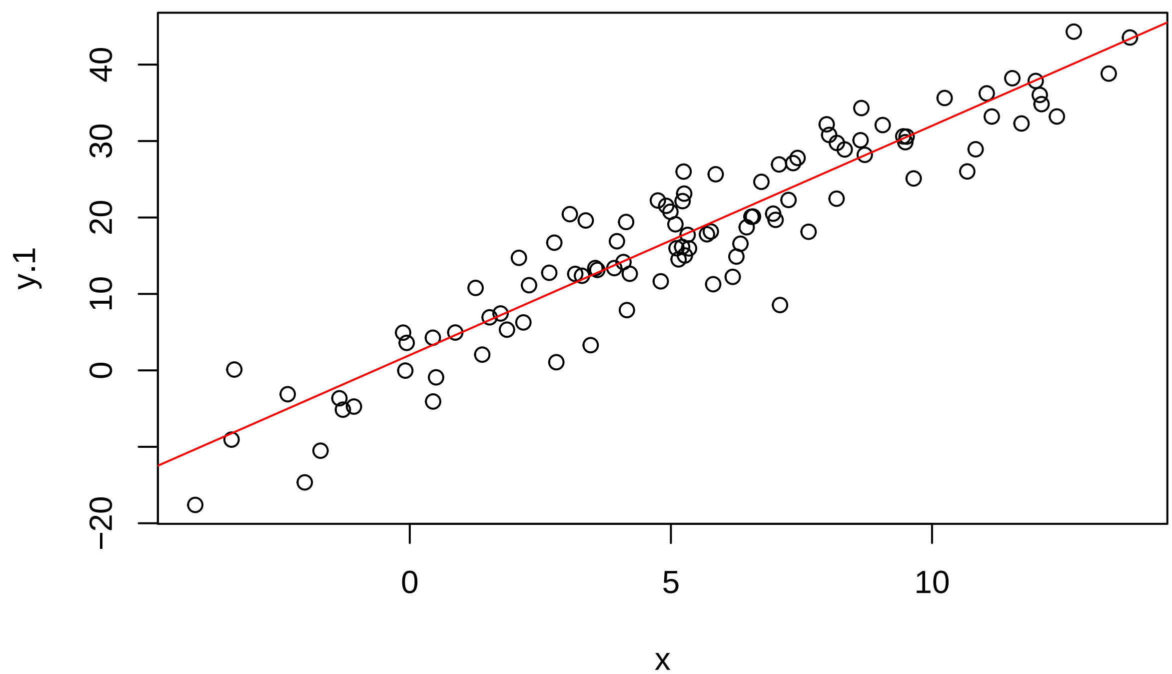
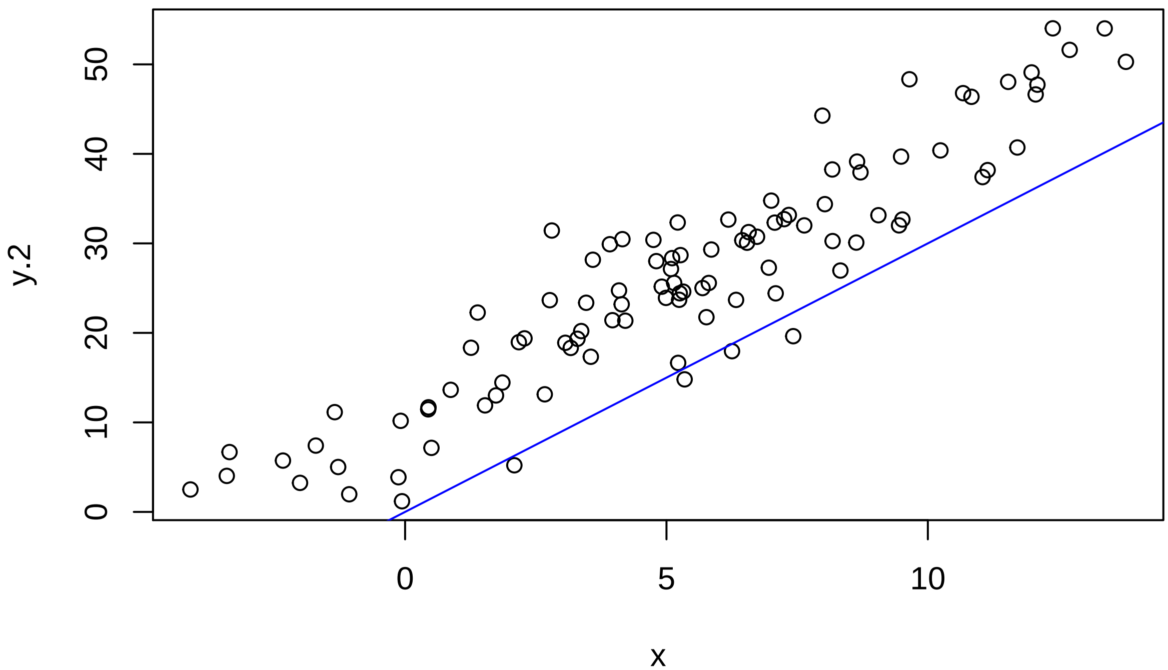
Comp stats notebook

Using simple linear regression to prove the properties of LS estimator

```r
## generating fake dataset
set.seed(21)
x <- rnorm(n=100, mean=5, sd=4)
# the regression lines in the following two models should
# be the same since the error mean is absorbed in the model intercept
eps.1 <- rnorm(n=100, mean=0, sd=1)
eps.2 <- rnorm(n=100, mean=2, sd=1)
y.1 <- 2 + 3*x + 5*eps.1
y.2 <- 3*x + 5*eps.2
plot(x,y.1)
abline(a=2,b=3, col="red")
```




```r
plot(x,y.2)
abline(a=0, b=3, col="blue")
```



Now let's build the regression model.

```
reg.1 <- lm(y.1~x)
reg.2 <- lm(y.2~x)
```

```
summary(reg.1)
```

```
##
## Call:
## lm(formula = y.1 ~ x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.1965  -2.8125   0.3336   3.4590   9.7342
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.5335     0.7624   2.011    0.047 *
## x             2.9921     0.1140  26.238   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.662 on 98 degrees of freedom
## Multiple R-squared:  0.8754, Adjusted R-squared:  0.8741
## F-statistic: 688.5 on 1 and 98 DF,  p-value: < 2.2e-16
```

```r
summary(reg.2)
```

```
##
## Call:
## lm(formula = y.2 ~ x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -12.6418  -2.7347   0.3618   3.4257  12.6856
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  10.5327     0.8038   13.10   <2e-16 ***
## x             2.9281     0.1202   24.35   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.915 on 98 degrees of freedom
## Multiple R-squared:  0.8582, Adjusted R-squared:  0.8568
## F-statistic: 593.1 on 1 and 98 DF,  p-value: < 2.2e-16
```

What if we want to compute the coefficients by hand? Let's do it!

```r
X <- matrix(cbind(rep(1,100), x), nrow = 100, ncol = 2)
xtx.inv <- solve(t(X)%*%X)
# coefficients
print("Beta 1")
```

```
## [1] "Beta 1"
```

```r
(beta.1 <- xtx.inv %*% t(X) %*% y.1)
```

```
##          [,1]
## [1,] 1.533473
## [2,] 2.992113
```

```r
print("Beta 2")
```

```
## [1] "Beta 2"
```

```r
(beta.2 <- xtx.inv %*% t(X) %*% y.2)
```

```
##           [,1]
## [1,] 10.532683
## [2,]  2.928132
```

```r
# predictions
y.hat.1 <- X%*%beta.1
y.hat.2 <- X%*%beta.2
# errors
eps.hat.1 <- y.1 - y.hat.1
eps.hat.2 <- y.2 - y.hat.2
# sd estimate
sd.hat.1 <- sum(eps.hat.1**2)/(100-2)
sd.hat.2 <- sum(eps.hat.2**2)/(100-2)
# se for beta
print("Se 1")
```

```
## [1] "Se 1"
```

```
(se.1 <- sqrt(xtx.inv[2,2]*sd.hat.1))
```

```
## [1] 0.1140356
```

```
print("Se 2")
```

```
## [1] "Se 2"
```

```
(se.2 <- sqrt(xtx.inv[2,2]*sd.hat.2))
```

```
## [1] 0.120232
```

```
# p-values of t-test
tv.1 <- beta.1[2]/(se.1)
tv.2 <- beta.2[2]/(se.2)
print("P-value 1")
```

```
## [1] "P-value 1"
```

```
(pv.1 <- pt(tv.1, df = 100-2, lower.tail = FALSE))
```

```
## [1] 2.063205e-46
```

```
print("P-value 2")
```

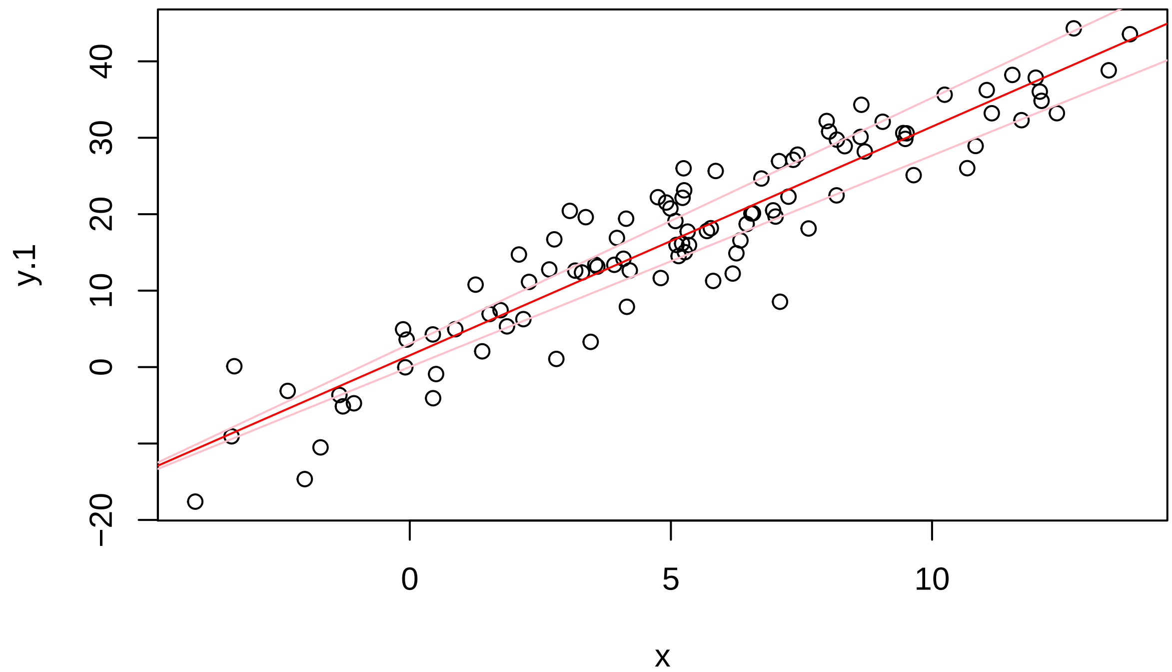
```
## [1] "P-value 2"
```

```
(pv.2 <- pt(tv.2, df = 100-2, lower.tail = FALSE))
```

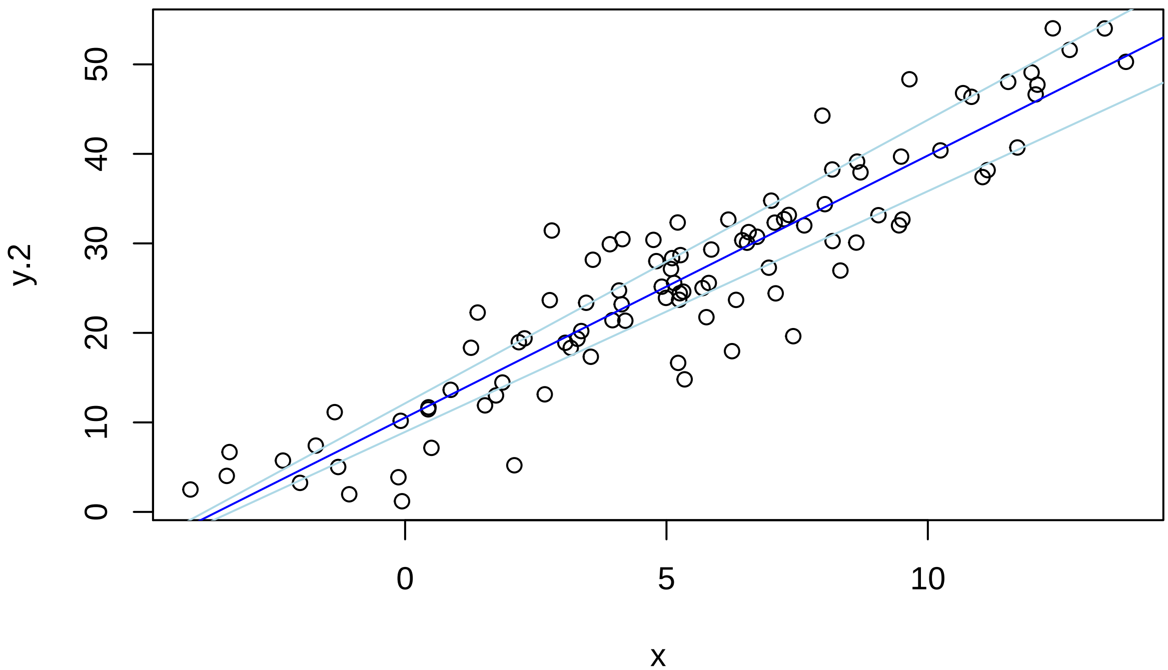
```
## [1] 1.171405e-43
```

Exactly what we have above! What about confidence intervals?

```
ci.1.95 <- qt(0.975, df=100-2)*se.1
ci.2.95 <- qt(0.975, df=100-2)*se.2

# doing the same procedure for the intercept
se.i.1 <- sqrt(xtx.inv[1,1]*sd.hat.1)
se.i.2 <- sqrt(xtx.inv[1,1]*sd.hat.2)
ci.1.i.95 <- qt(0.975, df=100-2)*se.i.1
ci.2.i.95 <- qt(0.975, df=100-2)*se.i.2



plot(x,y.1)
abline(a=beta.1[1],b=beta.1[2], col="red")
abline(a=beta.1[1]+ci.1.i.95,b=beta.1[2]+ci.1.95, col="pink")
abline(a=beta.1[1]-ci.1.i.95,b=beta.1[2]-ci.1.95, col="pink")
```




```
plot(x,y.2)
abline(a=beta.2[1],b=beta.2[2], col="blue")
abline(a=beta.2[1]+ci.2.i.95,b=beta.2[2]+ci.2.95, col="lightblue")
abline(a=beta.2[1]-ci.2.i.95,b=beta.2[2]-ci.2.95, col="lightblue")
```

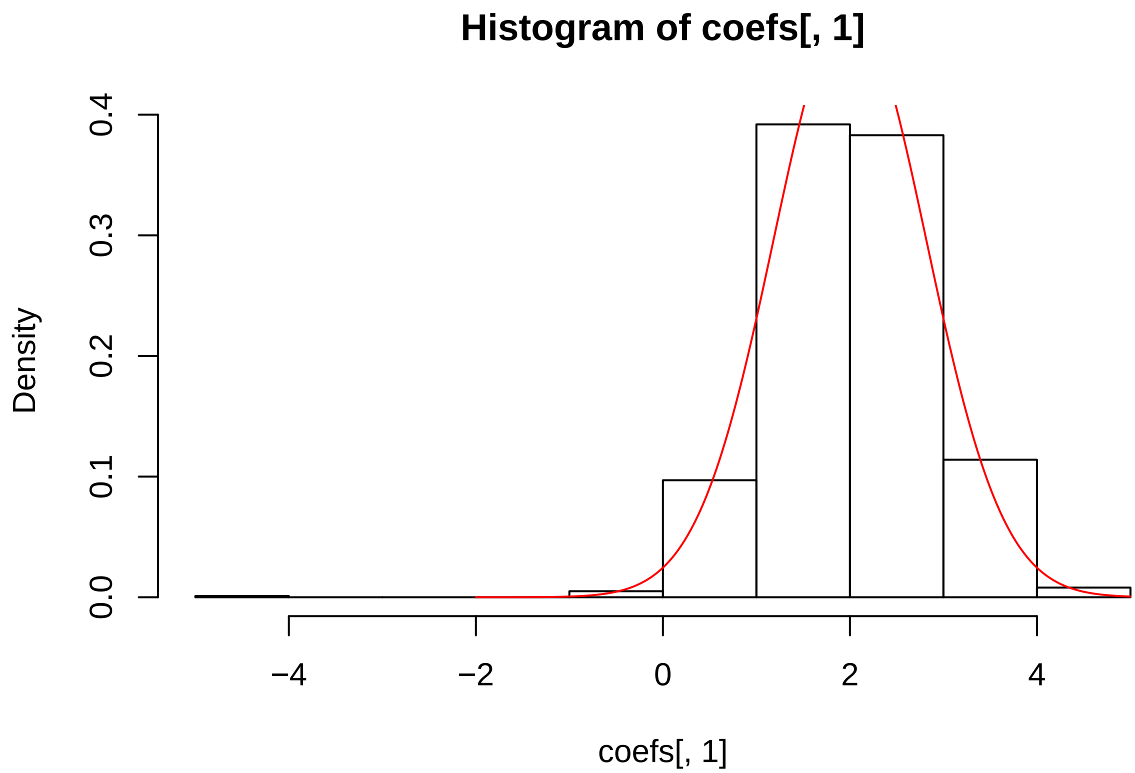


Now let's quickly see how stable this prediction is with a simulation.

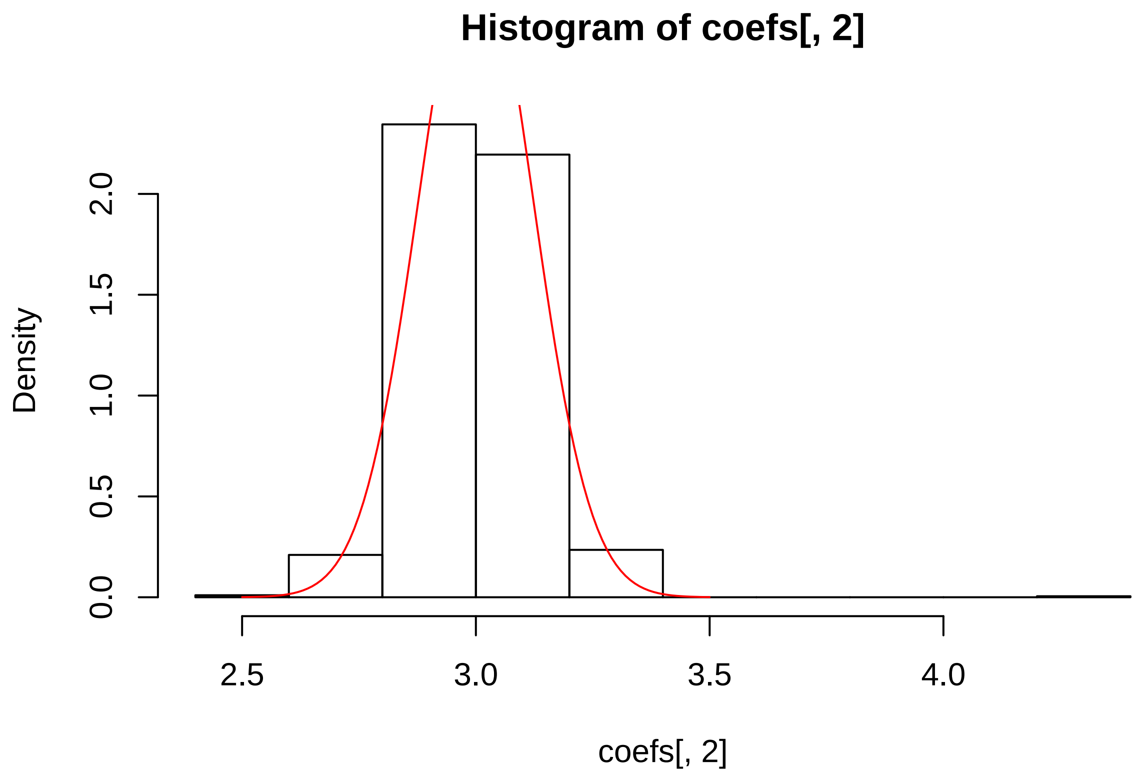
```
## generating fake dataset
set.seed(21)
nsim <- 1000

coefs <- matrix(nrow=nsim, ncol=2)

for(i in 1:nsim){
  eps <- rnorm(n=100, mean=0, sd=1)
  y <- 2 + 3*x + 5*eps
  reg <- lm(y~x)
  beta <- coef(reg)
  names(beta) <- NULL
  coefs[i,] <- beta
}
```

As expected, each coefficient is approximately normally distributed.

```
hist(coefs[,1], freq=FALSE)
lines(seq(-2, 5, by = 0.01), dnorm(seq(-2, 5, by = 0.01), mean= 2, sd = 5*sqrt(xtx.inv[1,1])), col="red
```



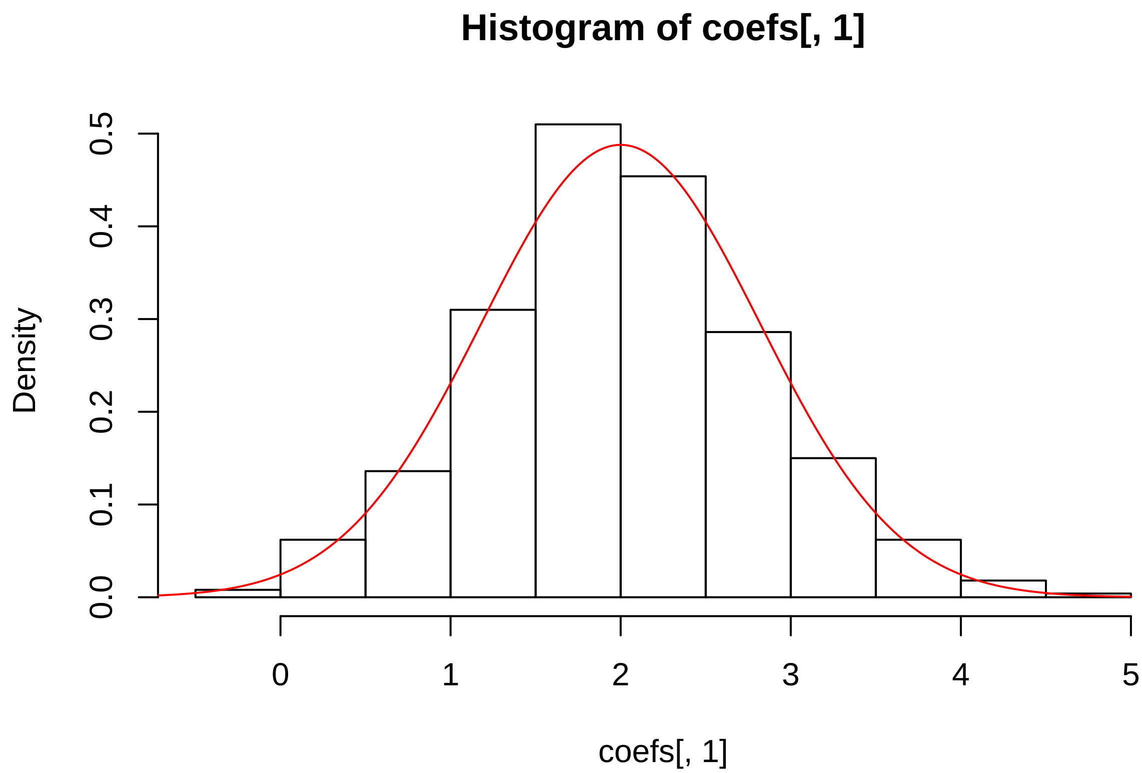

```
hist(coefs[,2], freq = FALSE)
lines(seq(2.5, 3.5, by = 0.01), dnorm(seq(2.5, 3.5, by = 0.01), mean= 3, sd = 5*sqrt(xtx.inv[2,2])),col=
```



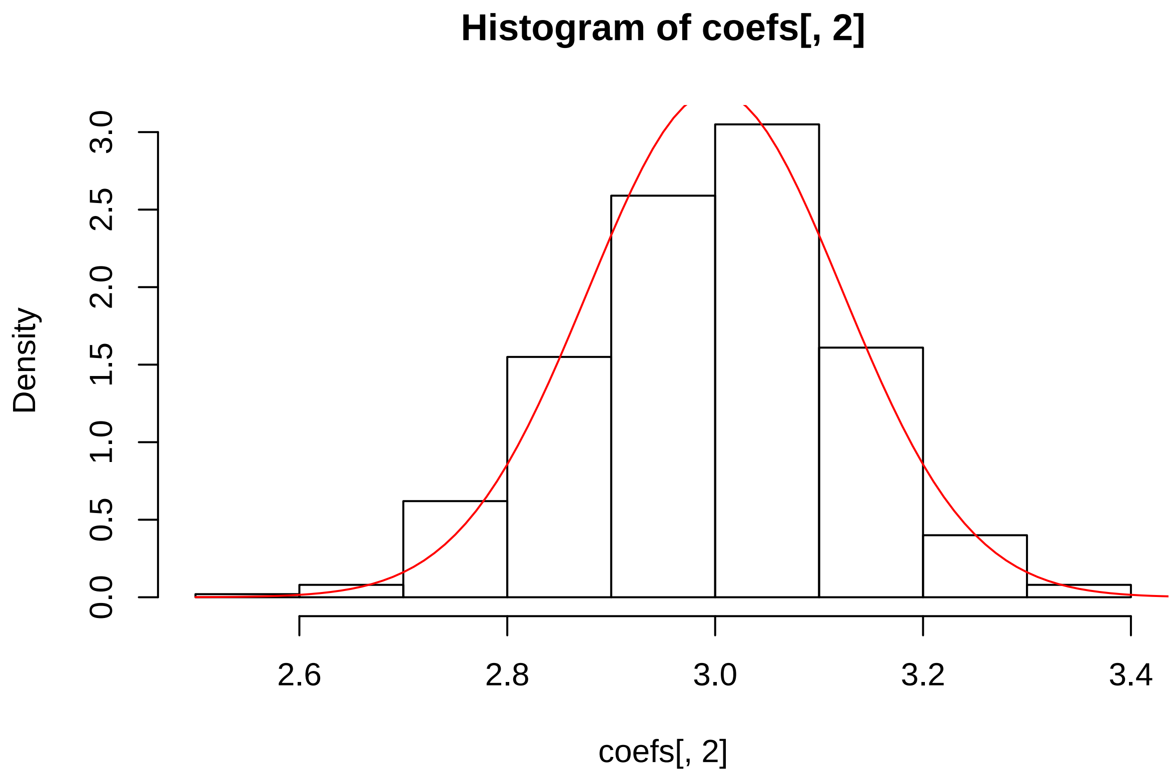
What if X can vary as well?

```
set.seed(21)
nsim <- 1000

coefs <- matrix(nrow=nsim, ncol=2)

for(i in 1:nsim){
  x<-rnorm(n=100, mean=5, sd=4)
  eps <- rnorm(n=100, mean=0, sd=1)
  y <- 2 + 3*x + 5*eps
  reg <- lm(y~x)
  beta <- coef(reg)
  names(beta) <- NULL
  coefs[i,] <- beta
}
```

```
hist(coefs[,1], freq=FALSE)
lines(seq(-2, 5, by = 0.01), dnorm(seq(-2, 5, by = 0.01), mean= 2, sd = 5*sqrt(xtx.inv[1,1])), col="red
```

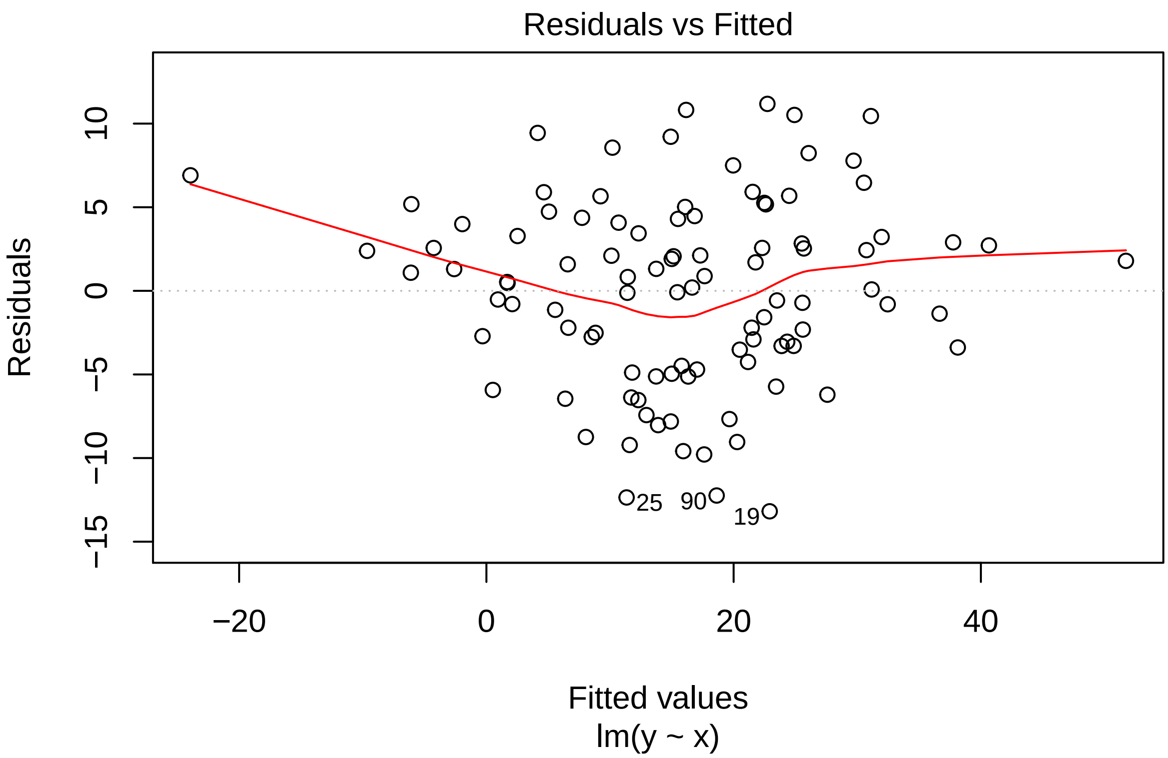


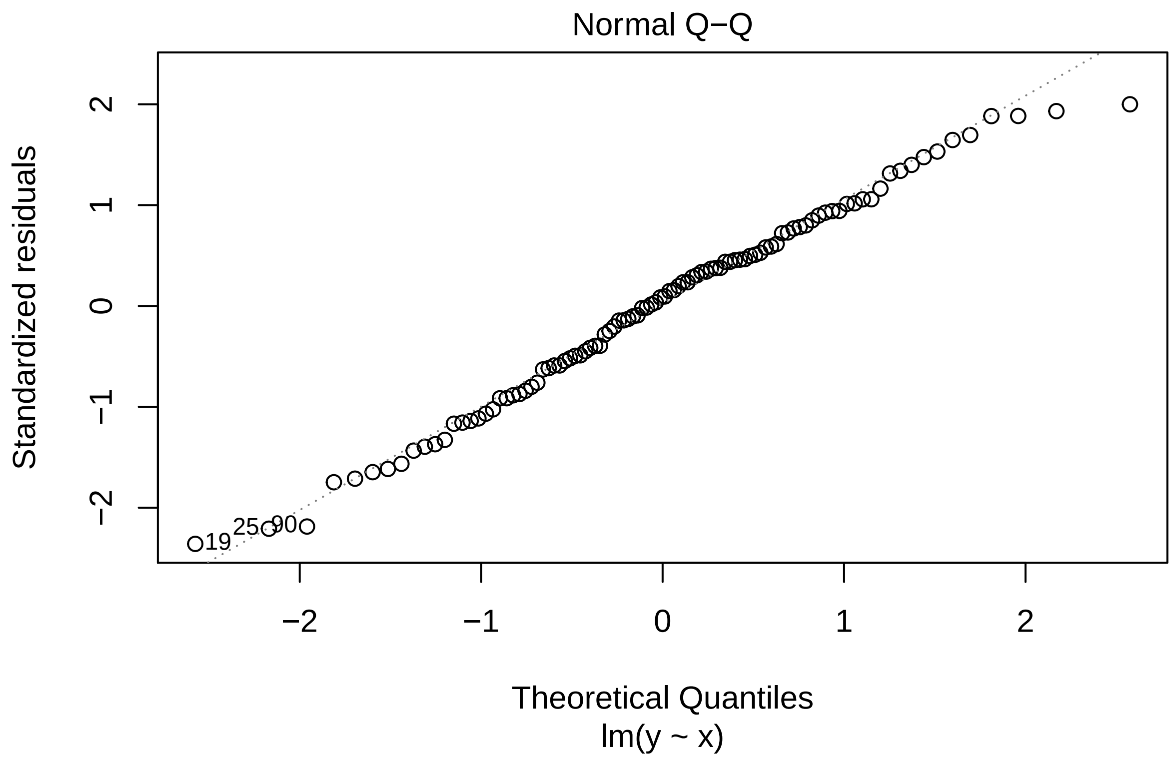

```
hist(coefs[,2], freq = FALSE)
lines(seq(2.5, 3.5, by = 0.01), dnorm(seq(2.5, 3.5, by = 0.01), mean= 3, sd = 5*sqrt(xtx.inv[2,2])),col=
```

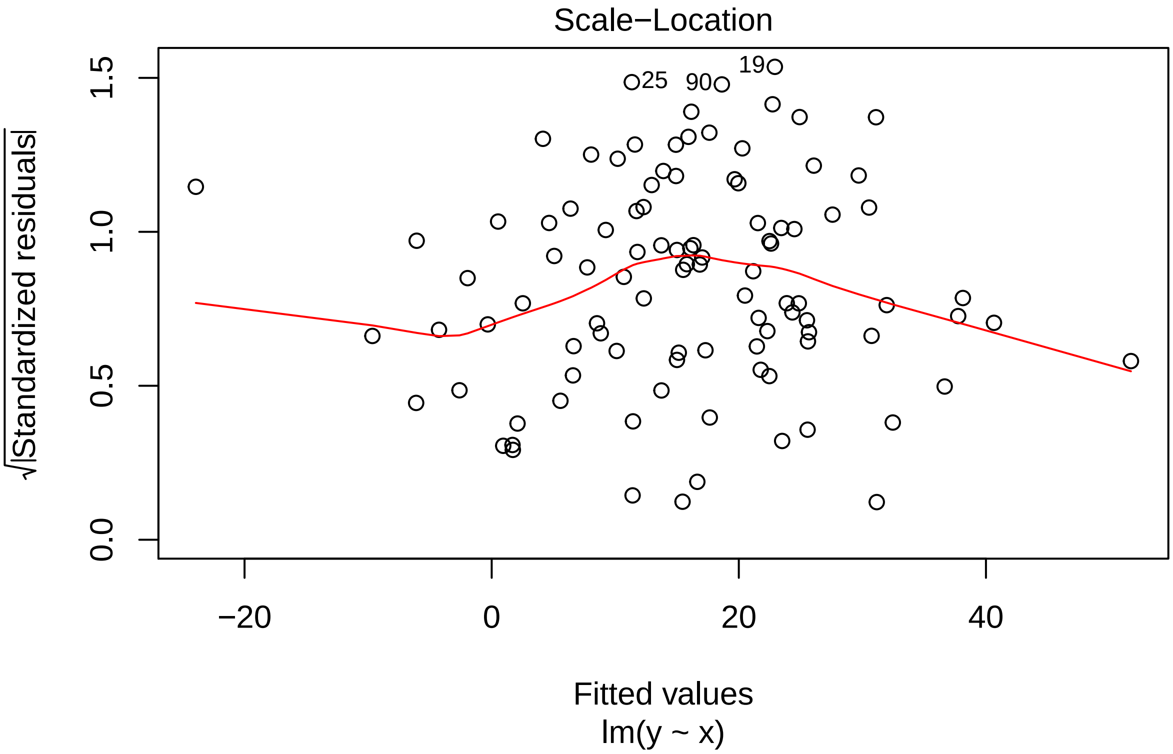


Let's take the last model and test the assumptions!

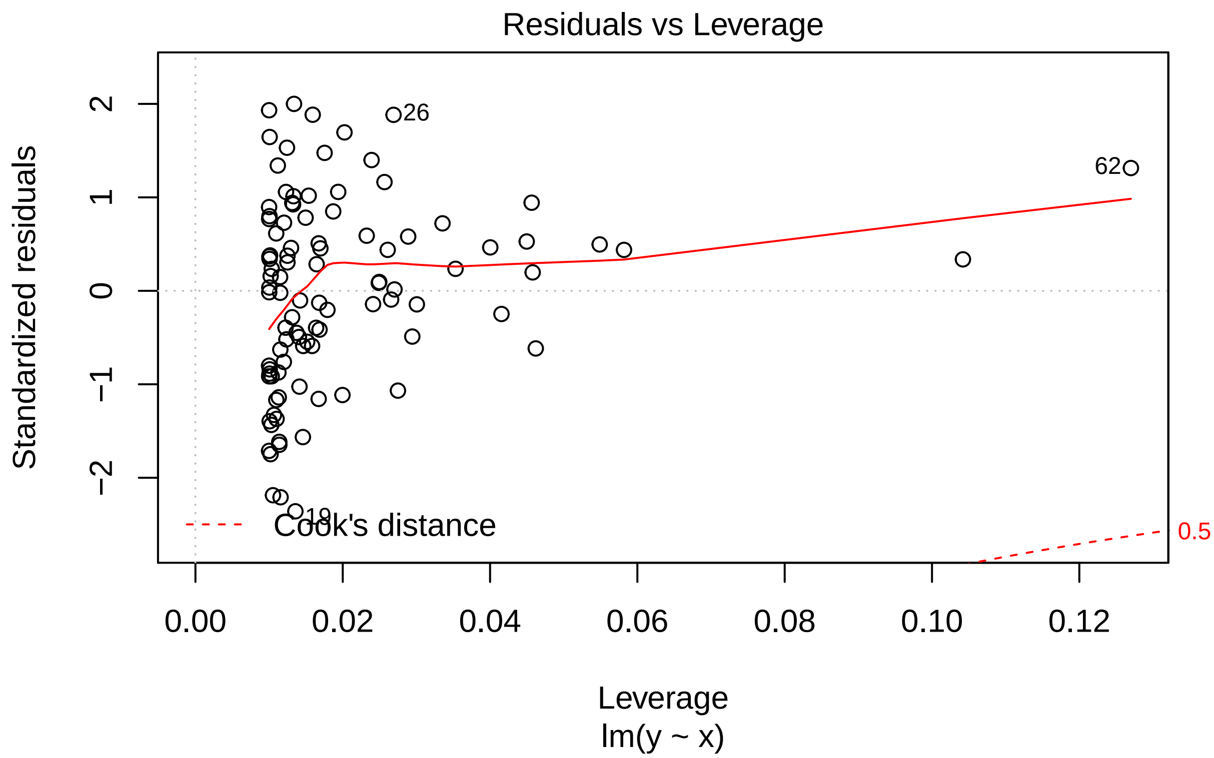
```
plot(reg)
```


Residuals vs Fitted
Residuals
Fitted values
lm(y ~ x)
10
5
0
−5
−10
−15
−20
0
20
40
25
90
19


Normal Q−Q
Standardized residuals
2
1
0
−1
−2
19
25
90
−2
−1
0
1
2
Theoretical Quantiles
lm(y ~ x)



Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
−20
0
20
40
Fitted values
lm(y ~ x)
25
90
19

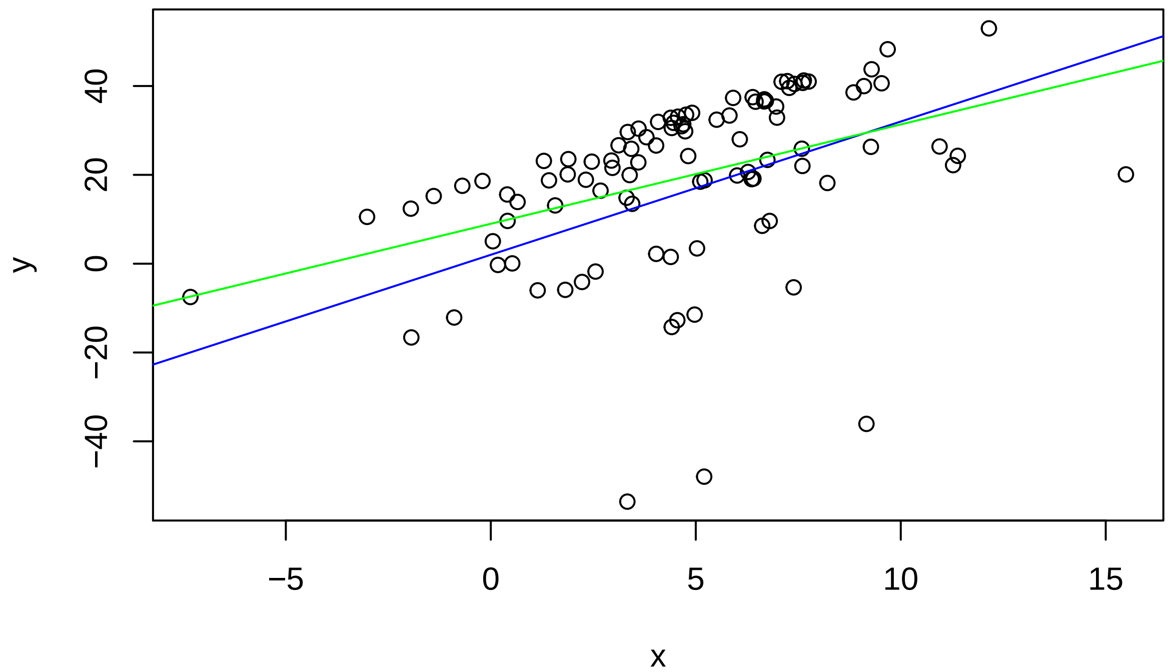


Now let's break some assumptions.

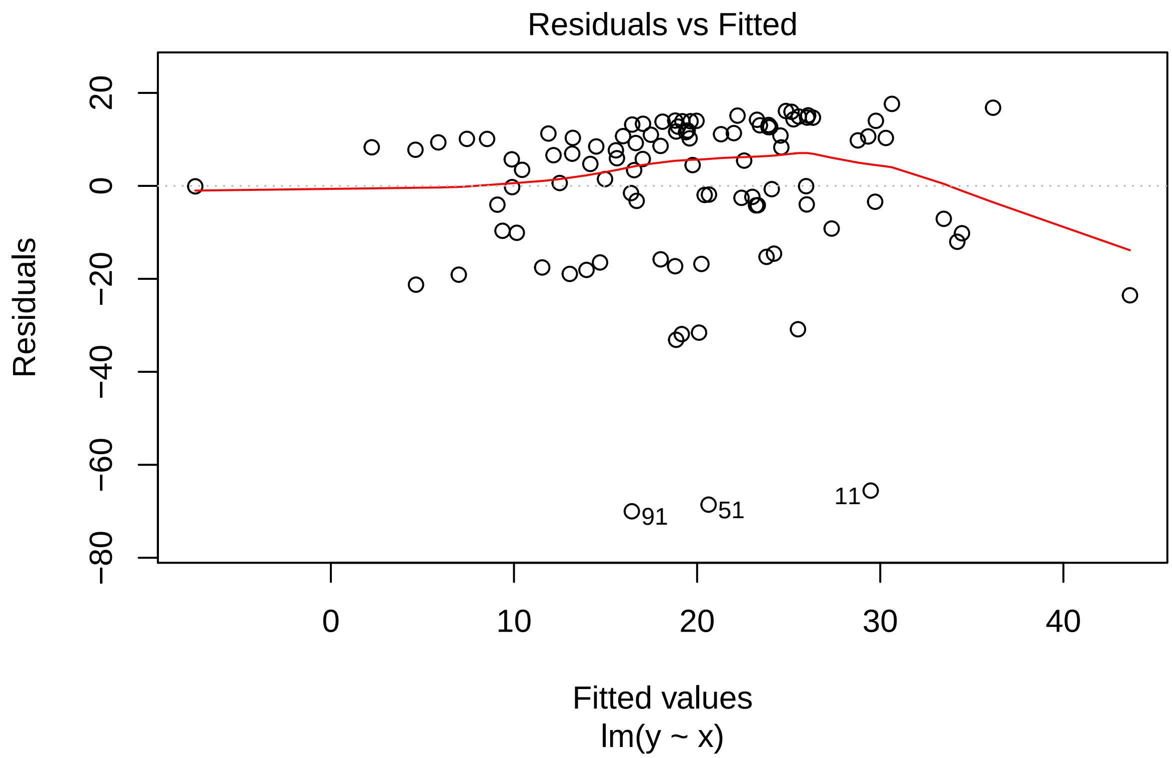
```r
# non normality of the noise
eps <- 5 * (1 - rchisq(40, df = 1)) / sqrt(2)
y <- 2 + 3*x + 5*eps
```

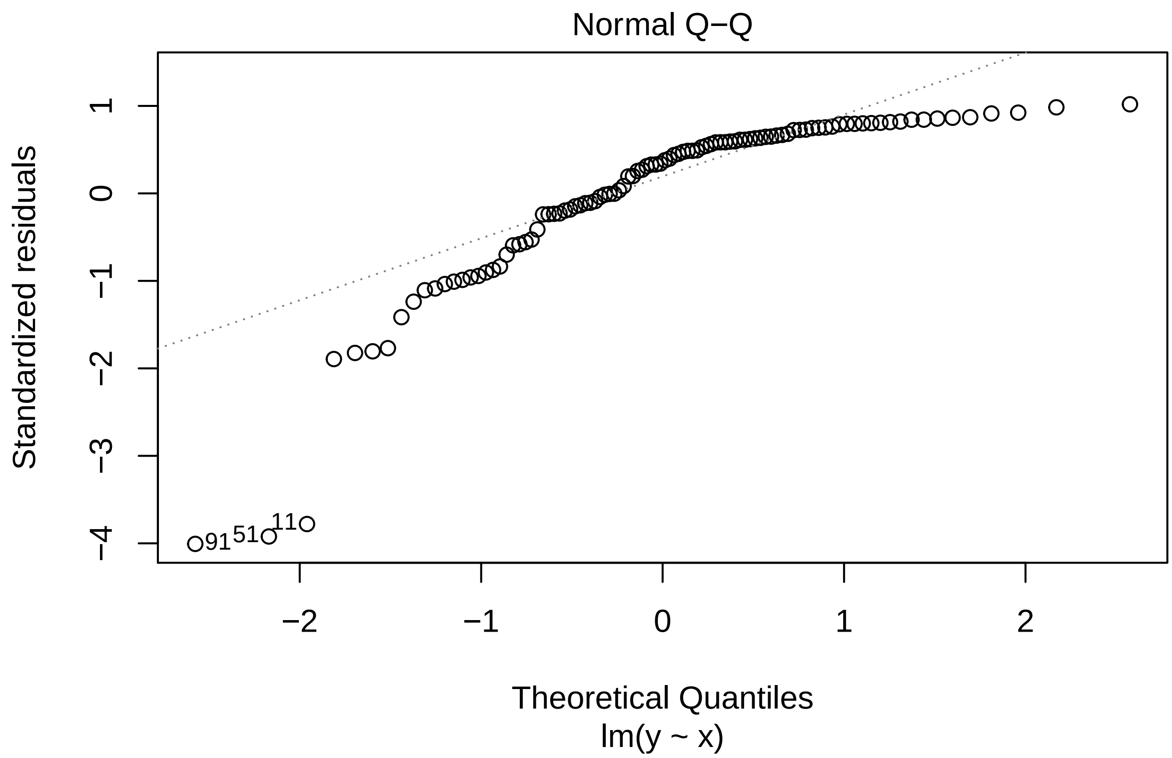
```
## Warning in 2 + 3 * x + 5 * eps: longer object length is not a multiple of
## shorter object length
```

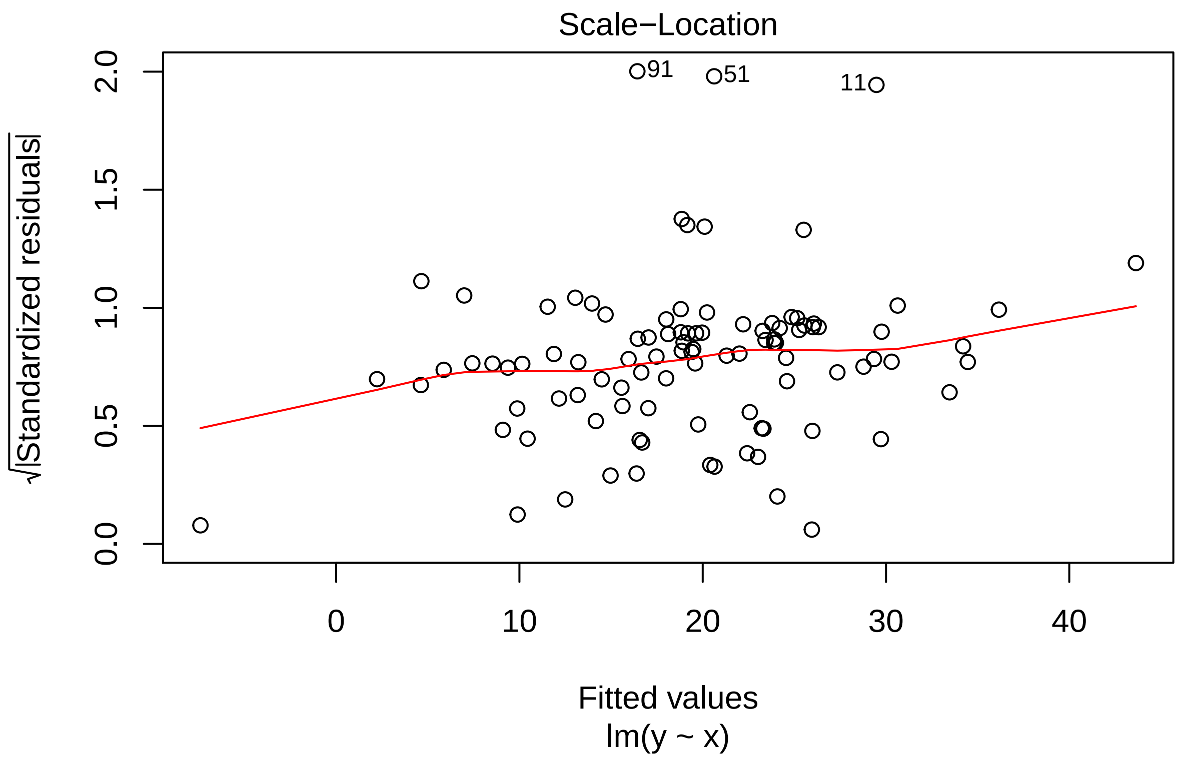
```r
plot(x,y)
abline(a=2, b=3, col="blue")
reg <- lm(y~x)
beta <- coef(reg)
names(beta) <- NULL
abline(a=beta[1], b=beta[2], col="green")
```

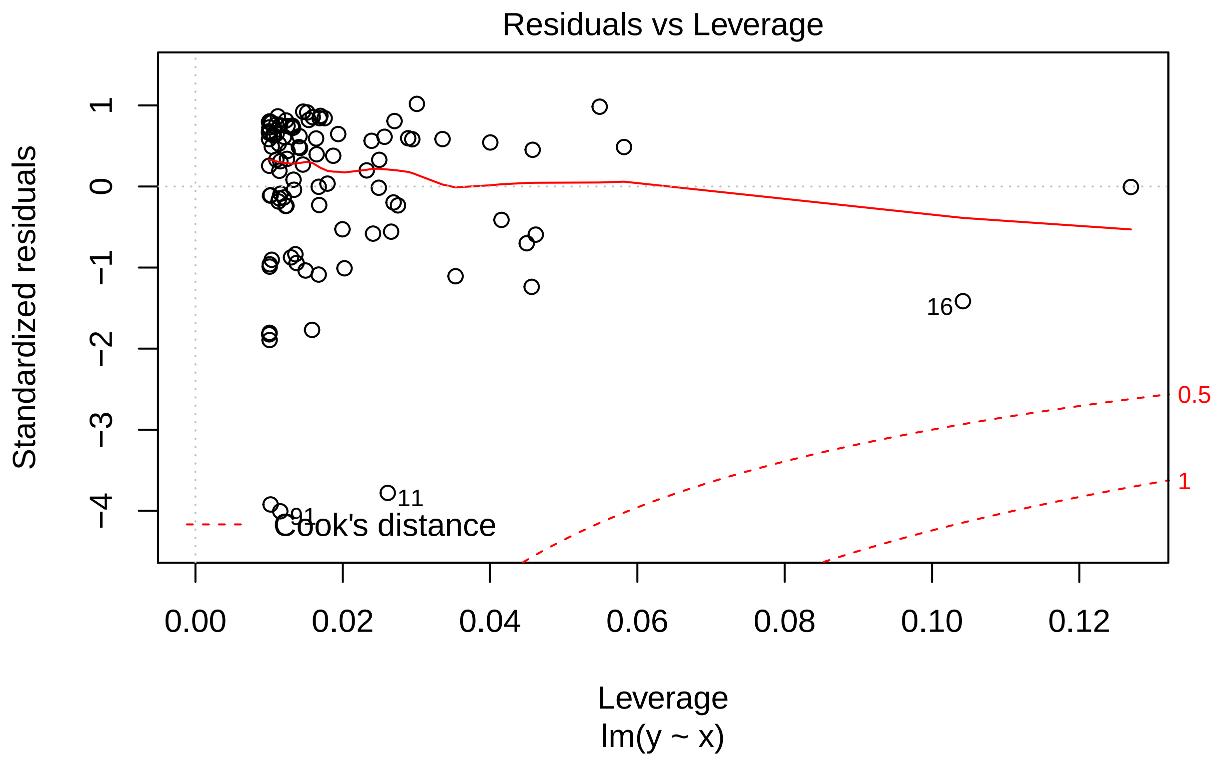



```r
plot(reg)
```


Residuals vs Fitted
Residuals
Fitted values
lm(y ~ x)
20
0
−20
−40
−60
−80
0
10
20
30
40
91
51
11


Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(y ~ x)
91
51
11
1
0
−1
−2
−3
−4
−2
−1
0
1
2

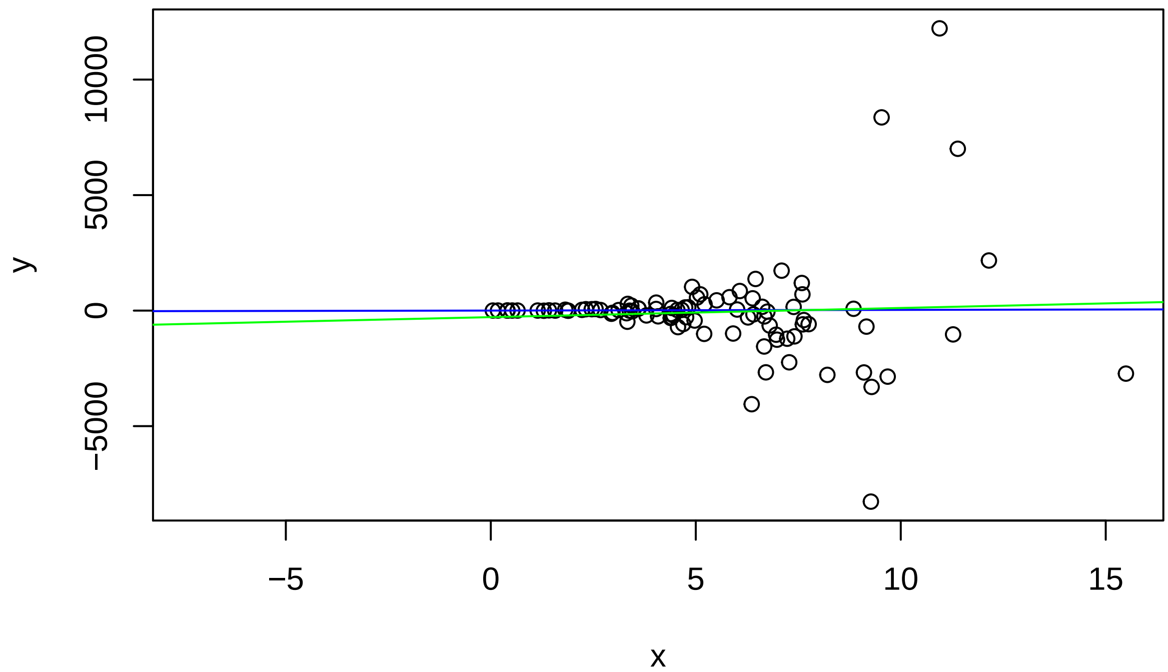

Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
91
51
11
0
10
20
30
40
Fitted values
lm(y ~ x)



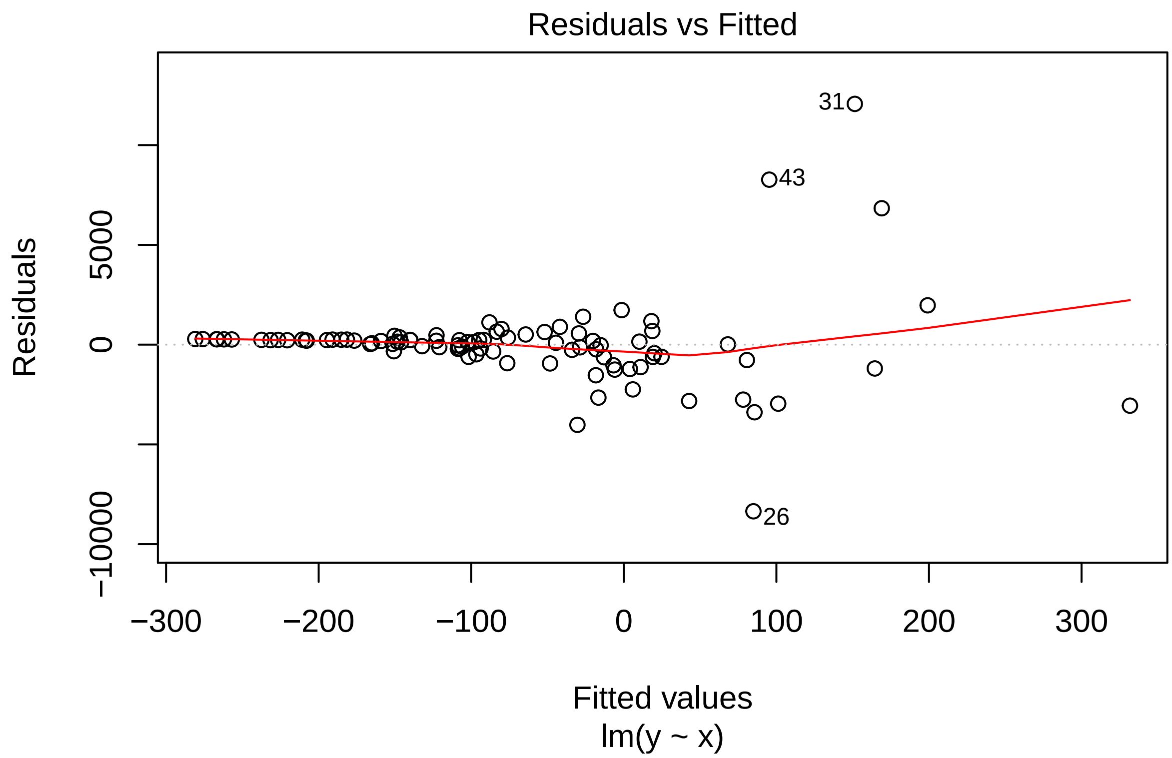

```
# heteroskedasticity
eps <- rnorm(n=100, mean=0, sd= x**3)
```

```
## Warning in rnorm(n = 100, mean = 0, sd = x^3): NAs produced
```

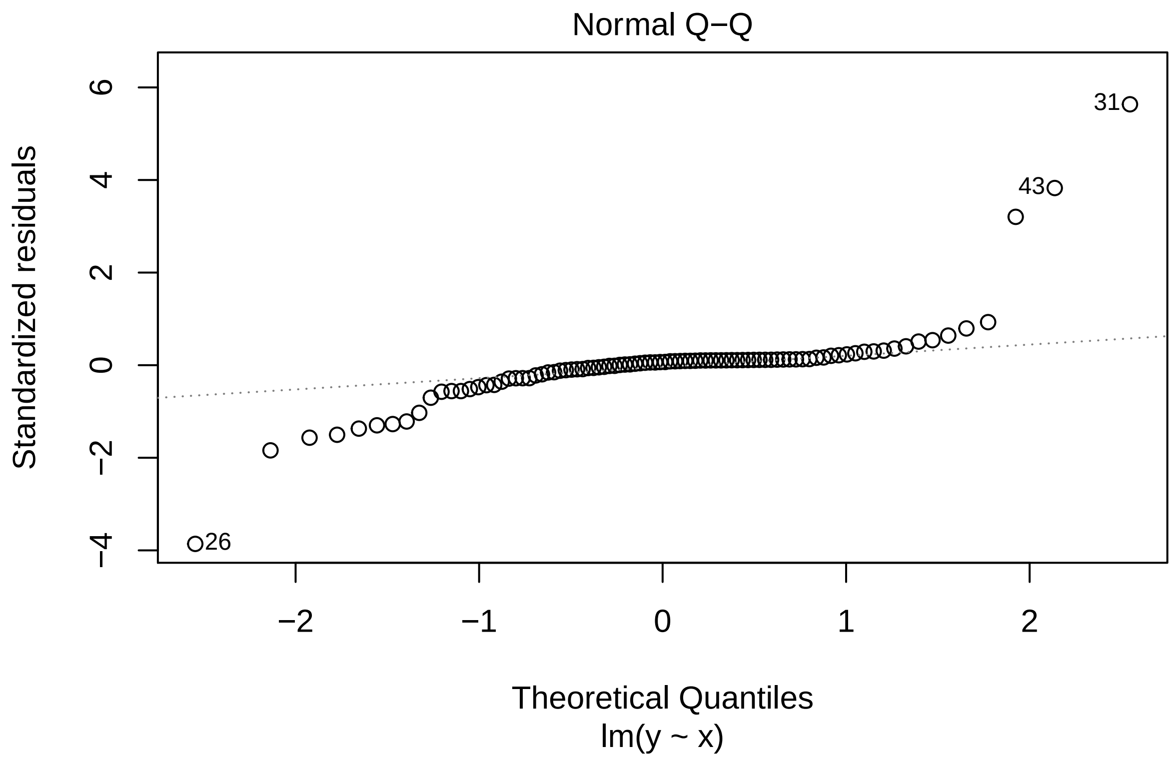
```
y <- 2 + 3*x + 5*eps
plot(x,y)
abline(a=2, b=3, col="blue")
reg <- lm(y~x)
beta <- coef(reg)
names(beta) <- NULL
abline(a=beta[1], b=beta[2], col="green")
```

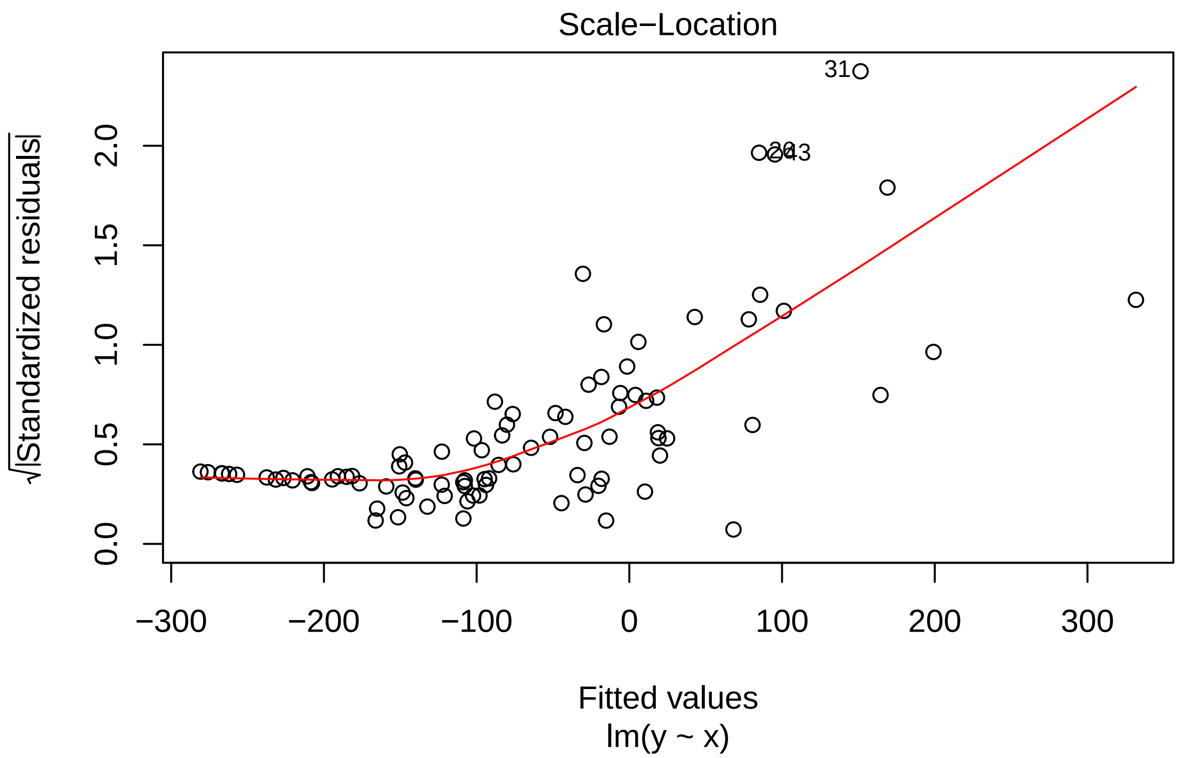


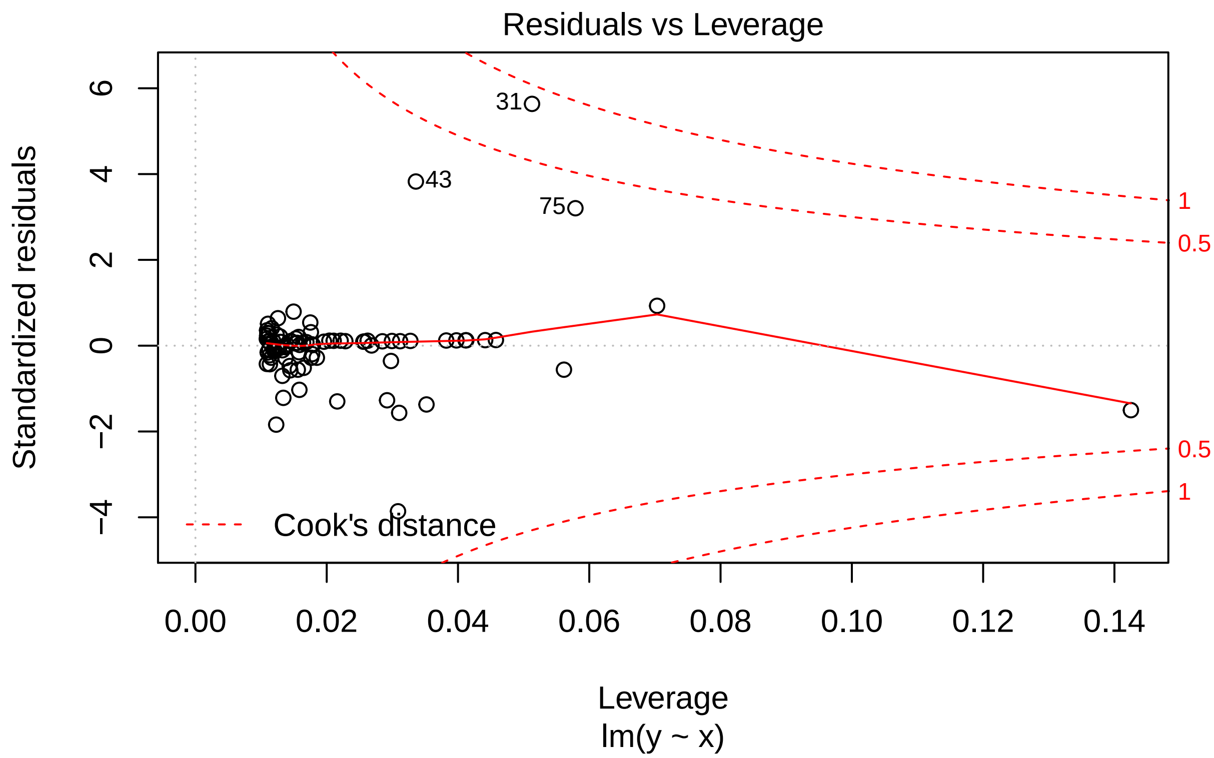

```r
plot(reg)
```



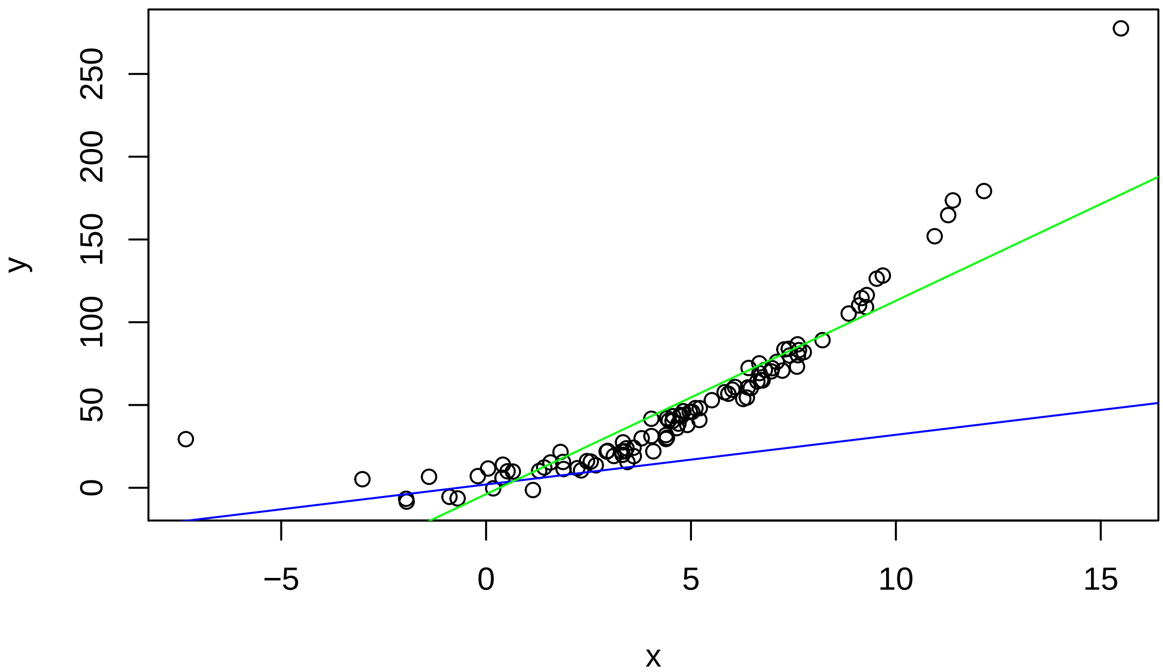
Residuals vs Fitted
Residuals
Fitted values
lm(y ~ x)
-10000
0
5000
-300
-200
-100
0
100
200
300
31
43
26


Normal Q−Q
Standardized residuals
6
4
2
0
−2
−4
31
43
26
−2
−1
0
1
2
Theoretical Quantiles
lm(y ~ x)

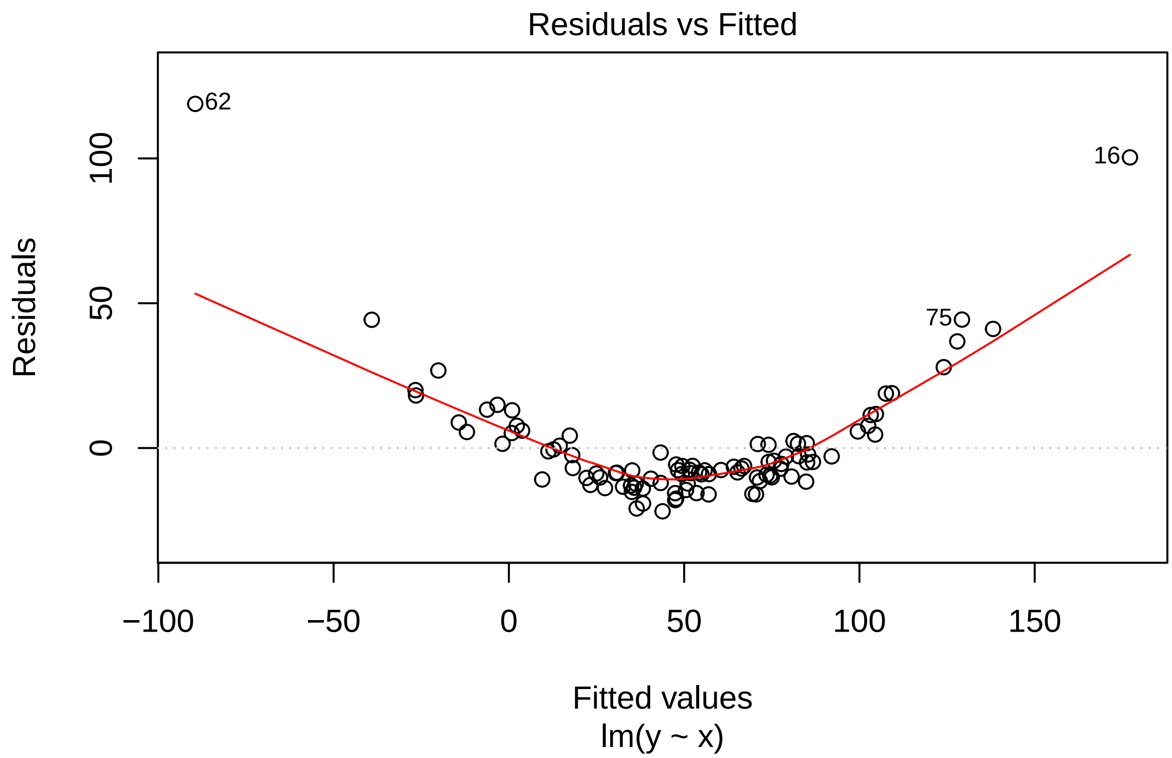

Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
−300
−200
−100
0
100
200
300
31
20
43
Fitted values
lm(y ~ x)

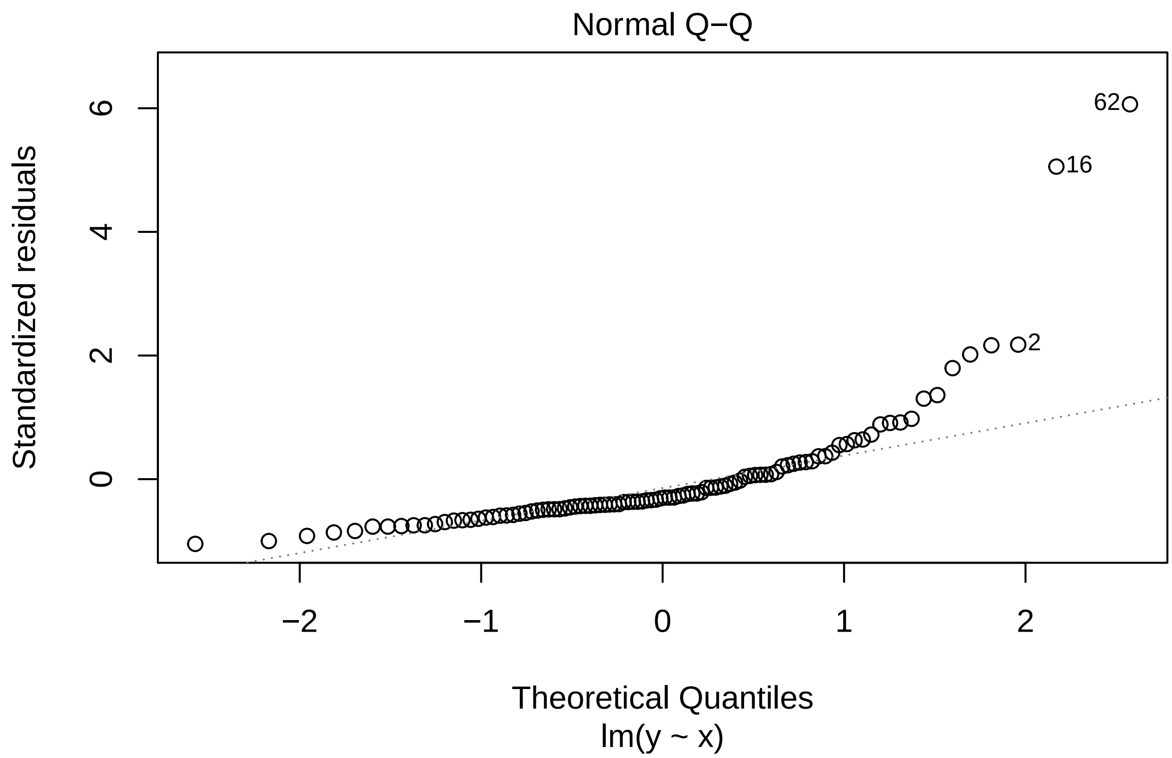


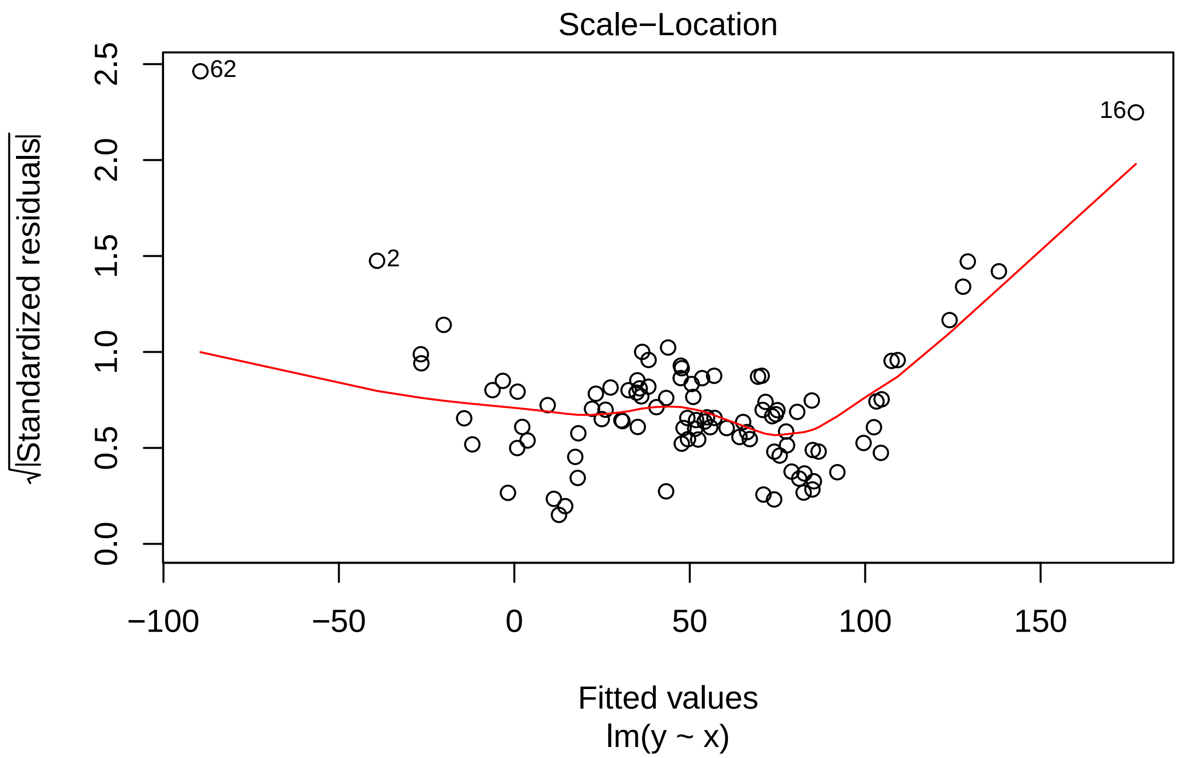

```r
# Missing squared term
eps <- rnorm(n=100, mean=0, sd= 1)
y <- 2 + 3*x + x**2  + 5*eps
plot(x,y)
abline(a=2, b=3, col="blue")
reg <- lm(y~x)
beta <- coef(reg)
names(beta) <- NULL
abline(a=beta[1], b=beta[2], col="green")
```

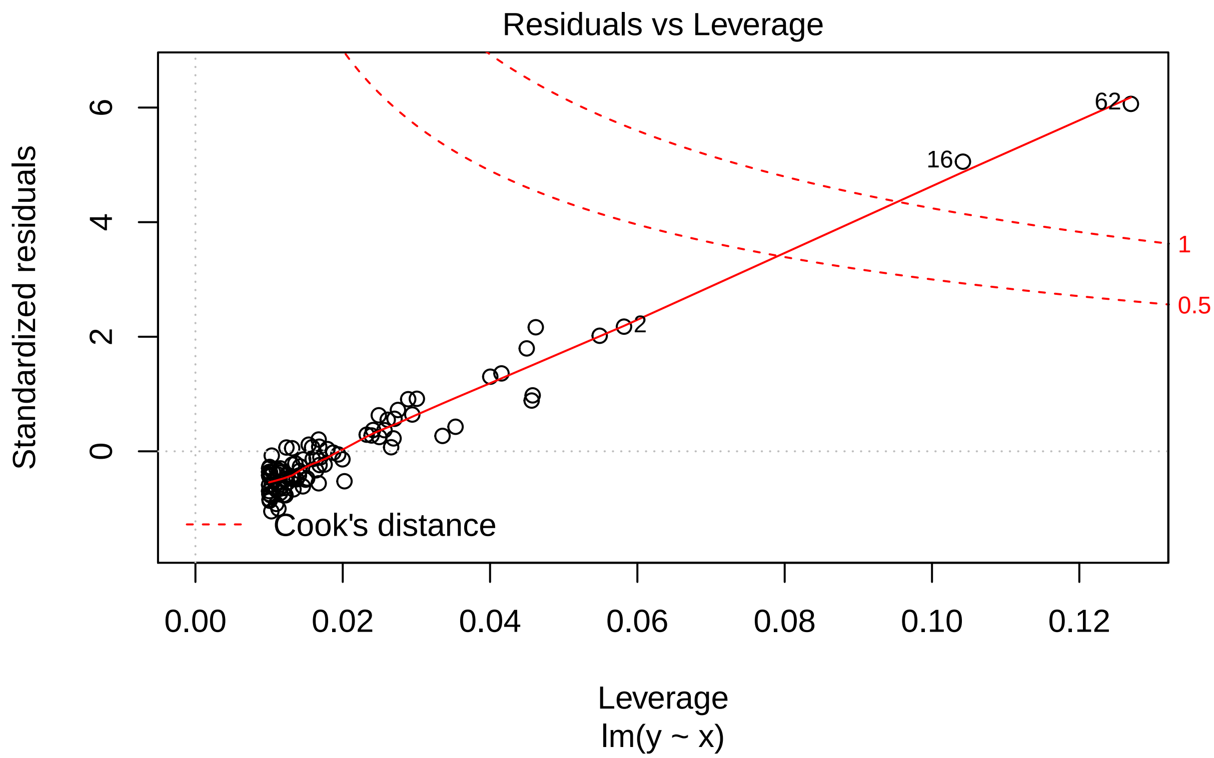



```
plot(reg)
```


Residuals vs Fitted
Residuals
Fitted values
lm(y ~ x)
100
50
0
−100
−50
0
50
100
150
62
16
75


Normal Q−Q
Standardized residuals
6
4
2
0
62
16
2
−2
−1
0
1
2
Theoretical Quantiles
lm(y ~ x)

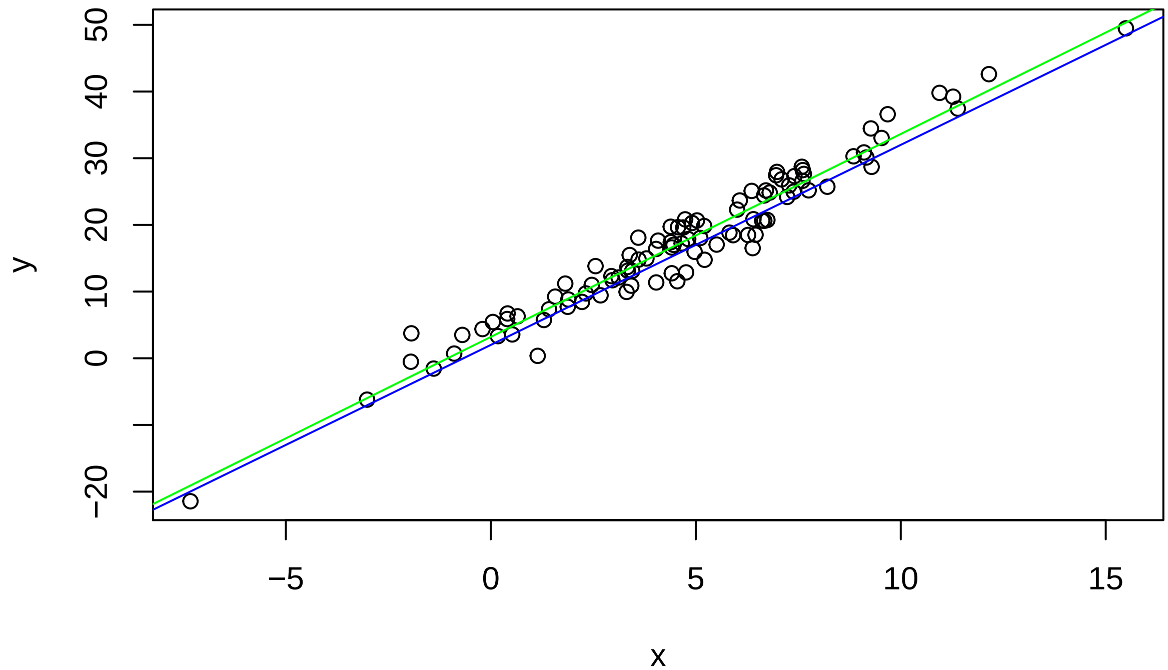

Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
2.5
−100
−50
0
50
100
150
62
2
16
Fitted values
lm(y ~ x)



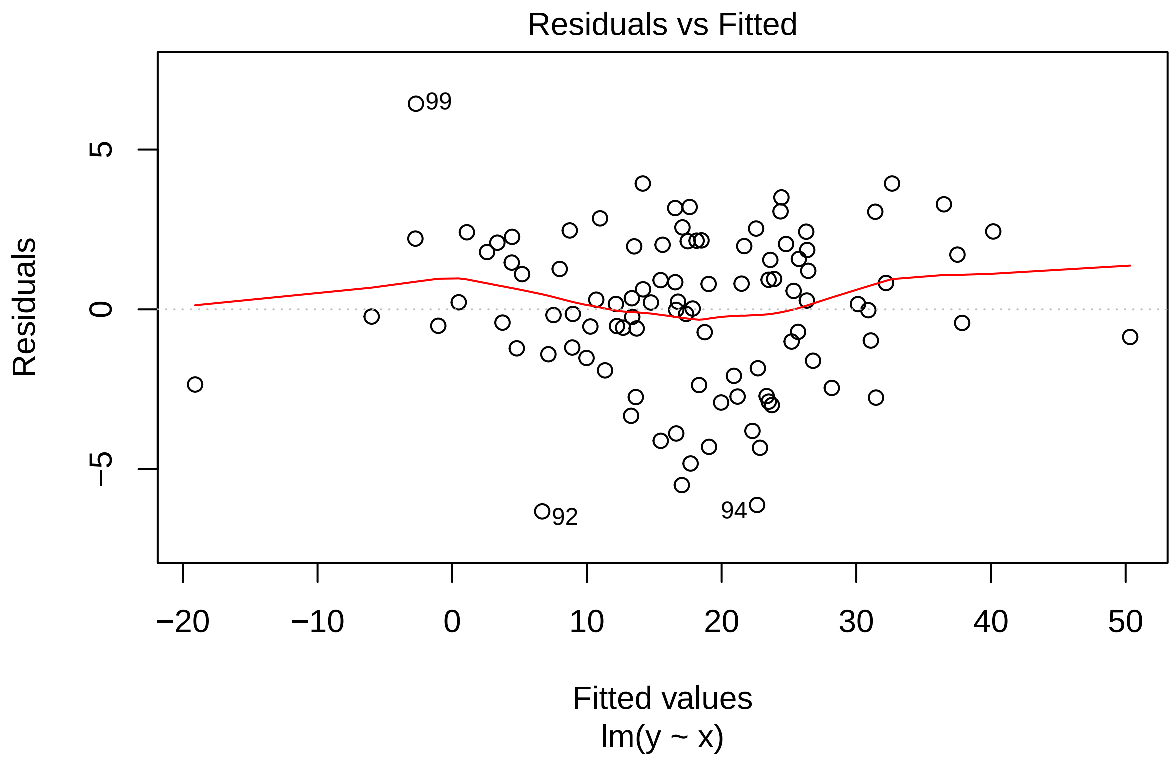

```r
# Correlated errors
require(MASS)
```

```
## Loading required package: MASS
```

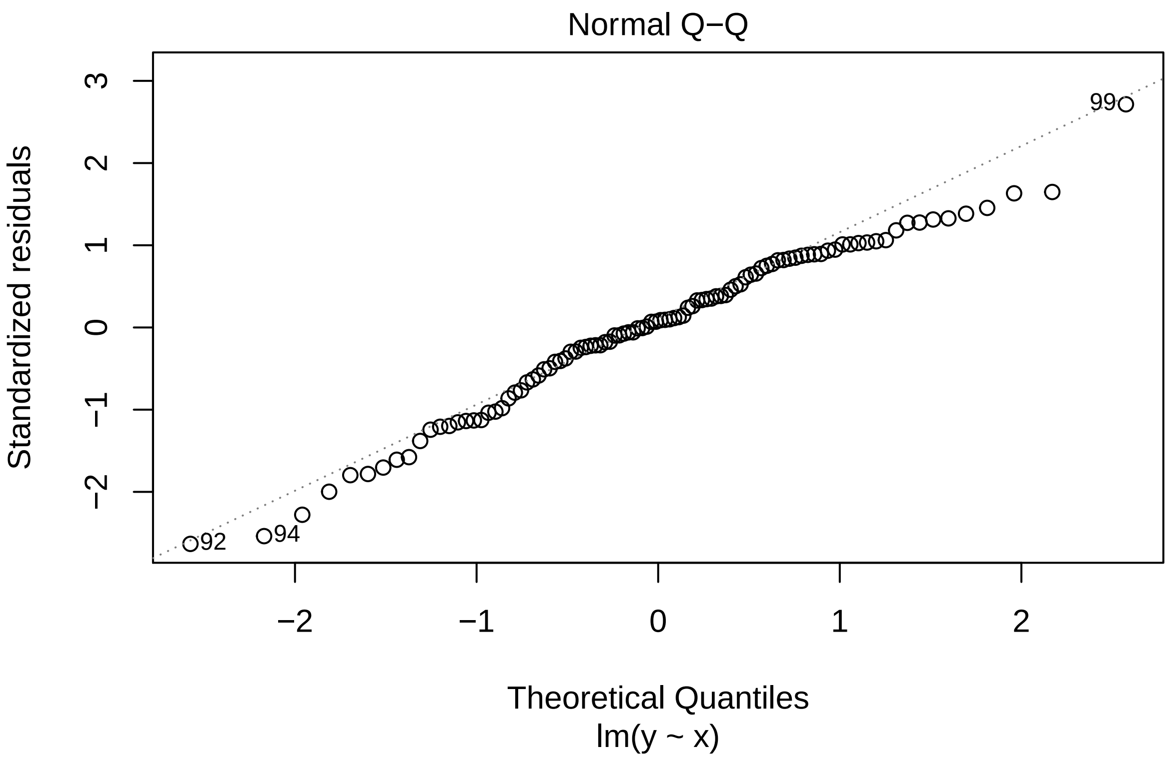
```r
Sigma <- matrix(0.7,100,100)
diag(Sigma) <- 1
eps <- mvrnorm(n = 1, mu = rep(0, length(x)), Sigma = Sigma)
y <- 2 + 3*x + 5*eps
plot(x,y)
abline(a=2, b=3, col="blue")
reg <- lm(y~x)
beta <- coef(reg)
names(beta) <- NULL
abline(a=beta[1], b=beta[2], col="green")
```

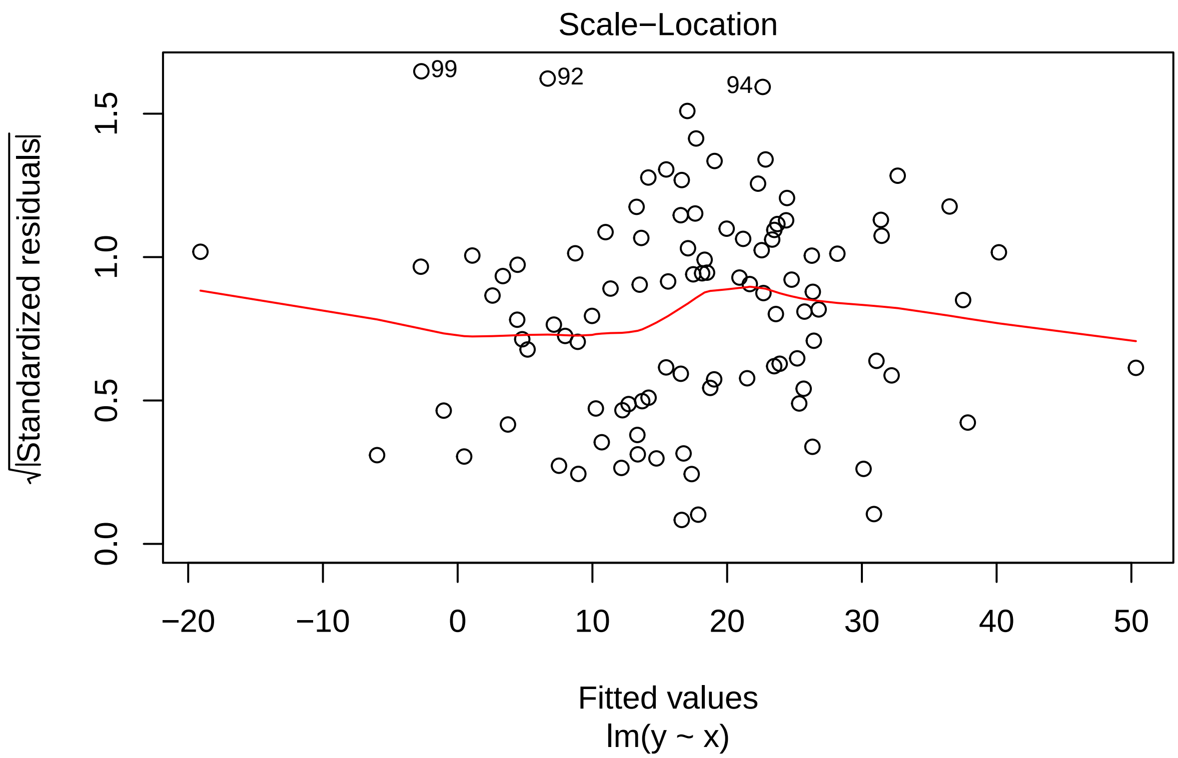



```
plot(reg)
```

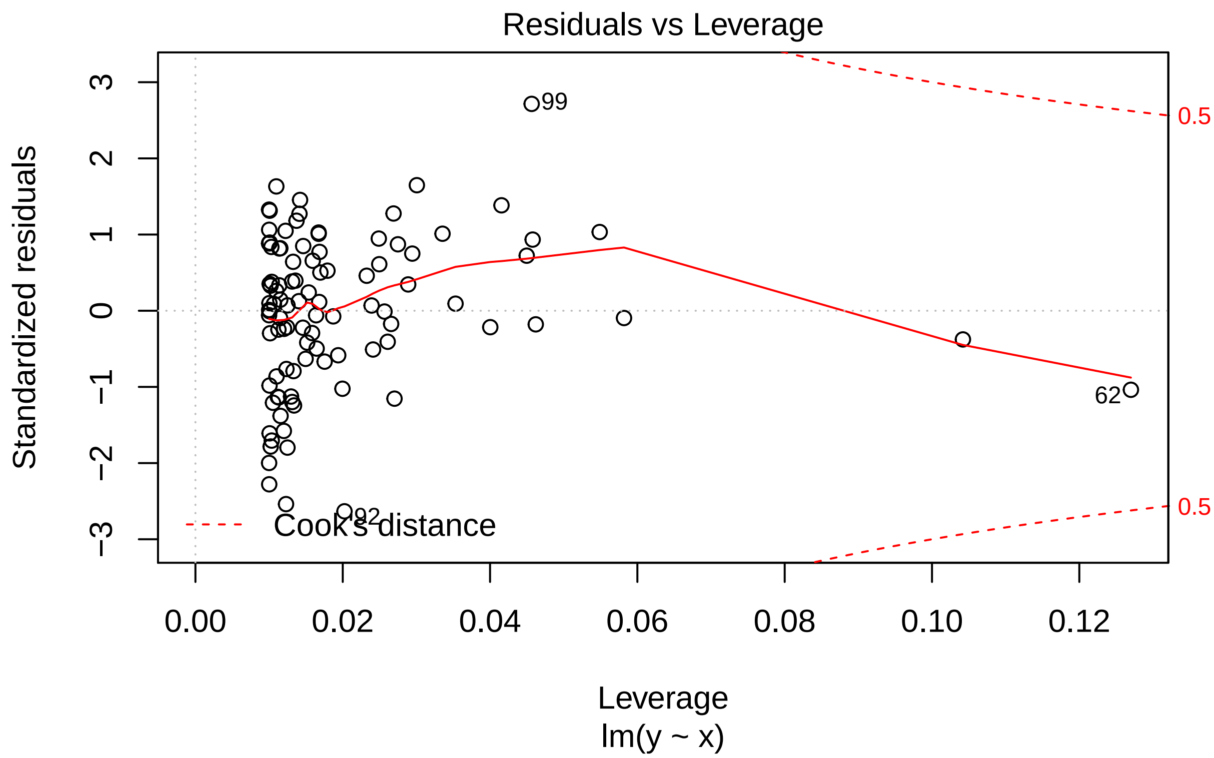


Residuals vs Fitted
Residuals
5
0
−5
99
92
94
−20
−10
0
10
20
30
40
50
Fitted values
lm(y ~ x)

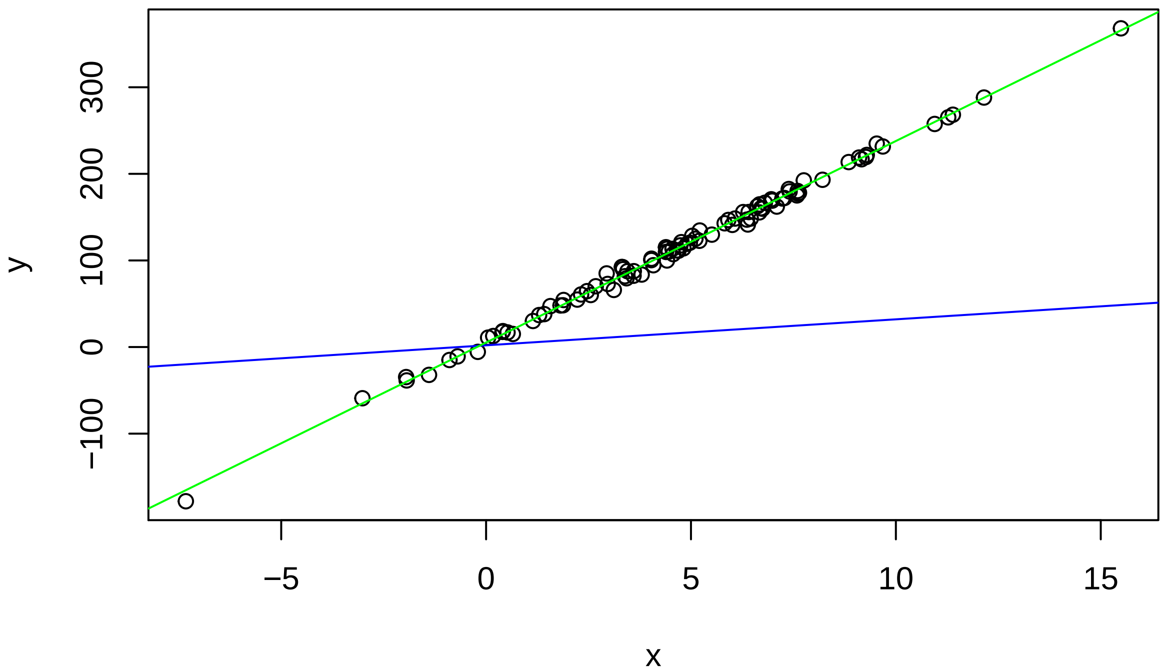

Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(y ~ x)
−2
−1
0
1
2
3
92
94
99



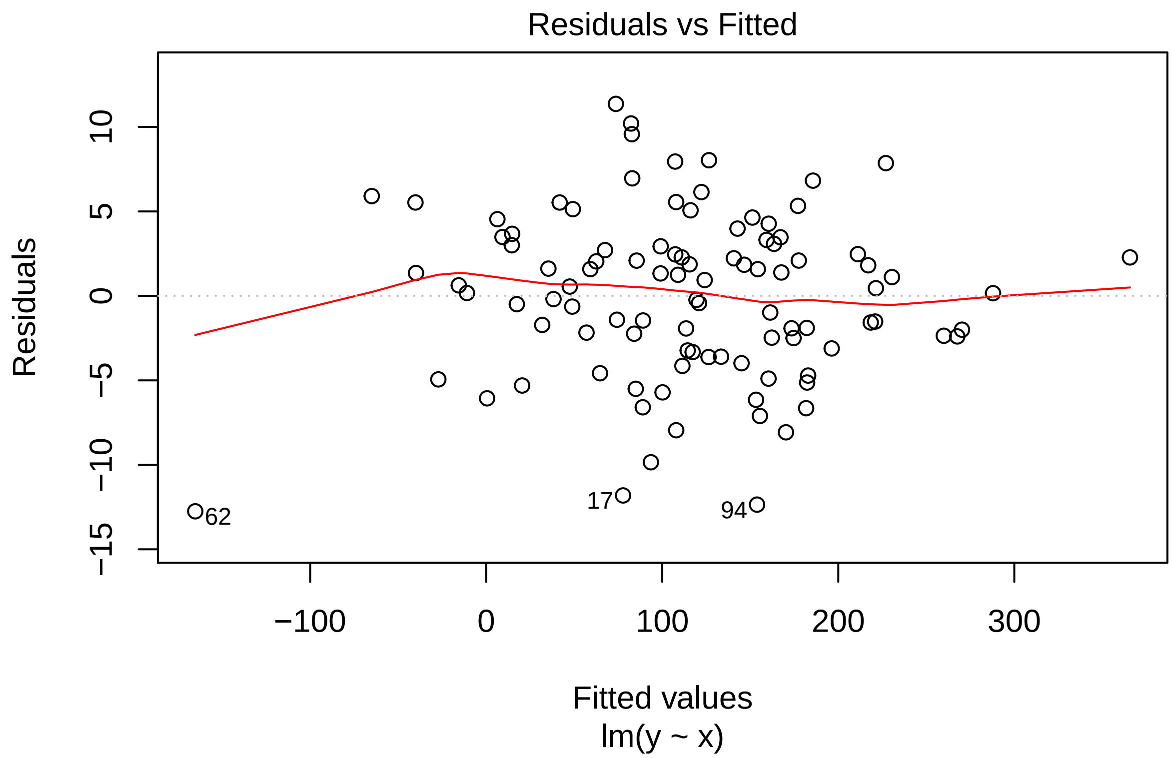
Scale−Location
99
92
94
√|Standardized residuals|
0.0
0.5
1.0
1.5
−20
−10
0
10
20
30
40
50
Fitted values
lm(y ~ x)

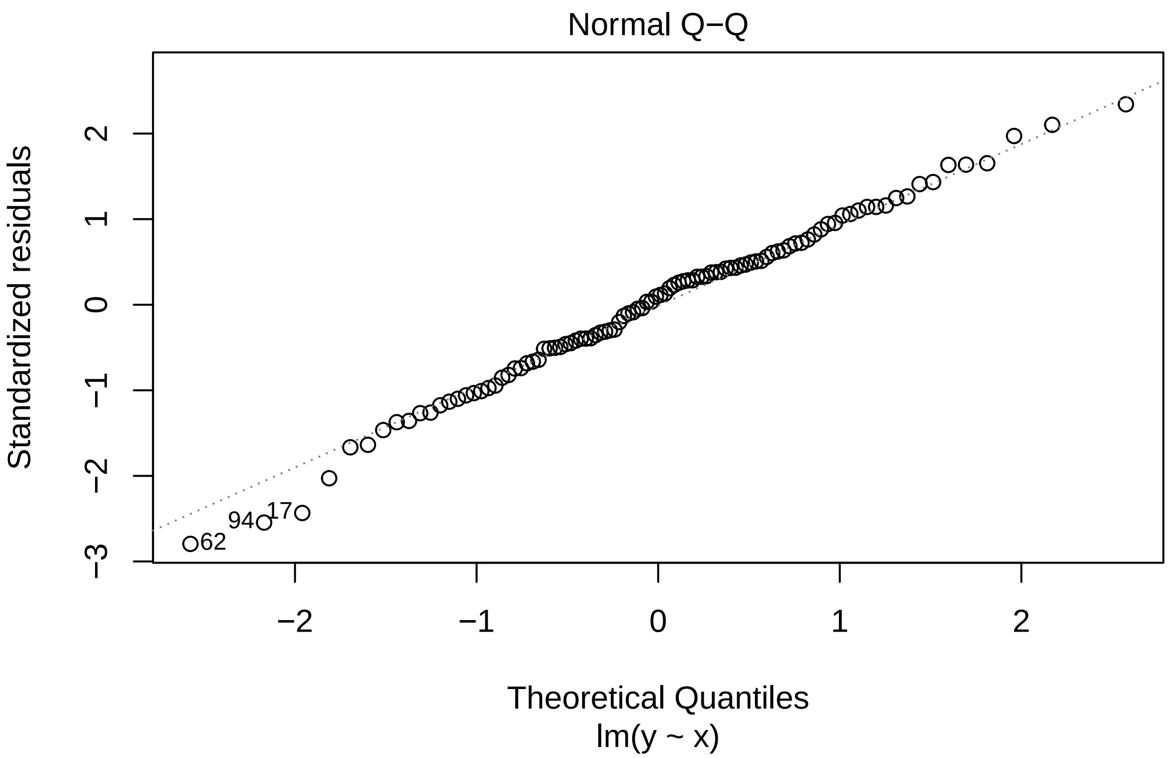


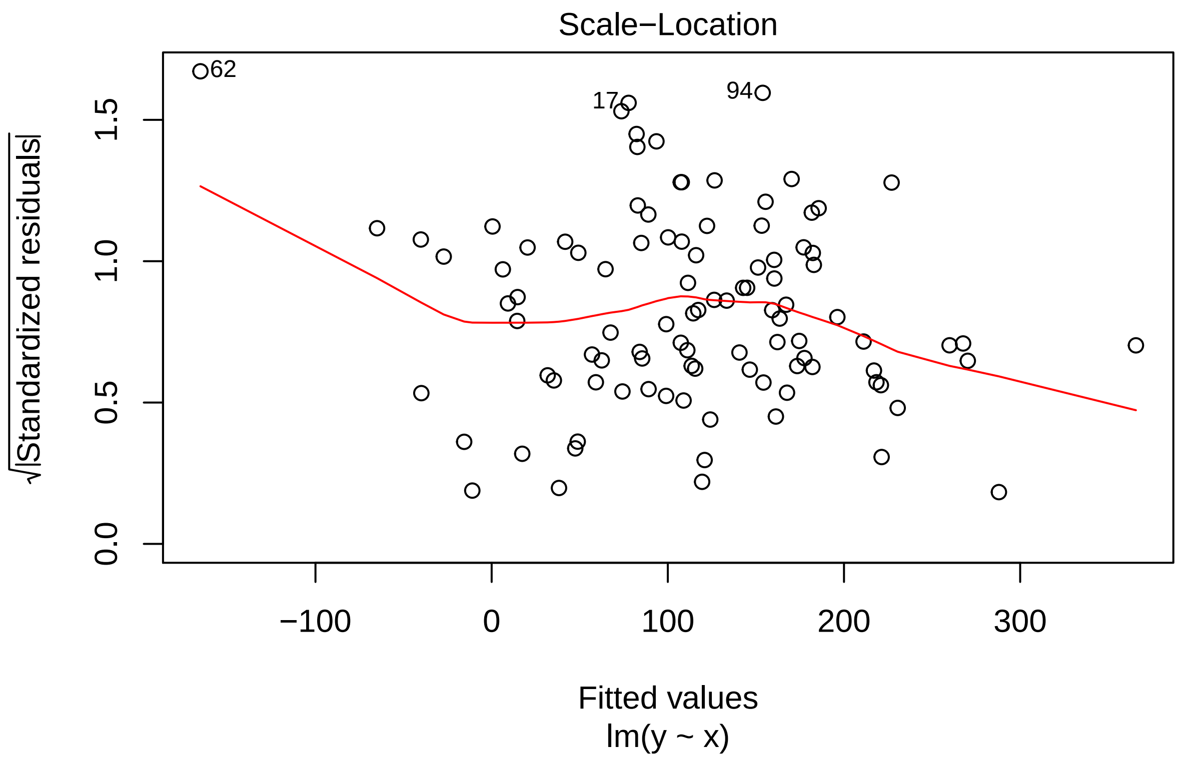

```r
# x, epsilon dependent
eps <- rnorm(n=100, mean=x*4+1, sd= 1)
y <- 2 + 3*x + 5*eps
plot(x,y)
abline(a=2, b=3, col="blue")
reg <- lm(y~x)
beta <- coef(reg)
names(beta) <- NULL
abline(a=beta[1], b=beta[2], col="green")
```



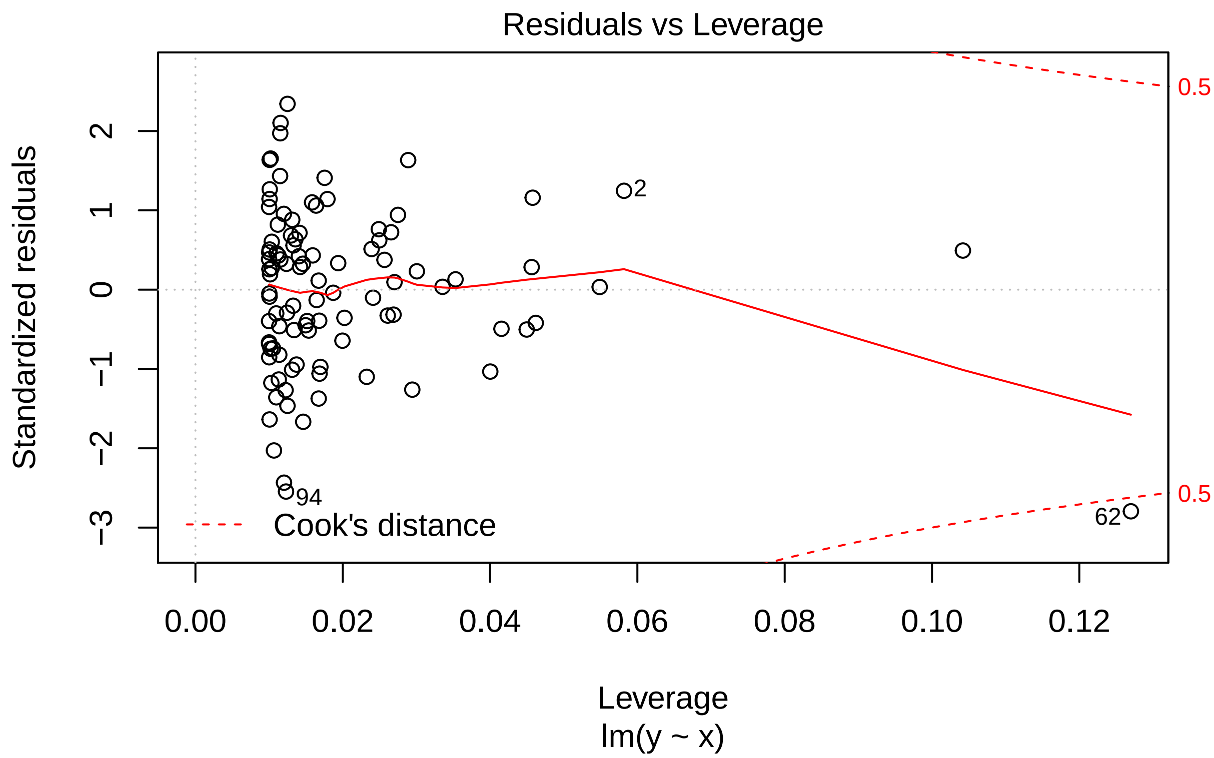

```
plot(reg)
```


Residuals vs Fitted
Residuals
10
5
0
−5
−10
−15
62
17
94
−100
0
100
200
300
Fitted values
lm(y ~ x)


Normal Q−Q
Standardized residuals
−3
−2
−1
0
1
2
62
94
17
−2
−1
0
1
2
Theoretical Quantiles
lm(y ~ x)



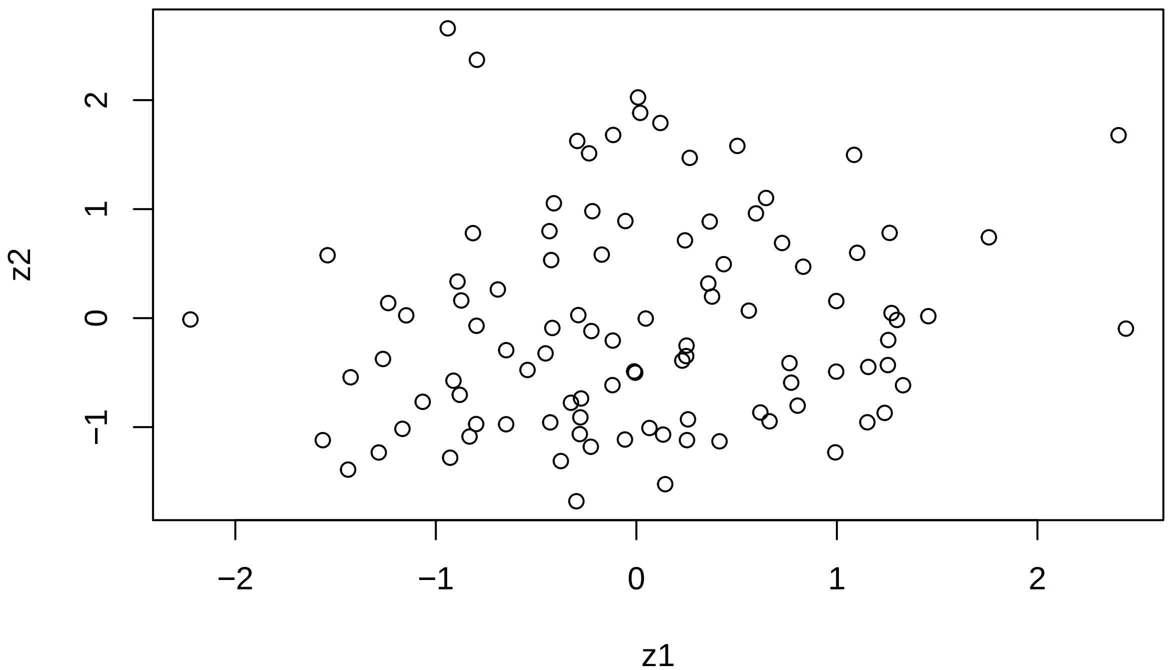
Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
−100
0
100
200
300
Fitted values
lm(y ~ x)
62
17
94



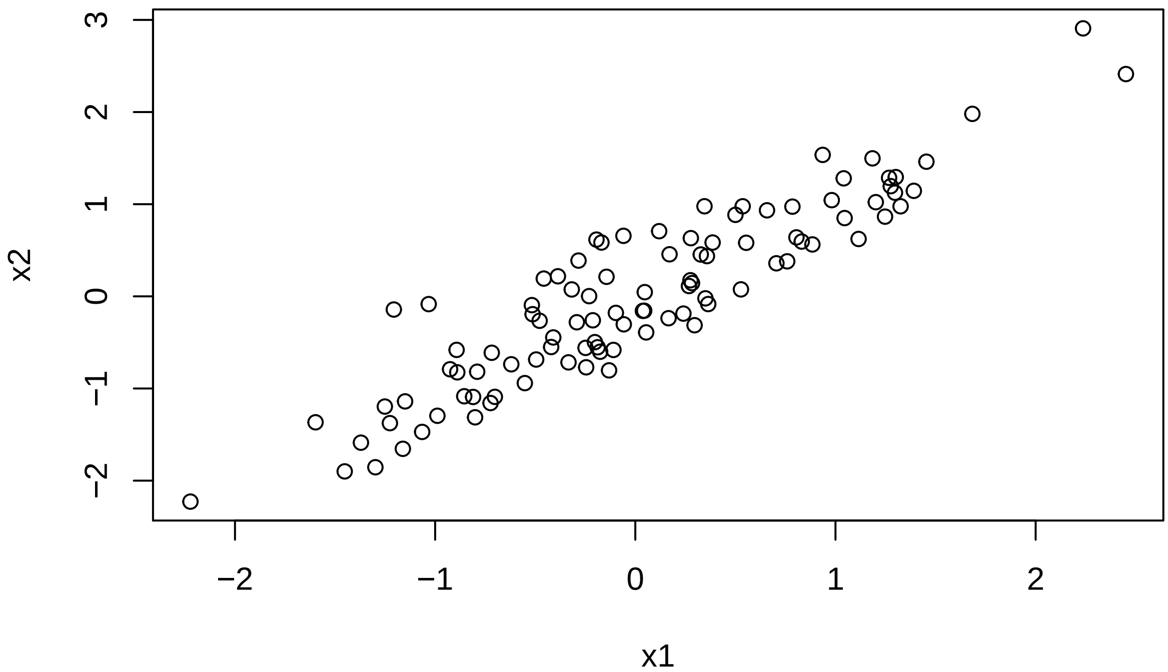
Linear regression testing tools : t-test and F-test

Now let's generate a fake dataset to work on p-values and perform and Anova test.

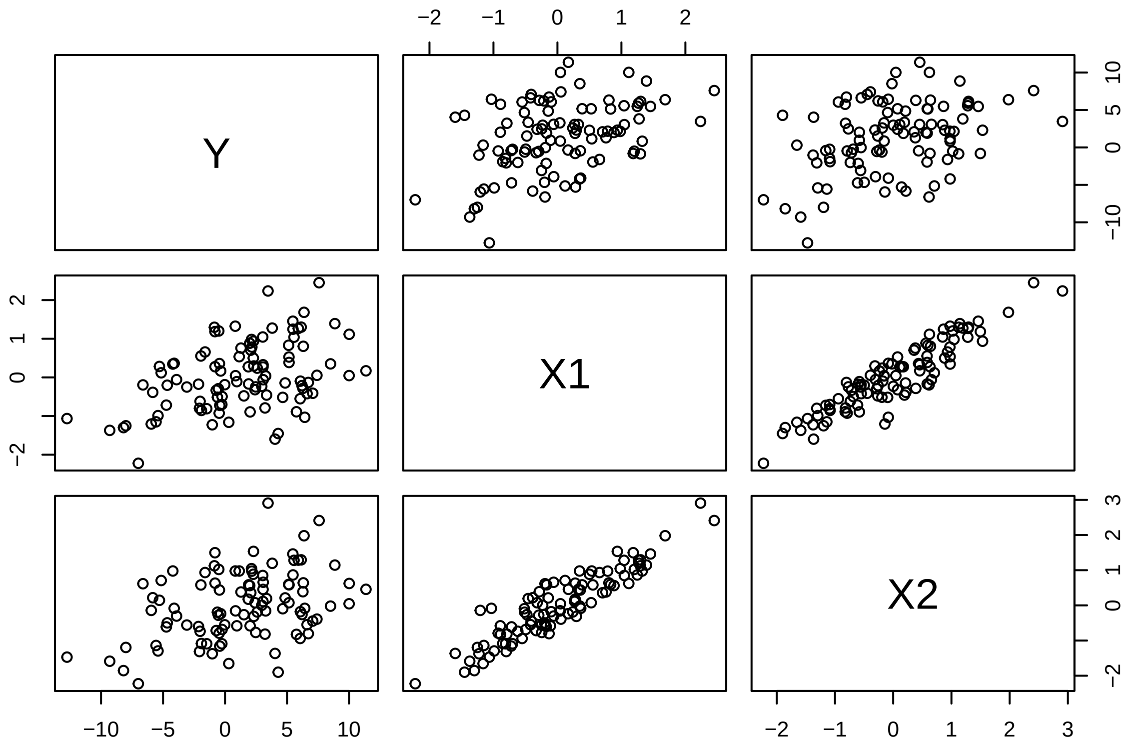
```
set.seed(0)
n <- 100
# two random normal and uncorrelated variables
z1 <- rnorm(n)
z2 <- rnorm(n)
# let's combine them with a linear transformation
TF <- matrix(c(1,1,-0.1,0.3), nrow = 2, ncol = 2)
X <- t(TF%*%rbind(z1,z2))
x1 <- X[,1]
x2 <- X[,2]
plot(z1,z2)
```




```
plot(x1,x2)
```




```r
# white noise
eps <- rnorm(n)
beta <- rbind(1,2,0)
# linear relationship btw Y and X
Y <- cbind(rep(1,n),X)%*%beta + 4*eps
data<-data.frame(X,Y)
pairs(Y~X1 + X2, data = data)
```



Note: - positive correlation btw X1 and X2 - positive correlation btw Y and X1 - weak positive correlation btw Y and X2 (side effect of the 2 above)

Let's fit a linear model and look at the output!

```
reg <- lm(Y~X)
summary(reg)
```

```
##
## Call:
## lm(formula = Y ~ X)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.5816  -2.9387  -0.0731   2.3559   9.7475
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.2581     0.4249   2.961  0.00386 **
## X1            2.4329     1.2146   2.003  0.04796 *
## X2           -0.1121     1.1041  -0.102  0.91935
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.242 on 97 degrees of freedom
## Multiple R-squared:  0.1897, Adjusted R-squared:  0.173
## F-statistic: 11.36 on 2 and 97 DF,  p-value: 3.7e-05
```

Now let's look at all the submodels of "reg".

```r
reg.0 <- lm(Y~1)
summary(reg.0)
```

```
##
## Call:
## lm(formula = Y ~ 1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.0729  -2.4654   0.6845   3.8141  10.0470
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.3233     0.4665   2.837  0.00553 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.665 on 99 degrees of freedom
```

```r
reg.1 <- lm(Y~x1)
summary(reg.1)
```

```
##
## Call:
## lm(formula = Y ~ x1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.5390  -2.9838  -0.0378   2.3469   9.7136
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.2602     0.4223   2.984  0.00359 **
## x1            2.3200     0.4844   4.789 5.94e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.221 on 98 degrees of freedom
## Multiple R-squared:  0.1897, Adjusted R-squared:  0.1814
## F-statistic: 22.94 on 1 and 98 DF,  p-value: 5.945e-06
```

```r
reg.2 <- lm(Y~x2)
summary(reg.2)
```

```
##
## Call:
## lm(formula = Y ~ x2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.2393  -2.7747   0.1608   2.5496   9.1909
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)   1.3061     0.4307   3.033   0.0031 **
## x2            1.9141     0.4494   4.260  4.7e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.307 on 98 degrees of freedom
## Multiple R-squared:  0.1562, Adjusted R-squared:  0.1476
## F-statistic: 18.14 on 1 and 98 DF,  p-value: 4.704e-05
```

Notice that when in the model alone x2 not only has a positive coefficient but it's also estimated to be significant in the model!

Indeed to get out the coefficient estimated in the full model fot x2 we need to perform the following:

```
res.y1 <- reg.1$residuals
cor21 <- lm(x2~x1)
res.21 <- cor21$residuals
# what x1 cannot explain in the y vs what x1 cannot explain of x2
reg.2.res <- lm(res.y1~res.21)
summary(reg.2.res)
```

```
##
## Call:
## lm(formula = res.y1 ~ res.21)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.5816  -2.9387  -0.0731   2.3559   9.7475
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.677e-16  4.220e-01   0.000    1.000
## res.21      -1.121e-01  1.098e+00  -0.102    0.919
##
## Residual standard error: 4.22 on 98 degrees of freedom
## Multiple R-squared:  0.0001062,  Adjusted R-squared:  -0.0101
## F-statistic: 0.01041 on 1 and 98 DF,  p-value: 0.9189
```

Now back to the full model: ### how do we obtain the R-squared and F-test values? Let's do it by hand.

```
reg.res <- reg$residuals
y.hat <- reg$fitted.values
y.mean <- mean(Y)
TSS <- sum((Y-y.mean)**2)
RSS <- sum((reg.res)**2)
MSS <- sum((y.hat - y.mean)**2)
# R-squared
print("R squared:")
```

```
## [1] "R squared:"
```

```
(R2 <- 1 - (RSS/TSS))
```

```
## [1] 0.1897415
```

```
# or equivalently
(MSS/TSS)
```

```
## [1] 0.1897415
```

```r
print("Adjusted R squared")
```

```
## [1] "Adjusted R squared"
```

```r
(R2.adj <- 1 - ((RSS/(n-3))/(TSS/(n-1)))) #accounting for the flexibility of the models
```

```
## [1] 0.1730351
```

```r
print("F statistic:")
```

```
## [1] "F statistic:"
```

```r
(F.stat <- (MSS/(3-1))/(RSS/(n-3)))
```

```
## [1] 11.35744
```

```r
print("p-value:")
```

```
## [1] "p-value:"
```

```r
(F.pv <- pf(F.stat, df1 =2, df2=(n-1), lower.tail = FALSE))
```

```
## [1] 3.626025e-05
```

Compare it with the values in the R output:

```r
summary(reg)
```

```
##
## Call:
## lm(formula = Y ~ X)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.5816  -2.9387  -0.0731   2.3559   9.7475
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.2581     0.4249   2.961  0.00386 **
## X1            2.4329     1.2146   2.003  0.04796 *
## X2           -0.1121     1.1041  -0.102  0.91935
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.242 on 97 degrees of freedom
## Multiple R-squared:  0.1897, Adjusted R-squared:  0.173
## F-statistic: 11.36 on 2 and 97 DF,  p-value: 3.7e-05
```

Alternatively, we can obtain the F-statistic in the summary from the anova test comparing the full model with the empty model:

```r
anova(reg.0, reg)
```

```
## Analysis of Variance Table
##
## Model 1: Y ~ 1
## Model 2: Y ~ X
##   Res.Df    RSS Df Sum of Sq      F  Pr(>F)
## 1     99 2154.2
## 2     97 1745.5  2    408.75 11.357 3.7e-05 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Working with categorical variables

```r
library(ISLR)
```

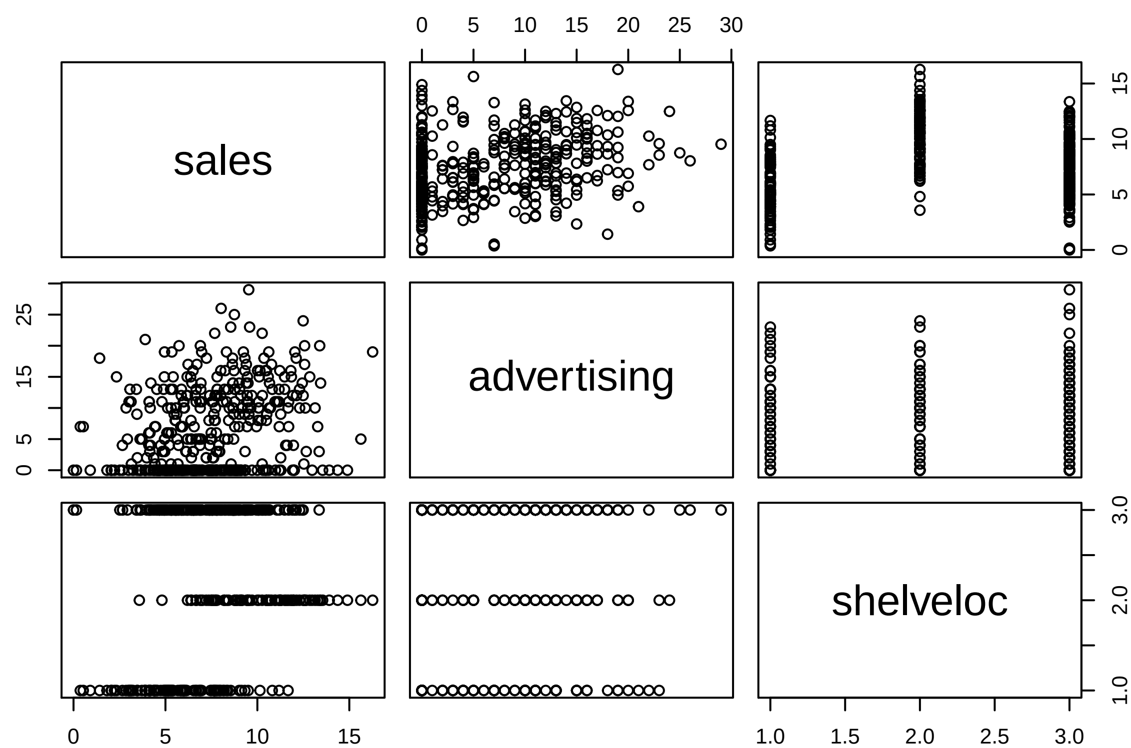
```
## Warning: package 'ISLR' was built under R version 3.6.3
```

```r
data(Carseats)
?Carseats
```

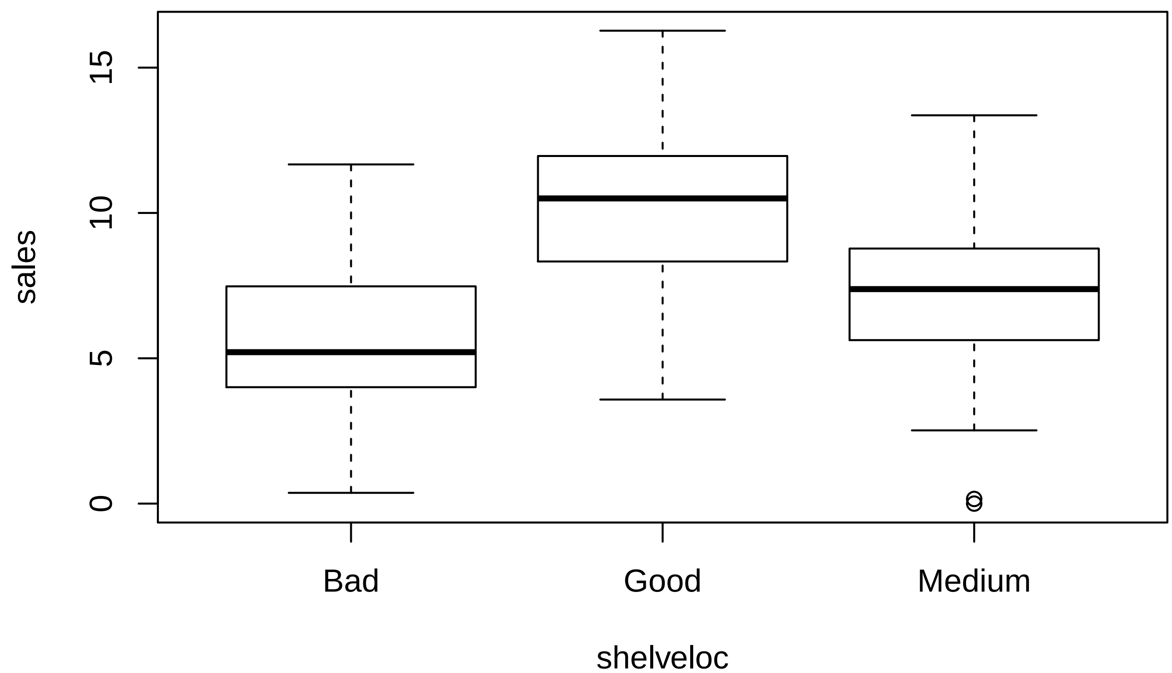
```
## starting httpd help server ... done
```

```r
# A factor with levels Bad, Good and Medium indicating the quality of the shelving location for the car
shelveloc=Carseats$ShelveLoc
# numerical predictor
advertising=Carseats$Advertising
# output variable
sales=Carseats$Sales
```

```r
pairs(sales ~ advertising + shelveloc)
```



```r
boxplot(sales ~shelveloc)
```



Notice there's no visible correlation between shelveloc and advertising (knowing something about the value of shelveloc doesn't tell me anything about the value of advertising).

Let's look at how R *lm* treats categorical variables.

```r
fit <- lm(sales ~ advertising + shelveloc)
summary(fit)
```

```
##
## Call:
## lm(formula = sales ~ advertising + shelveloc)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.6480 -1.6198 -0.0476  1.5308  6.4098
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      4.89662    0.25207  19.426  < 2e-16 ***
## advertising      0.10071    0.01692   5.951 5.88e-09 ***
## shelvelocGood    4.57686    0.33479  13.671  < 2e-16 ***
## shelvelocMedium  1.75142    0.27475   6.375 5.11e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.244 on 396 degrees of freedom
```

```
## Multiple R-squared:  0.3733, Adjusted R-squared:  0.3685
## F-statistic: 78.62 on 3 and 396 DF,  p-value: < 2.2e-16
```

R encodes automatically the categorical variable in 2 dummy variables. Let's do the same by hand:

```r
sl1 <- (shelveloc == "Good")*1
sl2 <- (shelveloc == "Medium")*1
fit.manual <- lm(sales ~ advertising + sl1 + sl2)
summary(fit.manual)
```

```
##
## Call:
## lm(formula = sales ~ advertising + sl1 + sl2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.6480 -1.6198 -0.0476  1.5308  6.4098
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.89662    0.25207  19.426  < 2e-16 ***
## advertising  0.10071    0.01692   5.951 5.88e-09 ***
## sl1          4.57686    0.33479  13.671  < 2e-16 ***
## sl2          1.75142    0.27475   6.375 5.11e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.244 on 396 degrees of freedom
## Multiple R-squared:  0.3733, Adjusted R-squared:  0.3685
## F-statistic: 78.62 on 3 and 396 DF,  p-value: < 2.2e-16
```

As expected, we get the same estimate. Now let's try encoding only bad and good.

```r
sl3 <- (shelveloc == "Bad")*1
fit.manual.2 <- lm(sales ~ advertising + sl1 + sl3)
summary(fit.manual.2)
```

```
##
## Call:
## lm(formula = sales ~ advertising + sl1 + sl3)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.6480 -1.6198 -0.0476  1.5308  6.4098
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.64805    0.18773  35.413  < 2e-16 ***
## advertising  0.10071    0.01692   5.951 5.88e-09 ***
## sl1          2.82543    0.28712   9.841  < 2e-16 ***
## sl3         -1.75142    0.27475  -6.375 5.11e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.244 on 396 degrees of freedom
## Multiple R-squared:  0.3733, Adjusted R-squared:  0.3685
```

```
## F-statistic: 78.62 on 3 and 396 DF,  p-value: < 2.2e-16
```

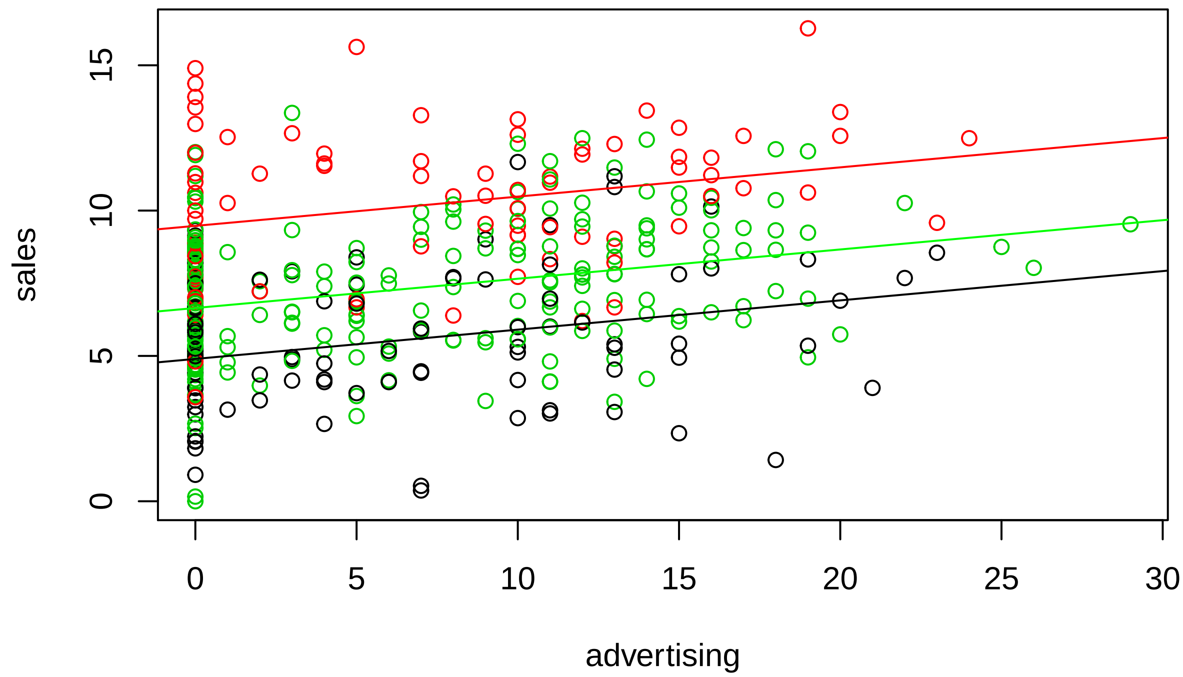
The estimate have changed but the t-test and F-test results remain the same. Now let's try giving the model all the variables.

```r
fit.manual.3 <- lm(sales ~ -1 + advertising + sl1 + sl2 + sl3)
summary(fit.manual.3)
```

```
##
## Call:
## lm(formula = sales ~ -1 + advertising + sl1 + sl2 + sl3)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.6480 -1.6198 -0.0476  1.5308  6.4098
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## advertising  0.10071    0.01692   5.951 5.88e-09 ***
## sl1          9.47348    0.27338  34.653  < 2e-16 ***
## sl2          6.64805    0.18773  35.413  < 2e-16 ***
## sl3          4.89662    0.25207  19.426  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.244 on 396 degrees of freedom
## Multiple R-squared:  0.9223, Adjusted R-squared:  0.9215
## F-statistic:  1175 on 4 and 396 DF,  p-value: < 2.2e-16
```

Note that we had to remove the intercept from the model. Let's visualize the different models:

```r
plot(advertising, sales, col=shelveloc)
beta = fit.manual.3$coefficients
legend(1,30,unique(shelveloc),col=1:3,pch=1)

abline(a=beta[2], b=beta[1], col="red")
abline(a=beta[3], b=beta[1], col="green")
abline(a=beta[4], b=beta[1], col="black")
```



Let's now answer the following question: is distinguishing between all three categories significantly better than distinguishing only between "bad" (level bad) and "not bad" (level medium or good), when accounting for advertising as well?

The answer is already right here.

```r
# any model summary will do
summary(fit)
```

```
##
## Call:
## lm(formula = sales ~ advertising + shelveloc)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.6480 -1.6198 -0.0476  1.5308  6.4098
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      4.89662    0.25207  19.426  < 2e-16 ***
## advertising      0.10071    0.01692   5.951 5.88e-09 ***
## shelvelocGood    4.57686    0.33479  13.671  < 2e-16 ***
## shelvelocMedium  1.75142    0.27475   6.375 5.11e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.244 on 396 degrees of freedom
## Multiple R-squared:  0.3733, Adjusted R-squared:  0.3685
```

```
## F-statistic: 78.62 on 3 and 396 DF,  p-value: < 2.2e-16
```

The p-value of the *shelvelocGood* or the *shelvelocMedium* variable can be interpreted as its *significance* wrt to *sales*, given that all the other variables are already in the model. If the p-value was larger than 0.05 it would mean that the distinction between Good and Medium is not significantly improving the model (if the model assumptions are met, relatively to this dataset). Having the opposite result we can answer the above question with a *yes*.

WIth partial F-test we should arrive at the same conclusion.

```r
# only accounting for "bad" and "not bad" here:
fit.2 <- lm(sales ~ advertising + sl3)
anova(fit.2, fit.manual.3)
```

```
## Analysis of Variance Table
##
## Model 1: sales ~ advertising + sl3
## Model 2: sales ~ -1 + advertising + sl1 + sl2 + sl3
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1    397 2482.1
## 2    396 1994.4  1    487.71 96.837 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As expected, the partial F-test confirms out hypothesis.

Working with time data, where LR fails

The dataset airline contains the monthly number of flight passengers in the USA in the years ranging from January 1949 to December 1960.

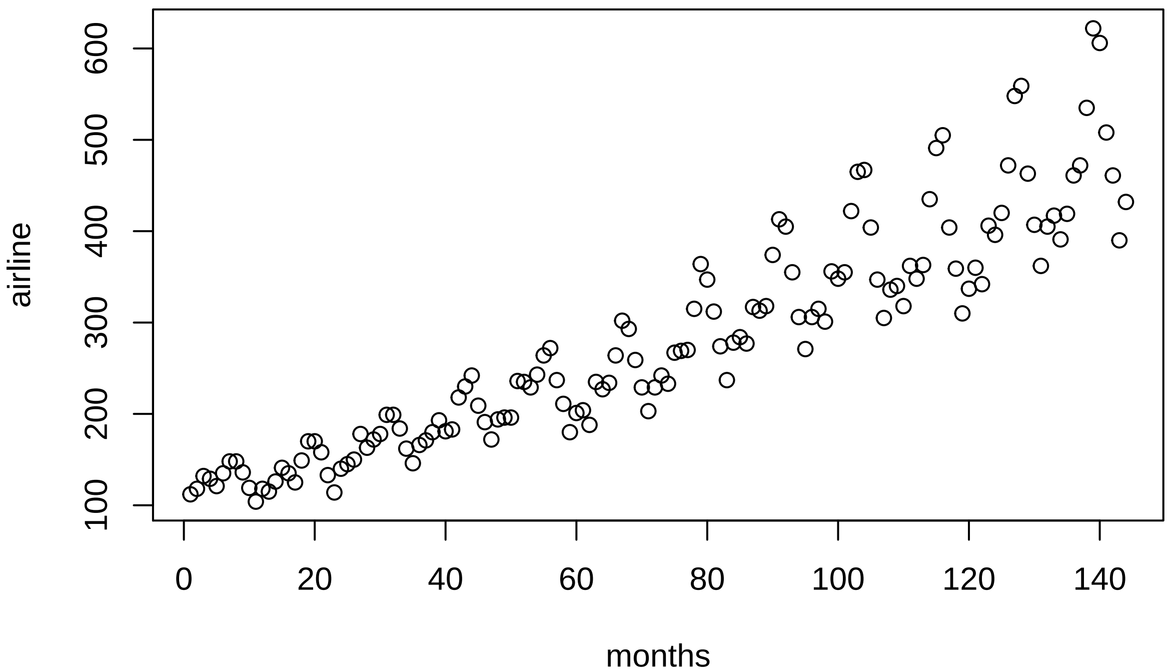
```r
airline <- scan("http://stat.ethz.ch/Teaching/Datasets/airline.dat")
head(airline)
```

```
## [1] 112 118 132 129 121 135
```

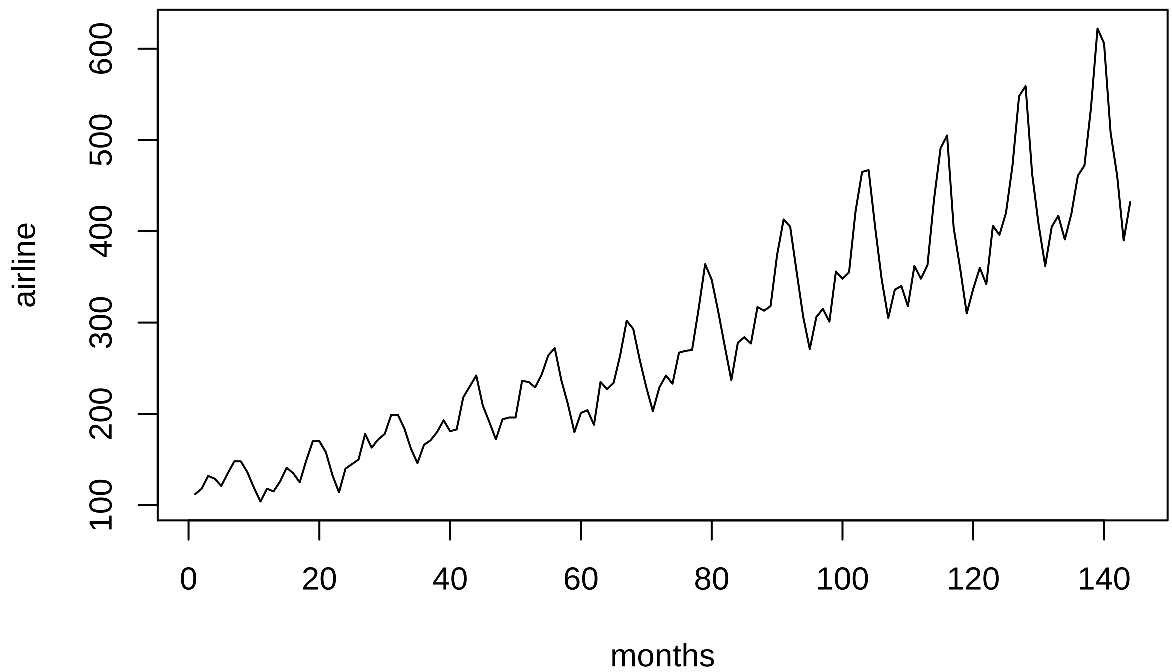
```r
length(airline)
```

```
## [1] 144
```

```r
months <- 1:144
plot(months, airline)
```

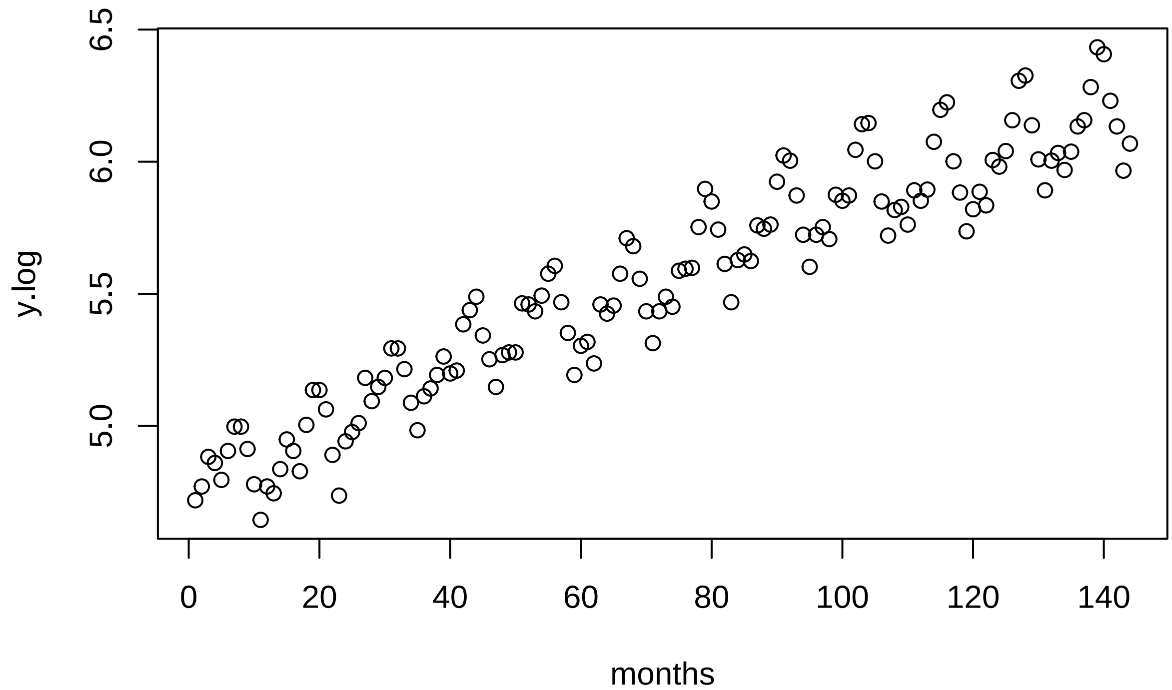



```
plot(months, airline, type="l")
```

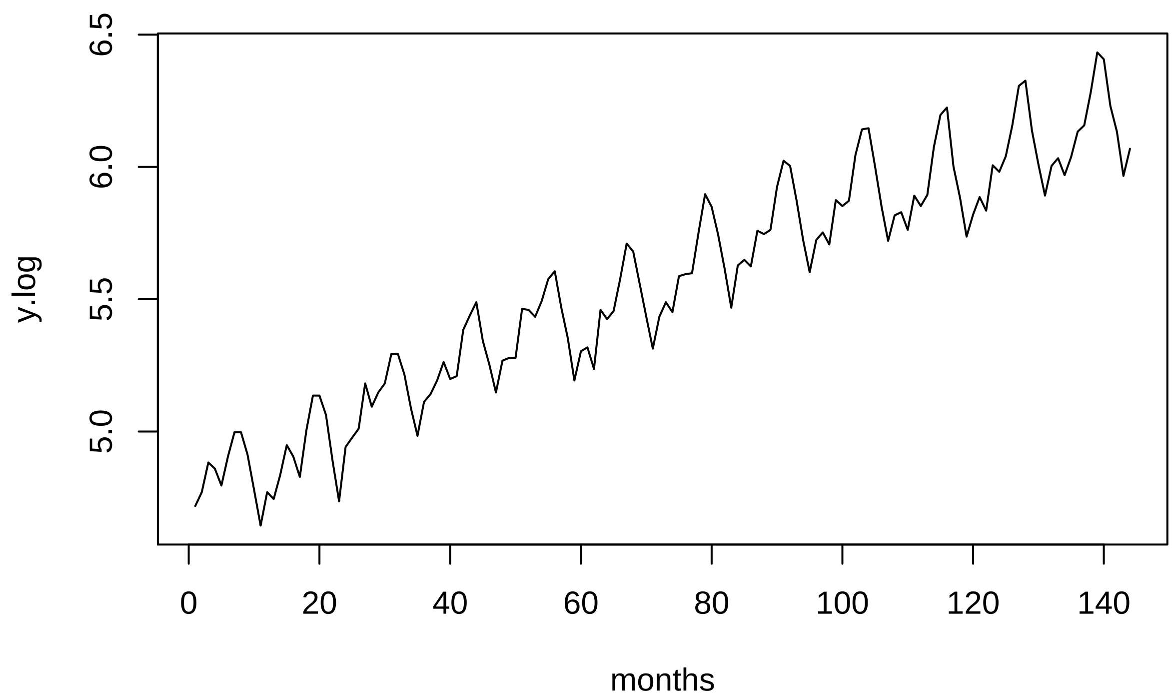


Resembles an exponential trend. Let's work with the logarithm of the output.

```
y.log <- log(airline)
plot(months, y.log)
```




```
plot(months, y.log, type="l")
```



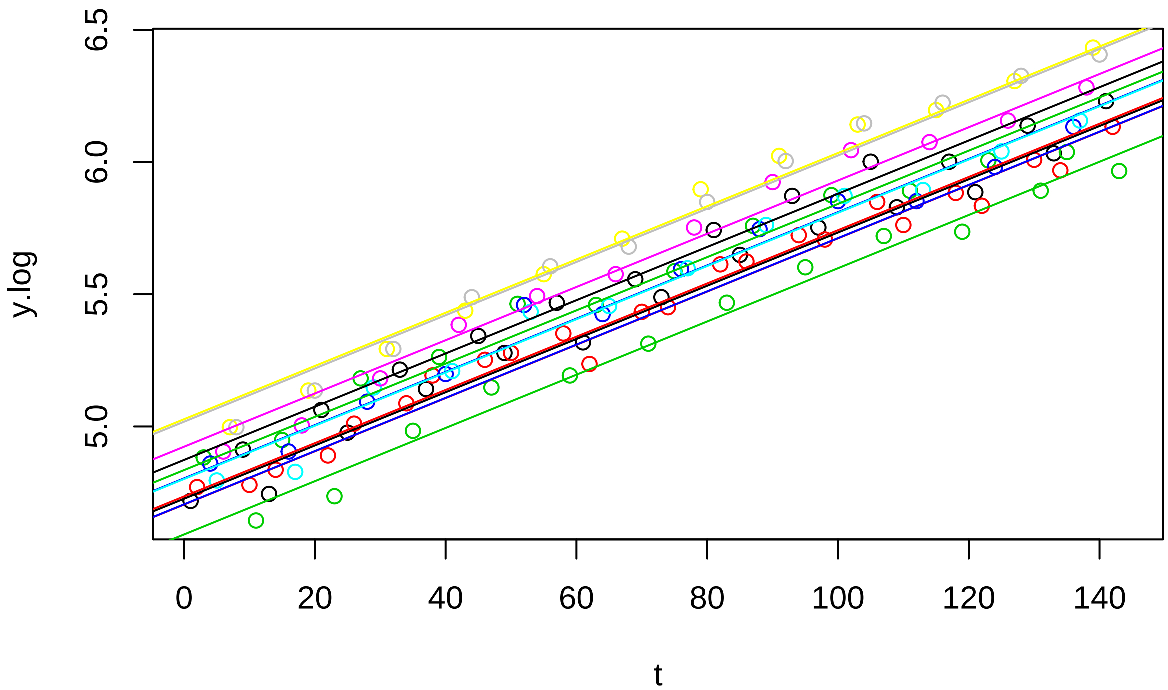
Let's now define a linear model depending on time and on the particular month of the year, regressing the logarithm transformed output.

```
x1<-rep(c(1,rep(0,11)),12)
x2<-rep(c(rep(0,1),1,rep(0,10)),12)
x3<-rep(c(rep(0,2),1,rep(0,9)),12)
x4<-rep(c(rep(0,3),1,rep(0,8)),12)
x5<-rep(c(rep(0,4),1,rep(0,7)),12)
x6<-rep(c(rep(0,5),1,rep(0,6)),12)
x7<-rep(c(rep(0,6),1,rep(0,5)),12)
x8<-rep(c(rep(0,7),1,rep(0,4)),12)
x9<-rep(c(rep(0,8),1,rep(0,3)),12)
x10<-rep(c(rep(0,9),1,rep(0,2)),12)
x11<-rep(c(rep(0,10),1,rep(0,1)),12)
x12<-rep(c(rep(0,11),1),12)
t <- months

# notice that we remove the intercept
model<- lm(y.log~-1+t+x1+x2+x3+x4+x5+x6+x7+x8+x9+x10+x11+x12)
summary(model)
```

```
##
## Call:
## lm(formula = y.log ~ -1 + t + x1 + x2 + x3 + x4 + x5 + x6 + x7 +
##     x8 + x9 + x10 + x11 + x12)
##
## Residuals:
```

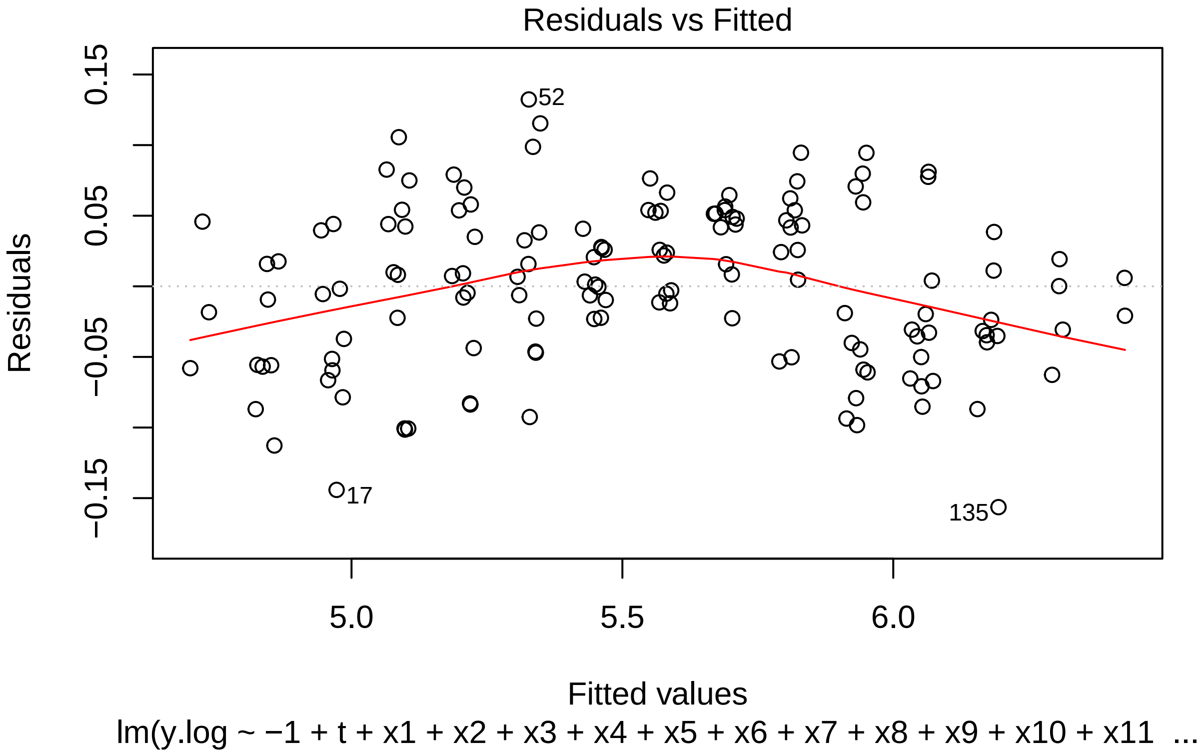
```
##       Min        1Q    Median        3Q       Max
## -0.156370 -0.041016  0.003677  0.044069  0.132324
##
## Coefficients:
##      Estimate Std. Error t value Pr(>|t|)
## t   0.0100688  0.0001193    84.4   <2e-16 ***
## x1  4.7267804  0.0188935   250.2   <2e-16 ***
## x2  4.7047255  0.0189443   248.3   <2e-16 ***
## x3  4.8349527  0.0189957   254.5   <2e-16 ***
## x4  4.8036838  0.0190477   252.2   <2e-16 ***
## x5  4.8013112  0.0191003   251.4   <2e-16 ***
## x6  4.9234574  0.0191535   257.1   <2e-16 ***
## x7  5.0273997  0.0192073   261.7   <2e-16 ***
## x8  5.0181049  0.0192617   260.5   <2e-16 ***
## x9  4.8734703  0.0193167   252.3   <2e-16 ***
## x10 4.7353120  0.0193722   244.4   <2e-16 ***
## x11 4.5915943  0.0194283   236.3   <2e-16 ***
## x12 4.7054593  0.0194850   241.5   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0593 on 131 degrees of freedom
## Multiple R-squared:  0.9999, Adjusted R-squared:  0.9999
## F-statistic: 9.734e+04 on 13 and 131 DF,  p-value: < 2.2e-16
```

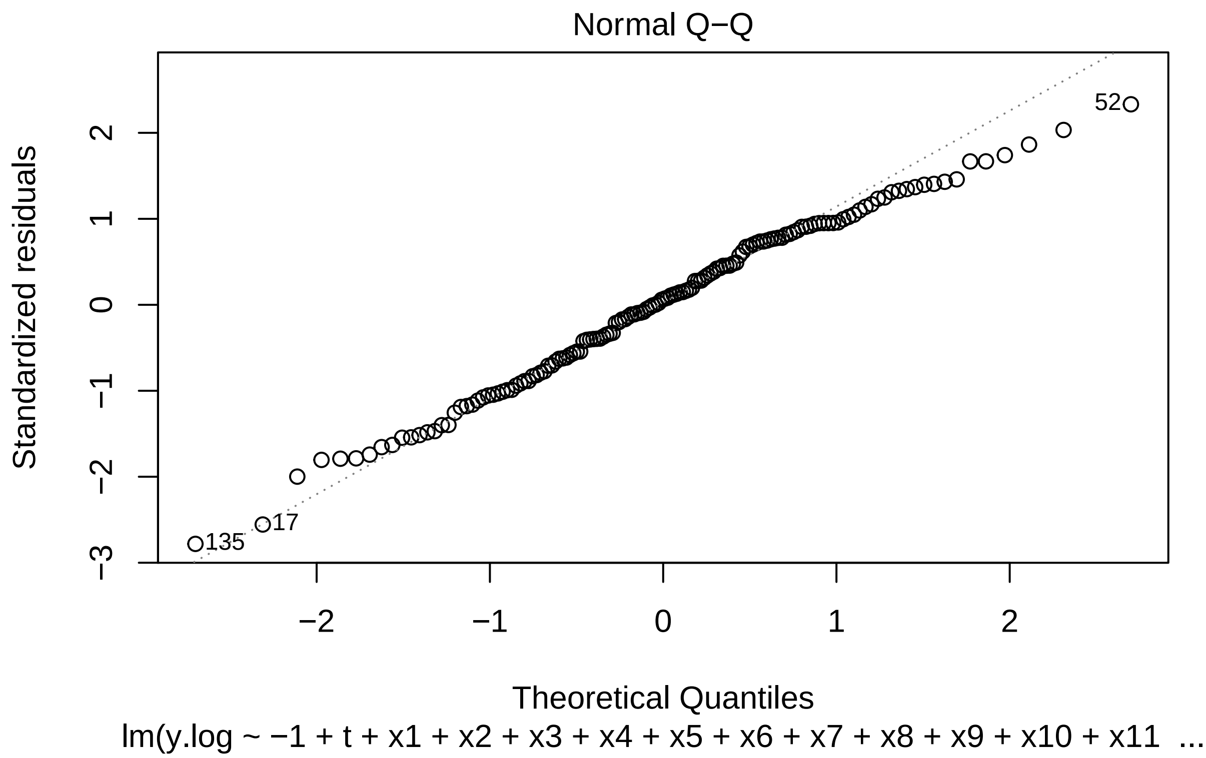
```
#let's visualize our model
plot(t, y.log, col=(t%%12))
beta = model$coefficients
for(i in 2:13){
  abline(a=beta[i], b=beta[1], col=i-1)
}
```



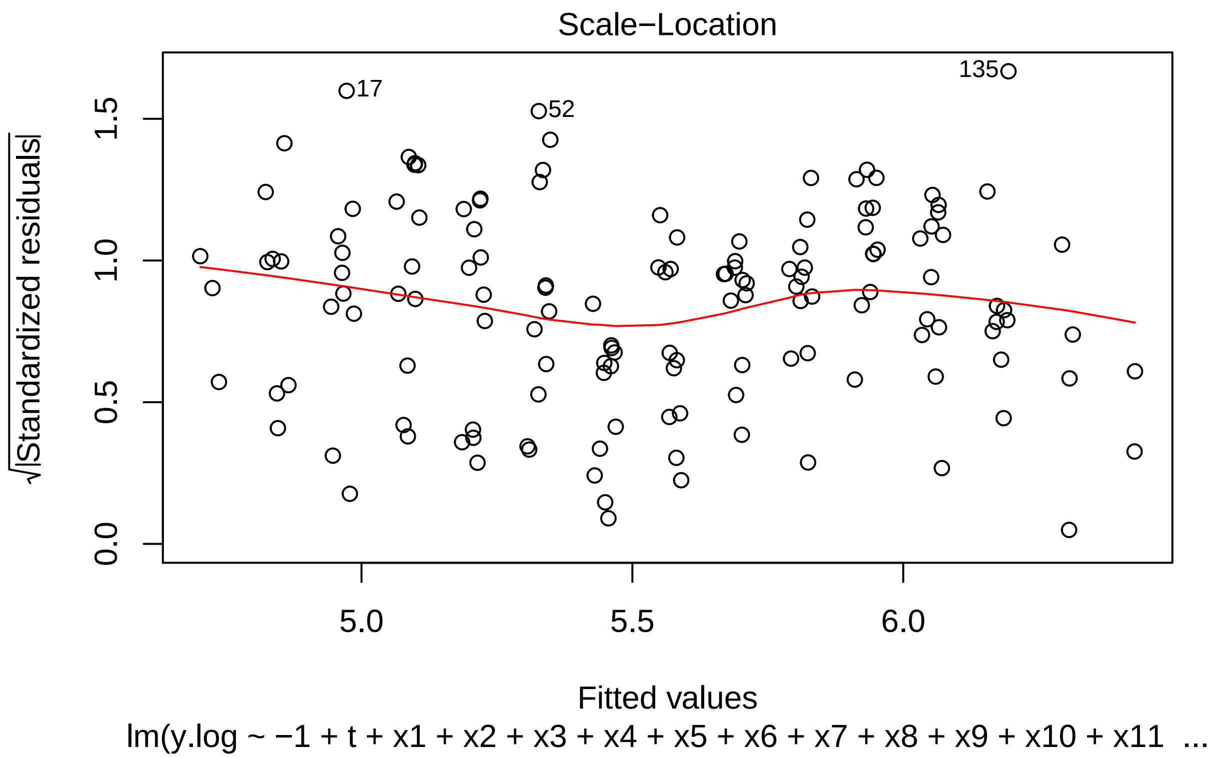
So our implicit model assumption is that the log of the output grows linearly in time from month to month, with some white noise added. Do these assumptions truly hold?

```
plot(model)
```

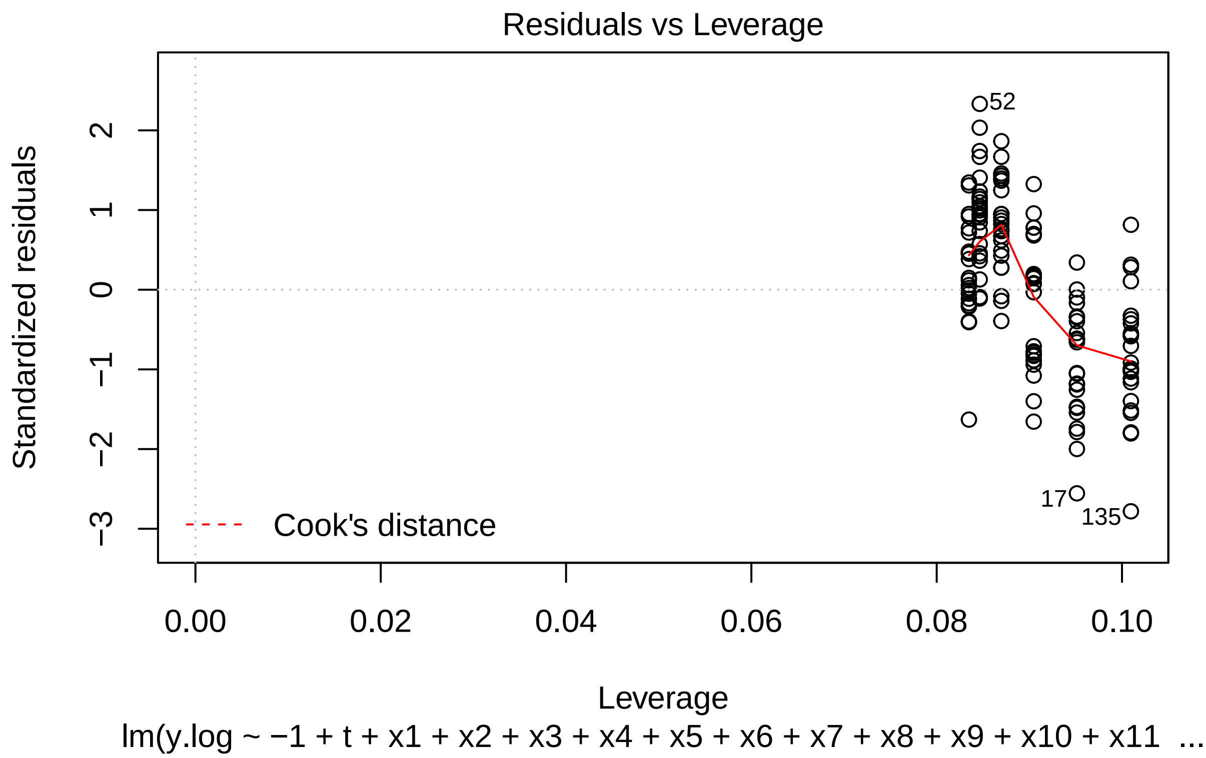

Residuals vs Fitted
Residuals
Fitted values
lm(y.log ~ −1 + t + x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9 + x10 + x11 ...
0.15
0.05
−0.05
−0.15
5.0
5.5
6.0
52
17
135



Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(y.log ~ −1 + t + x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9 + x10 + x11 ...
135
17
52
−3
−2
−1
0
1
2
−2
−1
0
1
2

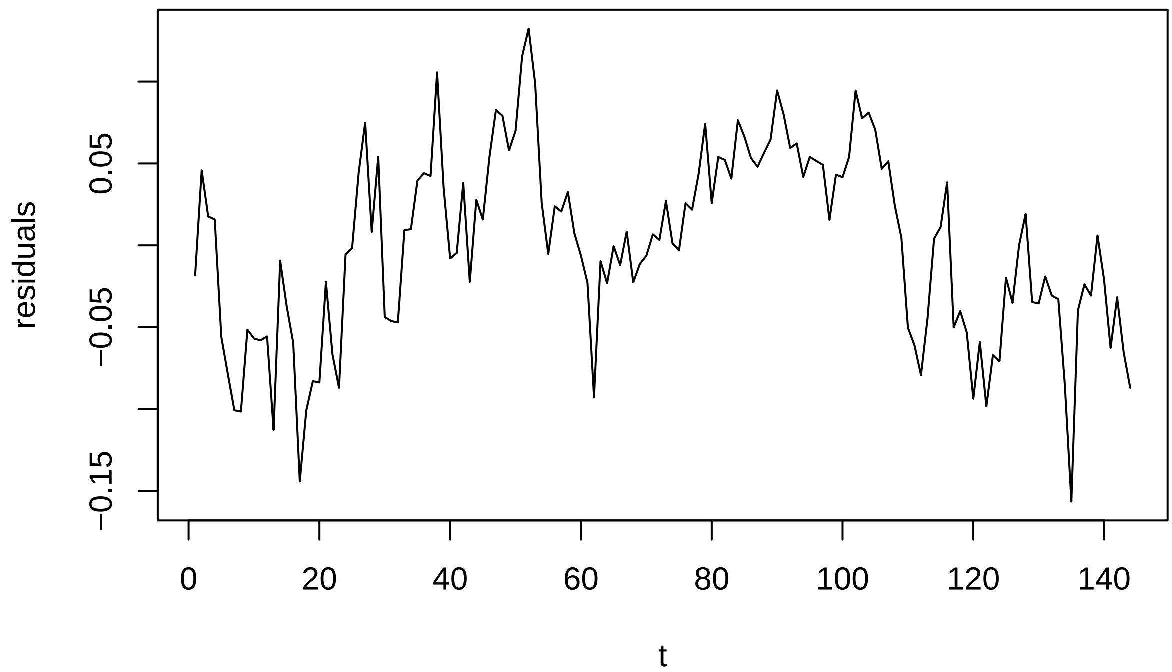


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
17
52
135
5.0
5.5
6.0
Fitted values
lm(y.log ~ −1 + t + x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9 + x10 + x11 ...

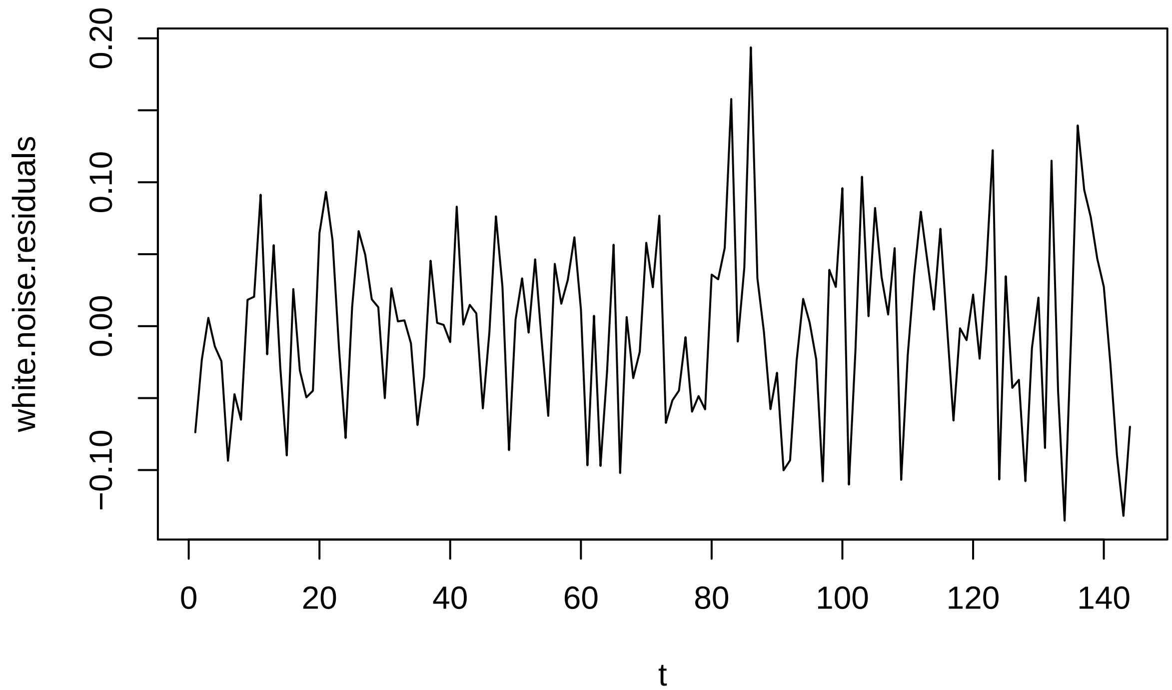


Checking for correlation in the residuals. We take a look at artificial data generated according to the model which we want to check (i.i.d. normally distributed residuals).

```
residuals <- model$residuals
plot(t,residuals, type="l")
```




```
s <- summary(model)
sigma <- s$sigma

white.noise.residuals <- rnorm(n=144, mean=0, sd=sigma)
plot(t,white.noise.residuals, type="l")
```



Since there seems to be serial correlation (violation of model assumptions), the standard errors and p-values are not valid.

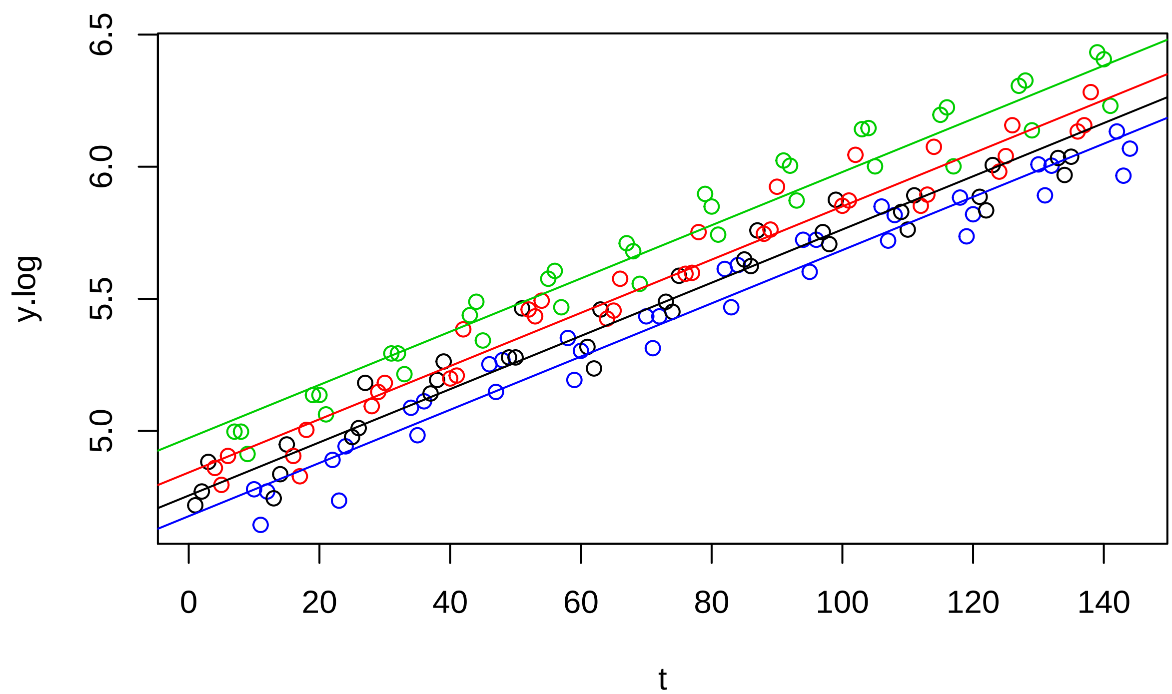
Let's try encoding only seasonal changes.

```
s1<-rep(c(rep(1,3),rep(0,9)),12)
s2<-rep(c(rep(0,3),rep(1,3),rep(0,6)),12)
s3<-rep(c(rep(0,6),rep(1,3),rep(0,3)),12)
s4<-rep(c(rep(0,9),rep(1,3)),12)
t <- months

seasons <- rep(c(rep(1,3),rep(2,3),rep(3,3),rep(4,3)),12)
# notice that we remove the intercept
model.seas<- lm(y.log~-1+t+s1+s2+s3+s4)
summary(model.seas)
```

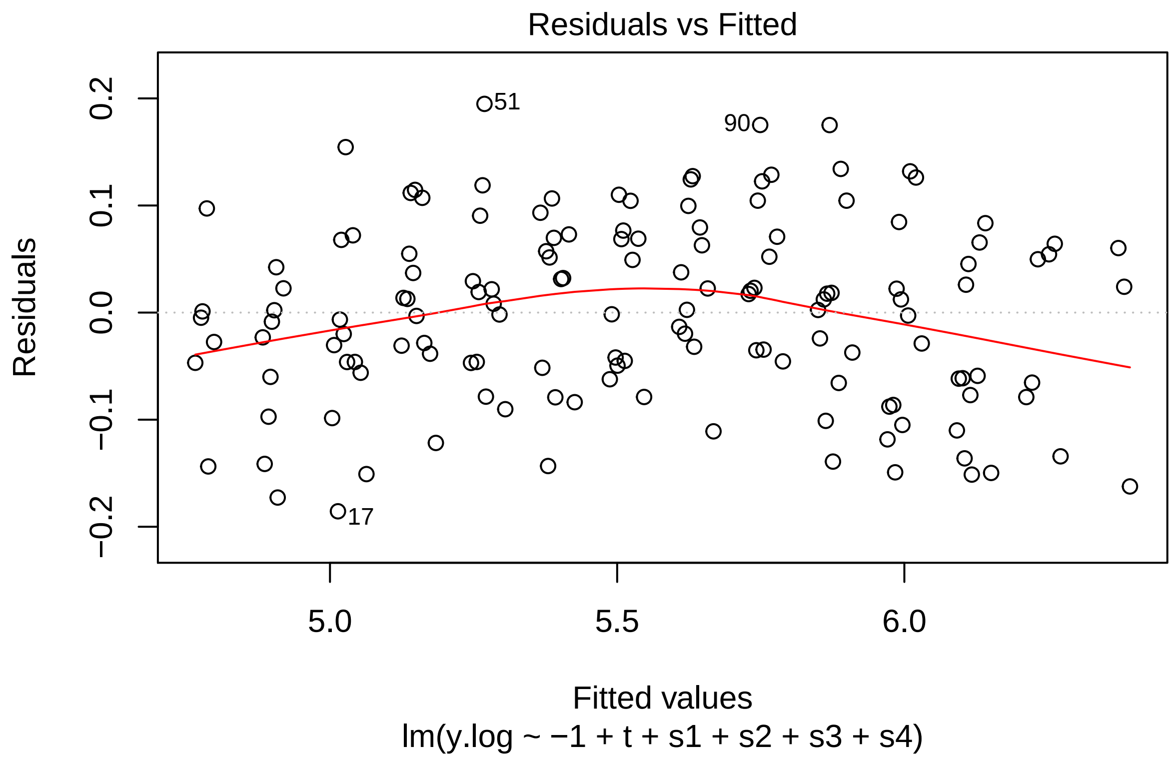
```
##
## Call:
## lm(formula = y.log ~ -1 + t + s1 + s2 + s3 + s4)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.185558 -0.059324  0.002313  0.064539  0.194873
##
## Coefficients:
##     Estimate Std. Error t value Pr(>|t|)
## t  0.0100709  0.0001716    58.7   <2e-16 ***
```

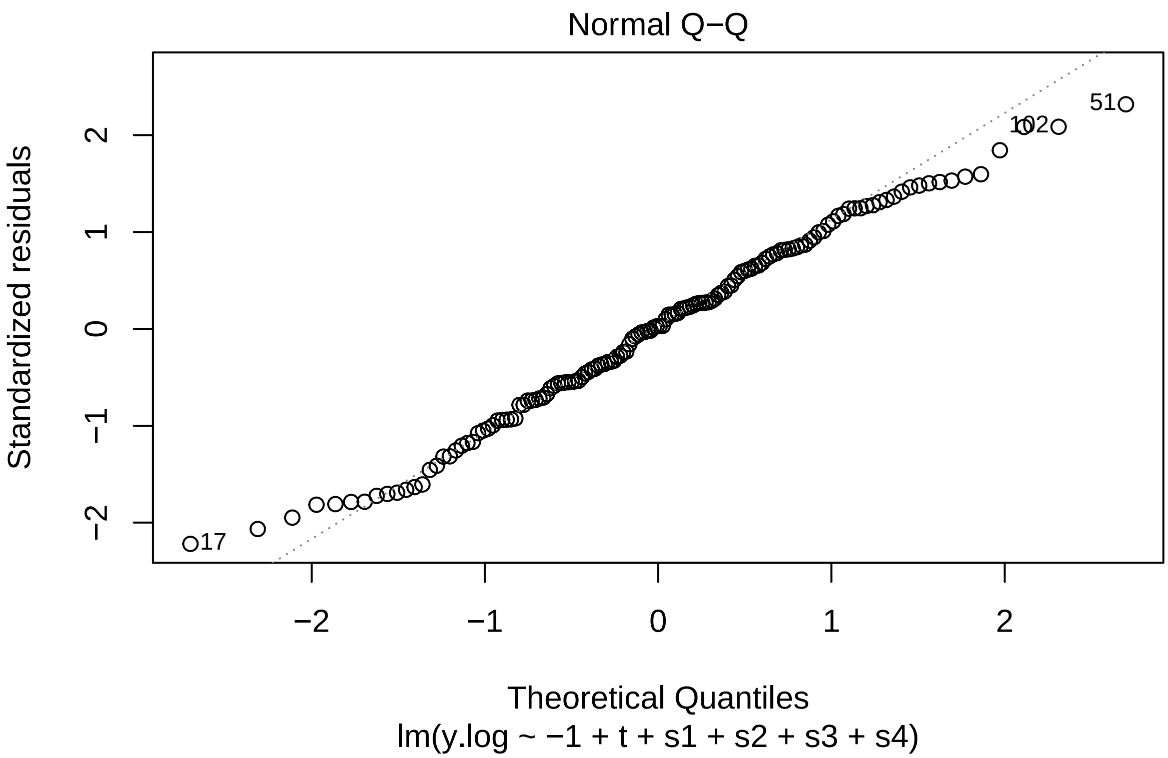
```
## s1 4.7553404  0.0183893   258.6   <2e-16 ***
## s2 4.8426652  0.0187200   258.7   <2e-16 ***
## s3 4.9728330  0.0190589   260.9   <2e-16 ***
## s4 4.6772901  0.0194055   241.0   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08529 on 139 degrees of freedom
## Multiple R-squared:  0.9998, Adjusted R-squared:  0.9998
## F-statistic: 1.223e+05 on 5 and 139 DF,  p-value: < 2.2e-16
```

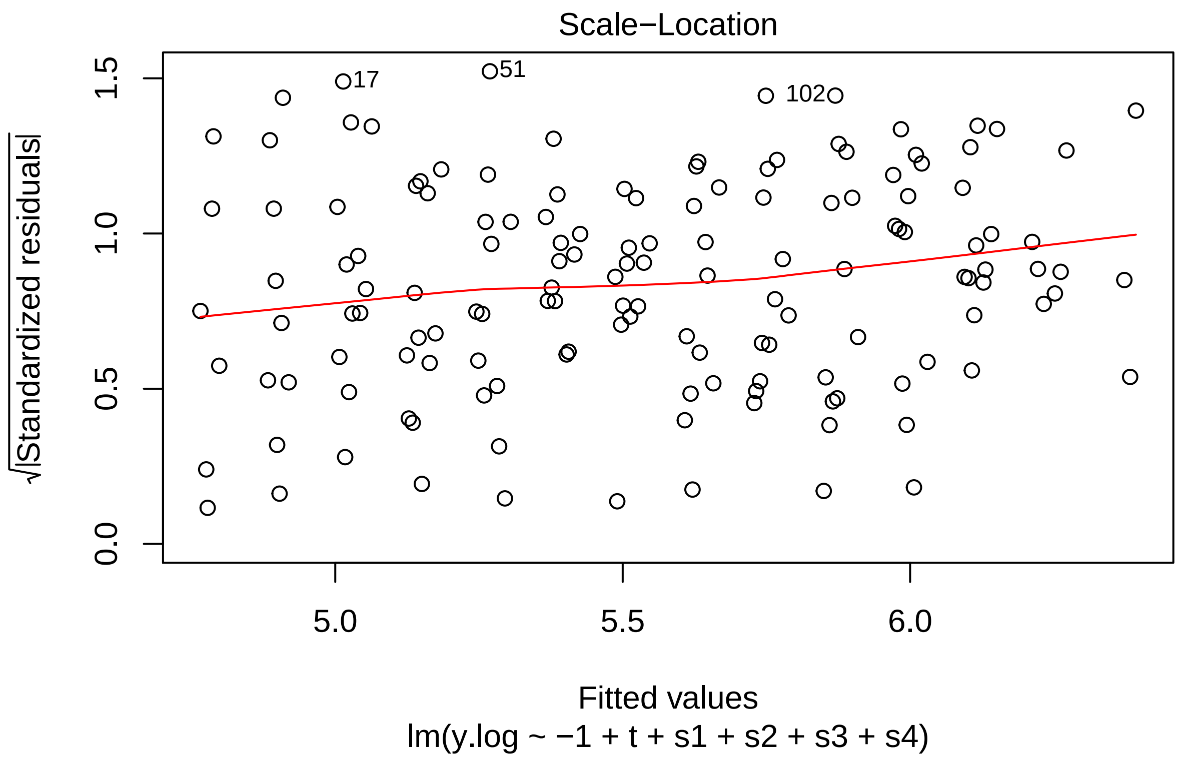
```r
#let's visualize our model
plot(t, y.log, col=seasons)
beta = model.seas$coefficients
for(i in 2:5){
  abline(a=beta[i], b=beta[1], col=i-1)
}
```

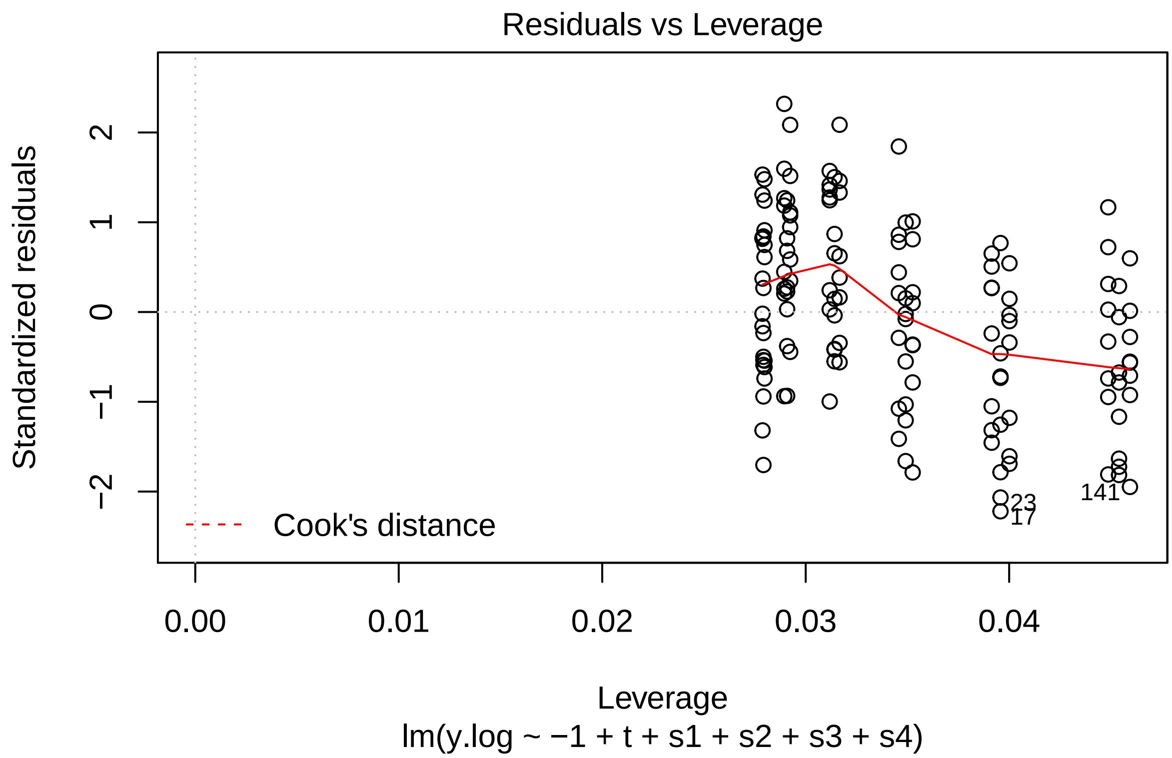



```r
plot(model.seas)
```


Residuals vs Fitted
Residuals
0.2
0.1
0.0
−0.1
−0.2
51
90
17
5.0
5.5
6.0
Fitted values
lm(y.log ~ −1 + t + s1 + s2 + s3 + s4)


Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(y.log ~ −1 + t + s1 + s2 + s3 + s4)
51
102
17
−2
−1
0
1
2


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
5.0
5.5
6.0
Fitted values
lm(y.log ~ −1 + t + s1 + s2 + s3 + s4)
17
51
102



Partial F-test to check whether in one there is at least one variable significant.

```
anova(model.seas, model)
```

```
## Analysis of Variance Table
##
## Model 1: y.log ~ -1 + t + s1 + s2 + s3 + s4
## Model 2: y.log ~ -1 + t + x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9 +
##     x10 + x11 + x12
##   Res.Df     RSS Df Sum of Sq      F    Pr(>F)
## 1    139 1.01123
## 2    131 0.46072  8   0.55051 19.567 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Seems like the full model is better…