiple R-squared: 0.4077,        Adjusted R-squared: 0.3963
F-statistic: 35.79 on 1 and 52 DF,  p-value: 2.050e-07

> anova(fm1)

Analysis of Variance Table

Response: y
          Df Sum Sq Mean Sq F value    Pr(>F)
x          1 2375.0  2375.0  35.793 2.050e-07
Residuals 52 3450.4    66.4

> par(mfrow=c(1,1))
> plot(y~x, pch=16)
> abline(coef(fm1), col=2)
```




```
> res=residuals(fm1)
> par(mfrow=c(2,2), mar=c(2,2,1,1))
> qqnorm(res);qqline(res)
> plot(x, res )
> plot(x, abs(res))
> plot(x, res^2)
```



```
> library(xtable)
> fm2=lm(abs(res)~x)
> summary(fm2)

Call:
lm(formula = abs(res) ~ x)

Residuals:
    Min      1Q  Median      3Q     Max
-9.7639 -2.7882 -0.1587  3.0757 10.0350

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.54948    2.18692  -0.709 0.481786
x            0.19817    0.05309   3.733 0.000470

Residual standard error: 4.461 on 52 degrees of freedom
Multiple R-squared: 0.2113,        Adjusted R-squared: 0.1962
F-statistic: 13.93 on 1 and 52 DF,  p-value: 0.0004705

> wi=0*y+1/((fitted(fm2))^2)
```

	x	y	res	abs(res)	weights
1	27.00	73.00	1.18	1.18	0.07
2	21.00	66.00	-2.34	2.34	0.15
3	22.00	63.00	-5.92	5.92	0.13
4	24.00	75.00	4.92	4.92	0.10
5	25.00	71.00	0.34	0.34	0.09
6	23.00	70.00	0.50	0.50	0.11

```
> fm3=lm(y~x, weights=wi)
> xtable(summary(fm3))
```

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	55.5658	2.5209	22.04	0.0000
x	0.5963	0.0792	7.53	0.0000

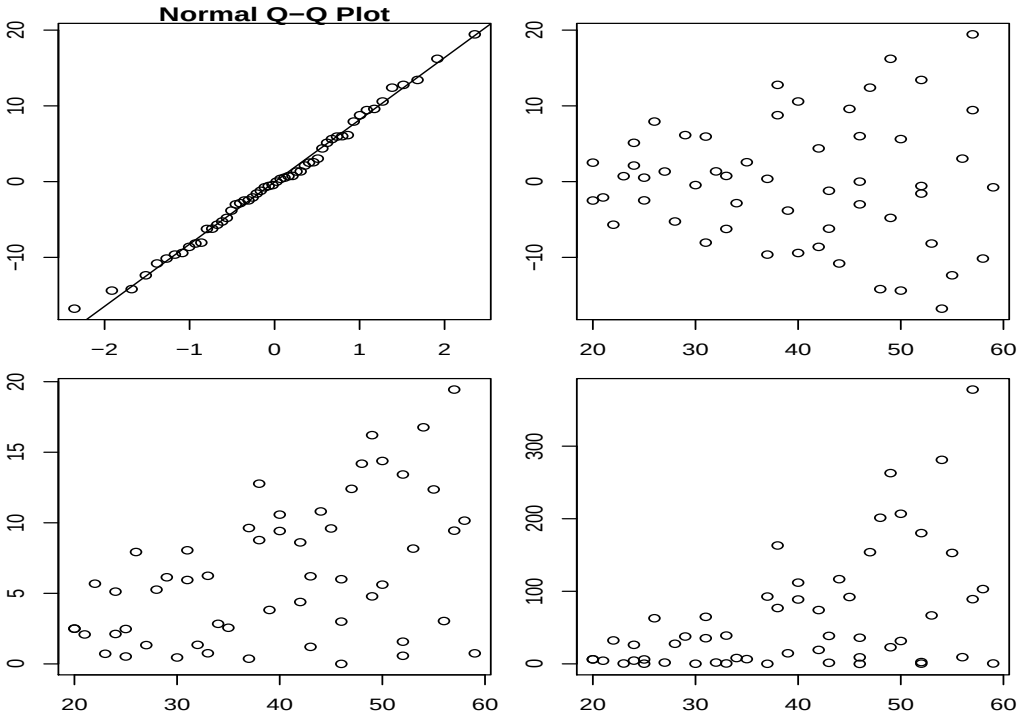
```
> confint(fm3, "x")
```

2.5 x 0.437339 0.7553445

```
> cov.b=vcov(fm3)
> ( c(sqrt(cov.b[1,1]),sqrt(cov.b[2,2])) )
```

[1] 2.52091762 0.07923803

```
> res=residuals(fm3)
> par(mfrow=c(2,2), mar=c(2,2,1,1))
> qqnorm(res);qqline(res)
> plot(x, res )
> plot(x, abs(res))
> plot(x, res^2)
```



2. Generalized Least square estimator.