Chapter 9 Statistical Modeling | Chapter 10 Regression

Qianqian Shan

May 30, 2017

Appropriate statistical mothods

Explanatory variables	Method
All continuous	Regression
All categorical	ANOVA
Both continuous and categorical	ANCOVA(covariance)

Response Variables Type	Method
Continous	Normal regresssion, ANOVA, ANCOVA
proportion	Logistic regression
Count	Log-linear models
Binary	Binary logistic regression
Time at death	Survival analysis

A model should be as simple as simple, but no simpler.

Types of statistical models:

Model	Fit	Degree of Freedom	Explanatory Power	Interpretation
Saturated model	Perfect	None	None	One parameter for every data point
Maximal model		$n-p-1$	Depends	Contatins all p factors, interactions, covariates etc.
Minimal adequate model	less than maximal but not significant	$n-p'-1$	$r^2 = SSR/SSY$	Simplified model wiht $1 \leq p' \leq p$ parameters
Null model	None	$n-1$	None	Just one parameter, i.e., the overall mean $\overline{y}$

Formulae in R :

Model	Formula	Comments
Null	$y \sim 1$	1 for intercept
Regression	$y \sim x$	x is continuous
Regression w/o intercept	$y \sim x - 1$	
One-way ANOVA	$y \sim sex$	Categorical variable
Two-way ANOVA	$y \sim sex + genotype$	Two categorical variables
Factorial ANOVA	$y \sim N * P * K$	Factors with all their interactions
Three-way ANOVA	$y \sim N * P * K - N : P : K$	Same as above except that no three-way interaction

Model	Formula	Comments
Analysis of Covariance	$y \sim x + sex$	sex categorical, x continuous, common slope for x with two intercepts for sex
Analysis of Covariance	$y \sim x * sex$	Two slopes and two intercepts
Nested ANOVA	$y \sim a/b/c$	Factor c nested within factor b within factor a
Split-plot ANOVA	$y \sim a * b * c + Error(a/b/c)$	Factorial experiment but with three plot sizes and three different error variances
Multiple Regression	$y \sim x * z$	Fit two continuous variables together with interactions
Multiple Regression	$y \sim x + I(x\hat{}2) + z + I(z\hat{}2)$	Quadratic term for each
Multiple Regression	$y \sim poly(x, 2) + z$	Quadratic polynomial for x and linear for z
Multiple Regression	$y \sim (x + z + 2)\hat{}2$	Fit three variables with interactions up to two-way
Non-parametric Model	$y \sim s(x) + s(z)$	A function of smoothed x and z in a generalized additive model
Transformed Response and Variables	$log(y) \sim I(1/x) + sqrt(z)$	All variables transformed

Note: we need to use I() if want to use a transformed variable in the formula.

Model formulae in R

```
# create formula objects using "collapse" and "paste"
xnames <- paste("x", 1:25, sep = "")
model.formula <- as.formula(paste("y ~", paste(xnames, collapse = "+")))
model.formula
```

```
## y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9 + x10 + x11 +
##     x12 + x13 + x14 + x15 + x16 + x17 + x18 + x19 + x20 + x21 +
##     x22 + x23 + x24 + x25
```

`update` function in model simplication

With `model` as the previously speicified model, the following statement removes the interaction term `A:B`: $model2 < -update(model, \ . - A : B)$.

Box-Cox transformations

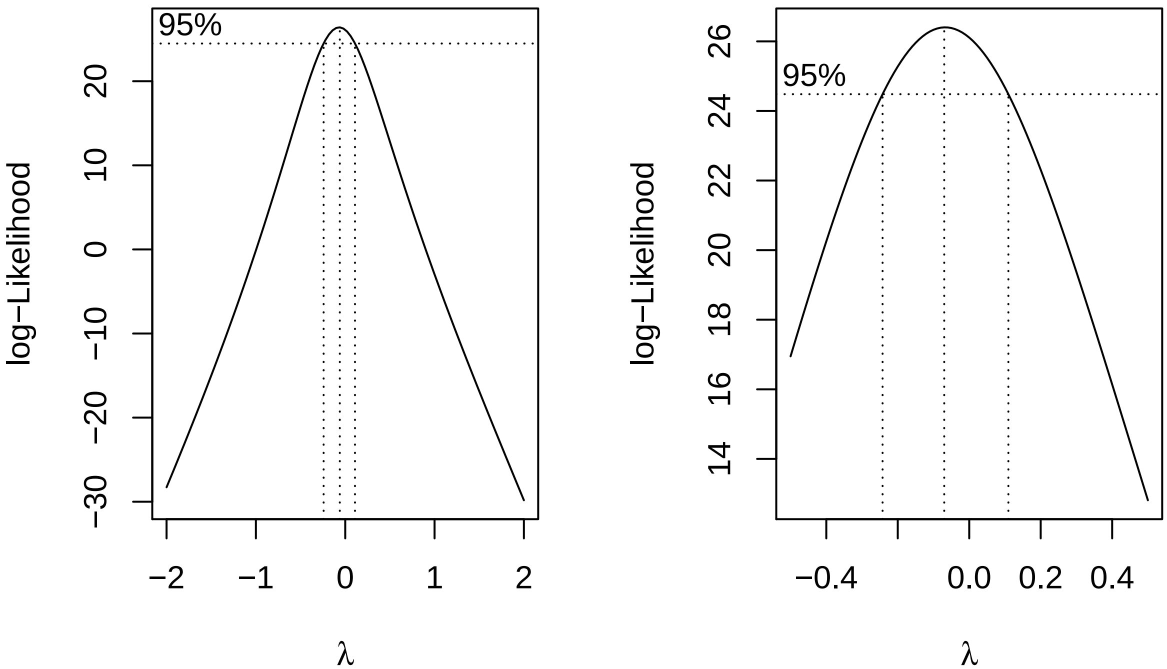
A simple empirical solution for optimal transformation of the response variables.

Idea: find the power transformation λ, that maximizes the likelihood when a speicified set of explanatory variables is fitted to $\frac{y^\lambda - 1}{\lambda}$, while the transformation is $log(y)$ when $\lambda = 0$.

```
data <- read.delim("timber.txt")
attach(data)
names(data)
```

```
## [1] "volume" "girth"  "height"
```

```r
library(MASS)

# boxcox : Computes and optionally plots profile log-likelihoods for the parameter of the Box-Cox power
par(mfrow = c(1, 2))
boxcox(volume ~ log(girth) + log(height))

# zoom the area with maximal likelihood
boxcox(volume ~ log(girth) + log(height), lambda = seq(-0.5, 0.5, 0.01))
```




```r
detach(data)
par(mfrow = c(1, 1))
```

Model checking

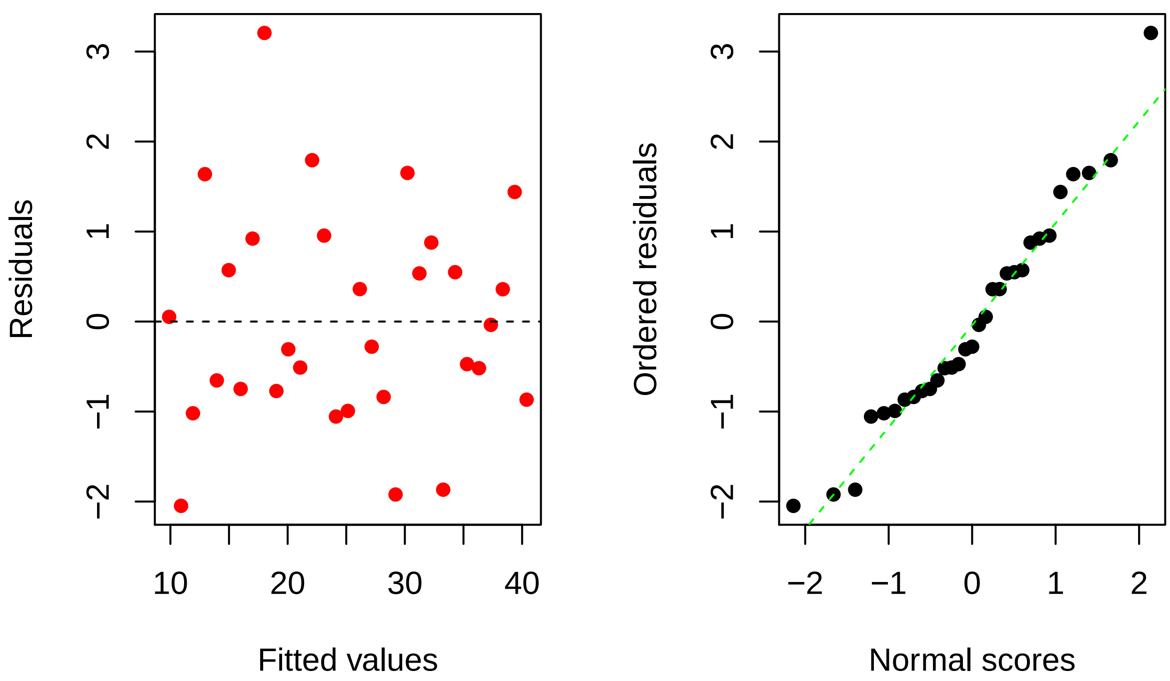
1. Residuals against -

- fitted values for heteroscedasticity (standardized residuals against fitted values)
- explanatory variables for evidence of curvature
- the sequence of data collection for temporal correlation
- standard normal deviates for non normality of errors.

2. Influential data points
3. Overdispertion
4. Depends

```r
# model check function:
# plot residuals vs fitted values and plot qqplot again normal data

mcheck <- function (obj,...){
      rs <- obj$resid
```

```
      fv <- obj$fitted
      par(mfrow = c(1,2))
      plot(fv, rs, xlab="Fitted values", ylab="Residuals", pch=16, col="red")
      abline(h=0, lty=2)
      qqnorm(rs, xlab = "Normal scores", ylab="Ordered residuals", main="", pch=16)
      qqline(rs, lty=2, col = "green")
      par(mfrow = c(1,1))
      invisible(NULL) }

x <- 0:30
e <- rnorm(31, 0, 1)
y <- 10 + x + e
mn <- lm(y ~ x)
mcheck(mn)
```

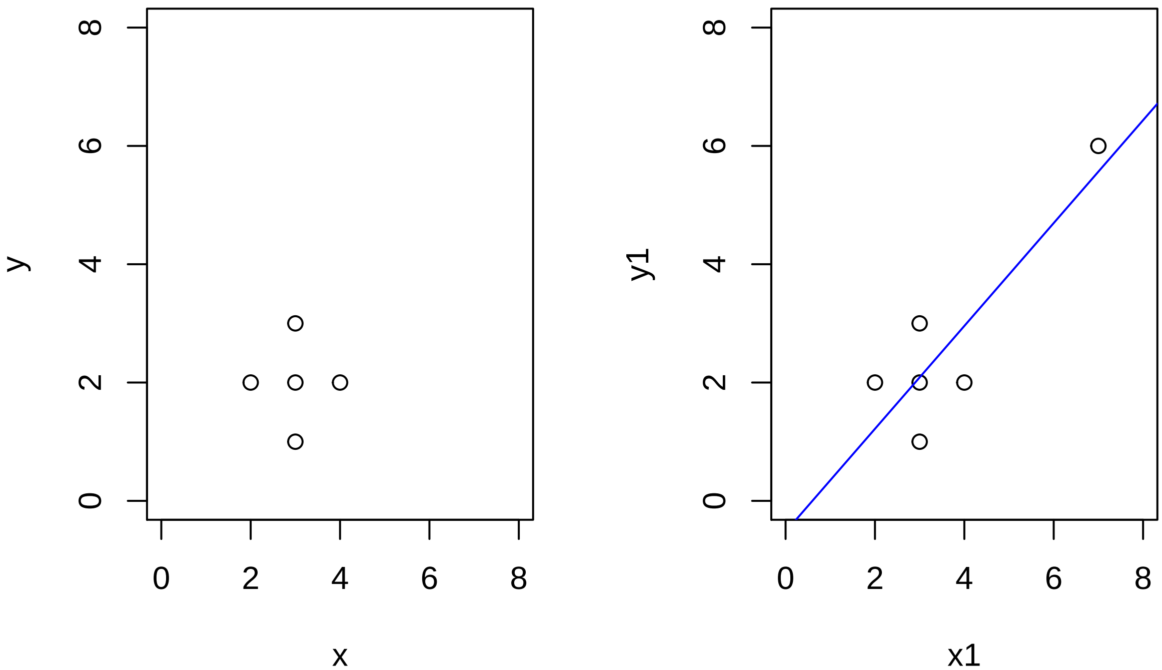



```
rm(x)
rm(y)
rm(e)
```

Influence

```
x <- c(2,3,3,3,4)
y <- c(2,3,2,1,2)
par(mfrow=c(1,2))
plot(x, y, xlim=c(0,8), ylim=c(0,8))

# add an outlier
x1 <- c(x, 7)
y1 <- c(y, 6)
```

```r
plot(x1, y1, xlim = c(0,8), ylim = c(0,8))
abline(lm(y1~x1), col = "blue")
```




```r
par(mfrow = c(1, 1))

# fit the regression
reg <- lm(y1~x1)
summary(reg)
```

```
##
## Call:
## lm(formula = y1 ~ x1)
##
## Residuals:
##        1        2        3        4        5        6
##  0.78261  0.91304 -0.08696 -1.08696 -0.95652  0.43478
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.5217     0.9876  -0.528   0.6253
## x1            0.8696     0.2469   3.522   0.0244 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9668 on 4 degrees of freedom
## Multiple R-squared:  0.7561, Adjusted R-squared:  0.6952
## F-statistic:  12.4 on 1 and 4 DF,  p-value: 0.02441
```

```r
# measure the influence of every point
influence.measures(reg)
```

```
## Influence measures of
##   lm(formula = y1 ~ x1) :
```

```
##
##   dfb.1_  dfb.x1   dffit cov.r  cook.d   hat inf
## 1  0.687 -0.5287  0.7326 1.529 0.26791 0.348
## 2  0.382 -0.2036  0.5290 1.155 0.13485 0.196
## 3 -0.031  0.0165 -0.0429 2.199 0.00122 0.196
## 4 -0.496  0.2645 -0.6871 0.815 0.19111 0.196
## 5 -0.105 -0.1052 -0.5156 1.066 0.12472 0.174
## 6 -3.023  4.1703  4.6251 4.679 7.62791 0.891   *
```

```
influence.measures(reg)$is.inf
```

```
##   dfb.1_ dfb.x1 dffit cov.r cook.d   hat
## 1  FALSE  FALSE FALSE FALSE  FALSE FALSE
## 2  FALSE  FALSE FALSE FALSE  FALSE FALSE
## 3  FALSE  FALSE FALSE FALSE  FALSE FALSE
## 4  FALSE  FALSE FALSE FALSE  FALSE FALSE
## 5  FALSE  FALSE FALSE FALSE  FALSE FALSE
## 6   TRUE   TRUE  TRUE  TRUE   TRUE FALSE
```

```
lm.influence(reg)
```

```
## $hat
##         1         2         3         4         5         6
## 0.3478261 0.1956522 0.1956522 0.1956522 0.1739130 0.8913043
##
## $coefficients
##   (Intercept)           x1
## 1  0.67826087 -0.130434783
## 2  0.37015276 -0.049353702
## 3 -0.03525264  0.004700353
## 4 -0.44065805  0.058754407
## 5 -0.10068650 -0.025171625
## 6 -2.52173913  0.869565217
##
## $sigma
##         1         2         3         4         5         6
## 0.9660918 0.9491580 1.1150082 0.8699177 0.9365858 0.8164966
##
## $wt.res
##           1           2           3           4           5           6
##  0.78260870  0.91304348 -0.08695652 -1.08695652 -0.95652174  0.43478261
```

```
# model withouth the outlier
summary.aov(lm(y1[-6]~x1[-6]))
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## x1[-6]       1      0  0.0000       0      1
## Residuals    3      2  0.6667
```

Summary of statistical models in R

Models in R	Description
lm	linear model with **normal** errors and **constant** variance; generally used for regression for continuous variables.

Models in R	Description
aov	fit analysis of variance with **normal** errors, **constant** variance and **identity** link; generally for categorical variables or ANCOVA with a mix of categorical and continuous variables.
glm	generalized linear models to data using categorical or continuous variables , by specifying one of a family of **error structure** and a particular **link function**.
gam	generalized additive models to data with a family of errosr structures in which the continuous varibles can be fitted as **arbitrary smoothed functions** using **non-parametric** smoothers rather than the specific parameter functions.
lme, lmer	fit linear mixed-effects models with specified mixtures of fixed effects and random effects, allow for the **specification of correlation structure among explanatory variables and autocorrelation of the response variable**.
nls	non-linear regression model via **least squares**.
nlme	non-linear mixed-effects model where parameters of the non-linear function are assumed to be random effects; allows for **specification of correlation structure among explanatory variables and autocorrelation of the response variable**.
loess	local regression model using **non-parametric** techniques to produce a smoothed model surface.
tree, rpart	fit a regression tree/classification tree using binary recursive partitioning.

Optional arguments in model-fitting functions

1. `subset`, fit the model to a subset of the data.
2. `weights`, fit the model with data points of unequal weights.
3. `offset`, fit **generalized linear models** to specify part of the variation in the response.
4. `na.action`, deal with missing values:

- `na.action = na.omit` to leave out any row which has at least one variable missing
- `na.action = na.fail` to fail the fitting process
- `na.action = NULL` to carry out regression with time series data taht include missing values, so the residuals and fitted values are time series as well.

```r
# the use of "subset"
data <- read.table("ipomopsis.txt", header = TRUE)
attach(data)
names(data)
```

```
## [1] "Root"    "Fruit"   "Grazing"
```

```r
model <- lm(Fruit ~ Root, subset = (Grazing == "Grazed"))
# summary(model)

# weights are equal by default
model <- lm(Fruit ~ Grazing, weights = Root) # fit by weighted least squares
detach(data)
```

Akaike's information criterion (AIC)

Also known as penalized log likelihood.

$AIC = -2 \times loglikelihood + 2(p+1)$, where p is the number of parameters in the model, 1 is added for the estimated variance.

```
data <- read.table("regression.txt", header = TRUE)
attach(data)
names(data)
```

```
## [1] "growth" "tannin"
```

```
model <- lm(growth ~ tannin)

# calculate the log likelihood by hand
n <- length(growth)
sse <- sum((growth - fitted(model))^2)
s2 <- sse/ (n - 2)
s <- sqrt(s2)

# the log likelihood
loglike <- -(n/2) * log(2*pi) - n*log(s) - sse/(2*s2)
loglike
```

```
## [1] -16.51087
```

```
# AIC
-2*loglike + 2*(2 + 1)
```

```
## [1] 39.02173
```

```
# use an easier to calculate likelihood and AIC
logLik(model)
```

```
## 'log Lik.' -16.37995 (df=3)
```

```
AIC(model)
```

```
## [1] 38.7599
```

```
detach(data)

data <- read.table("ipomopsis.txt", header = TRUE)
attach(data)
# AIC as a measure of the fit of a model
model.1 <- lm(Fruit ~ Grazing * Root)
model.2 <- lm(Fruit ~ Grazing + Root)
AIC(model.1, model.2)
```

```
##         df      AIC
## model.1  5 273.0135
## model.2  4 271.1279
```

```
# compare multiple models using AIC
model.3 <- lm(Fruit ~ Grazing * Root + I(Root^2))

models <- list(model.1, model.2, model.3)
aic <- unlist(lapply(models, AIC))
aic
```

```
## [1] 273.0135 271.1279 275.0079
```

```r
detach(data)
```

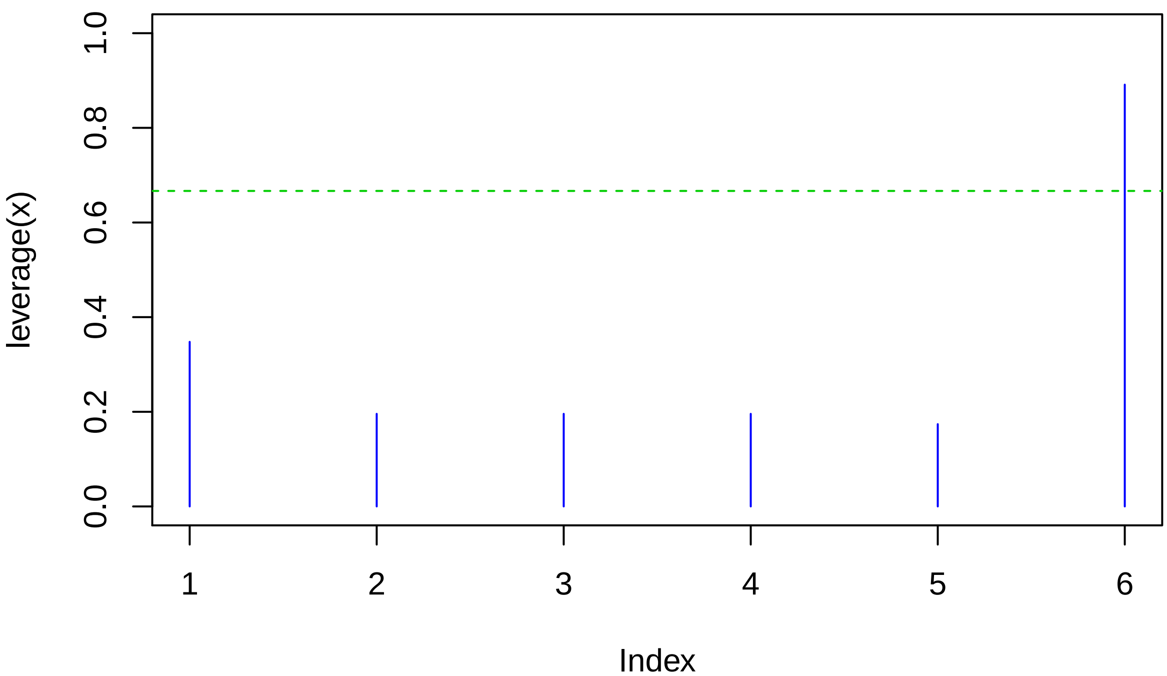
Leverage

Measures of levearage for a given data point y are proportional to $(x - \overline{x})^2$.

One common measure is :

$h_i = \frac{1}{n} + \frac{(x_i - \overline{x})^2}{\sum (x_i - \overline{x})^2}$ and the rule of thumb that a point is highly influential is if $h_i > \frac{2p}{n}$, where p is the number of parameters and n is the sample size.

```r
x <- c(2, 3, 3, 3, 4, 7)
leverage <- function(x) {1/length(x) + (x - mean(x))^2 / sum((x-mean(x))^2)}

plot(leverage(x), type = "h", ylim=c(0, 1), col = "blue")
abline(h = (2 * 2)/length(x), lty = 2, col = 3) # the sixth data point is influential
```

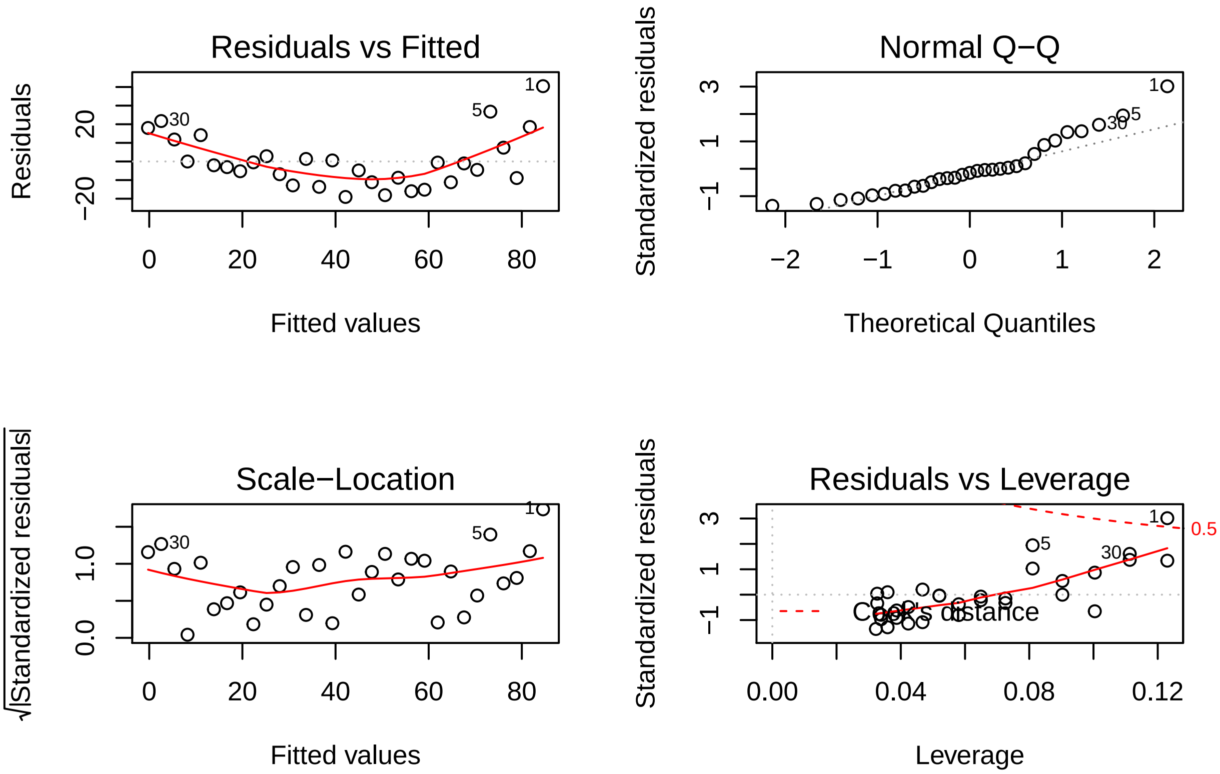


Model checking in R using `plot(model)`

```r
decay <- read.table("Decay.txt", header = TRUE)
attach(decay)
names(decay)
```

```
## [1] "time"   "amount"
```

```r
model <- lm(amount ~ time)
par(mfrow = c(2, 2))
plot(model) # the first two plots are important
```




```
# the third is a postive valued version of the first graph
# the fourth plot is standardized residual vs. levearge tother with Cook's distance,
# which is a combination of leverage and residuals in a single measure.
par(mfrow = c(1, 1))
detach(decay)
```

Extracting information from model objects

1. by name, e.g. , coef(model)
2. with list subscripts, e.g., summary(model)[[3]]
3. using $, e.g., model$resid

```
# by name
data <- read.table("regression.txt",header=T)
attach(data)
names(data)
```

```
## [1] "growth" "tannin"
```

```
model <- lm(growth~tannin)
summary(model)
```

```
##
## Call:
## lm(formula = growth ~ tannin)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.4556 -0.8889 -0.2389  0.9778  2.8944
```

```
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  11.7556     1.0408  11.295 9.54e-06 ***
## tannin       -1.2167     0.2186  -5.565 0.000846 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.693 on 7 degrees of freedom
## Multiple R-squared:  0.8157, Adjusted R-squared:  0.7893
## F-statistic: 30.97 on 1 and 7 DF,  p-value: 0.0008461
```

```
coef(model)
```

```
## (Intercept)      tannin
##   11.755556   -1.216667
```

```
fitted(model)
```

```
##         1         2         3         4         5         6         7
## 11.755556 10.538889  9.322222  8.105556  6.888889  5.672222  4.455556
##         8         9
##  3.238889  2.022222
```

```
resid(model)
```

```
##          1          2          3          4          5          6
##  0.2444444 -0.5388889 -1.3222222  2.8944444 -0.8888889  1.3277778
##          7          8          9
## -2.4555556 -0.2388889  0.9777778
```

```
vcov(model) # variance covariance matrix
```

```
##             (Intercept)      tannin
## (Intercept)    1.083263 -0.19116402
## tannin        -0.191164  0.04779101
```

```
# by list subscripts
summary.aov(model)
```

```
##             Df Sum Sq Mean Sq F value   Pr(>F)
## tannin       1  88.82   88.82   30.97 0.000846 ***
## Residuals    7  20.07    2.87
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
str(summary.aov(model)) # list of 1
```

```
## List of 1
##  $ :Classes 'anova' and 'data.frame':    2 obs. of  5 variables:
##   ..$ Df     : num [1:2] 1 7
##   ..$ Sum Sq : num [1:2] 88.8 20.1
##   ..$ Mean Sq: num [1:2] 88.82 2.87
##   ..$ F value: num [1:2] 31 NA
##   ..$ Pr(>F) : num [1:2] 0.000846 NA
##  - attr(*, "class")= chr [1:2] "summary.aov" "listof"
```

```
summary.aov(model)[[1]][1]
```

```
##             Df
```

```
## tannin       1
## Residuals    7
```

```r
as.numeric(unlist(summary.aov(model)[[1]][4]))[1]
```

```
## [1] 30.97398
```

```r
# using lists with models
x <- 0:100
y <- 17 + 0.2 * x + 3 * rnorm(101)


model0 <- lm(y ~ 1)
model1 <- lm(y ~ x)
model2 <- lm(y ~ x + I(x^2))

models <- list(model0, model1, model2)
lapply(models, coef) # check coefs of all models
```

```
## [[1]]
## (Intercept)
##    27.59817
##
## [[2]]
## (Intercept)           x
##  18.5206497   0.1815504
##
## [[3]]
##   (Intercept)             x        I(x^2)
## 16.9751930368  0.2752144566 -0.0009366404
```

```r
# get a vector as output, all three intercepts
as.vector(unlist(lapply(models,coef)))[c(1,2,4)]
```

```
## [1] 27.59817 18.52065 16.97519
```

```r
lapply(models,AIC) # AIC
```

```
## [[1]]
## [1] 652.8401
##
## [[2]]
## [1] 503.6935
##
## [[3]]
## [1] 499.152
```

```r
rm(x)
rm(y)
detach(data)
```

Contrasts

Rules for constructing coefficients: 1. Treatment to be lumped together get the same sign;

2. Groups of means to be contrasted get opposite sign;
3. Factor levels to be excluded get a contrast coefficient of 0;

4. The coeffcients add up to 0.

Contrasts sum of squares: $SSC = \frac{(\sum \frac{c_i T_i}{n_i})^2}{\sum \frac{c_i^2}{n_i}}$, where T_i is the total of the y values within factor level i.

```
# example of contrast
comp <- read.table("competition.txt",header = TRUE)
attach(comp)
names(comp)
```

```
## [1] "biomass"  "clipping"
```

```
model1 <- aov(biomass ~ clipping)
summary(model1)
```

```
##             Df Sum Sq Mean Sq F value  Pr(>F)
## clipping     4  85356   21339   4.302 0.00875 **
## Residuals   25 124020    4961
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary.lm(model1) # summary for lm method
```

```
##
## Call:
## aov(formula = biomass ~ clipping)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -103.333  -49.667    3.417   43.375  177.667
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   465.17      28.75  16.177  9.4e-15 ***
## clippingn25    88.17      40.66   2.168  0.03987 *
## clippingn50   104.17      40.66   2.562  0.01683 *
## clippingr10   145.50      40.66   3.578  0.00145 **
## clippingr5    145.33      40.66   3.574  0.00147 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 70.43 on 25 degrees of freedom
## Multiple R-squared:  0.4077, Adjusted R-squared:  0.3129
## F-statistic: 4.302 on 4 and 25 DF,  p-value: 0.008752
```

```
levels(clipping)
```

```
## [1] "control" "n25"     "n50"     "r10"     "r5"
```

```
# A priori contrasts
# use specified priori test
contrasts(clipping) <- cbind(c(4,-1,-1,-1,-1),
                             c(0,1,1,-1,-1),
                             c(0,0,0,1,-1),
                             c(0,1,-1,0,0))



clipping
```

```
##  [1] n25     n25     n25     n25     n25     n25     n50     n50    
##  [9] n50     n50     n50     n50     r5      r5      r5      r5     
## [17] r5      r5      control control control control control control
## [25] r10     r10     r10     r10     r10     r10    
## attr(,"contrasts")
##         [,1] [,2] [,3] [,4]
## control    4    0    0    0
## n25       -1    1    0    1
## n50       -1    1    0   -1
## r10       -1   -1    1    0
## r5        -1   -1   -1    0
## Levels: control n25 n50 r10 r5
```

```r
# refit the model
model2 <- aov(biomass ~ clipping)
summary.lm(model2) # coefs are different, se and df are the same
```

```
## 
## Call:
## aov(formula = biomass ~ clipping)
## 
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -103.333  -49.667    3.417   43.375  177.667 
## 
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 561.80000   12.85926  43.688  < 2e-16 ***
## clipping1   -24.15833    6.42963  -3.757 0.000921 ***
## clipping2   -24.62500   14.37708  -1.713 0.099128 .  
## clipping3     0.08333   20.33227   0.004 0.996762    
## clipping4    -8.00000   20.33227  -0.393 0.697313    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 70.43 on 25 degrees of freedom
## Multiple R-squared:  0.4077, Adjusted R-squared:  0.3129 
## F-statistic: 4.302 on 4 and 25 DF,  p-value: 0.008752
```

```r
# first coef is the mean of biamass
mean(biomass)
```

```
## [1] 561.8
```

```r
# the means for different factor levels
tapply(biomass, clipping, mean)
```

```
##  control      n25      n50      r10       r5 
## 465.1667 553.3333 569.3333 610.6667 610.5000
```

```r
# the first contrast
c1 <- factor(1 + (clipping != "control"))
tapply(biomass, c1, mean)
```

```
##        1        2 
## 465.1667 585.9583
```

```r
# the second estimate in the summary is the difference between the overall mean and
# the mean of the four other treatments
mean(biomass) - tapply(biomass,c1,mean)[2]
```

```
##         2
## -24.15833
```

```r
# third contrast for the corresponding group means comparison
c2 <- factor(2*(clipping == "n25") + 2*(clipping == "n50")+
                        (clipping == "r10") + (clipping == "r5"))

tapply(biomass, c2, mean)
```

```
##        0        1        2
## 465.1667 610.5833 561.3333
```

```r
(tapply(biomass, c2, mean)[3]- tapply(biomass, c2, mean)[2])/2
```

```
##       2
## -24.625
```

```r
# rm(biomass)
rm(clipping)
detach(comp)
```

Comparison of three kinds of contrasts

- Treatment contrasts: the default contrast
- Helmert contrasts: default in S-PLUS
- Sum contrasts: see below

```r
# treatment contrasts
options(contrasts = c("contr.treatment", "contr.poly"))
comp <- read.table("competition.txt",header = TRUE)
attach(comp)
contrasts(clipping) # NOT  orthogonal
```

```
##         n25 n50 r10 r5
## control   0   0   0  0
## n25       1   0   0  0
## n50       0   1   0  0
## r10       0   0   1  0
## r5        0   0   0  1
```

```r
output.treatment <- lm(biomass ~ clipping)
summary(output.treatment)
```

```
##
## Call:
## lm(formula = biomass ~ clipping)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -103.333  -49.667    3.417   43.375  177.667
##
```

```
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   465.17      28.75  16.177  9.4e-15 ***
## clippingn25    88.17      40.66   2.168  0.03987 *
## clippingn50   104.17      40.66   2.562  0.01683 *
## clippingr10   145.50      40.66   3.578  0.00145 **
## clippingr5    145.33      40.66   3.574  0.00147 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 70.43 on 25 degrees of freedom
## Multiple R-squared:  0.4077, Adjusted R-squared:  0.3129
## F-statistic: 4.302 on 4 and 25 DF,  p-value: 0.008752
```

```r
# Helmert contrasts
options(contrasts = c("contr.helmert", "contr.poly"))
contrasts(clipping)
```

```
##         [,1] [,2] [,3] [,4]
## control   -1   -1   -1   -1
## n25        1   -1   -1   -1
## n50        0    2   -1   -1
## r10        0    0    3   -1
## r5         0    0    0    4
```

```r
output.helmert <- lm(biomass~clipping)
summary(output.helmert)
```

```
##
## Call:
## lm(formula = biomass ~ clipping)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -103.333  -49.667    3.417   43.375  177.667
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  561.800     12.859  43.688   <2e-16 ***
## clipping1     44.083     20.332   2.168   0.0399 *
## clipping2     20.028     11.739   1.706   0.1004
## clipping3     20.347      8.301   2.451   0.0216 *
## clipping4     12.175      6.430   1.894   0.0699 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 70.43 on 25 degrees of freedom
## Multiple R-squared:  0.4077, Adjusted R-squared:  0.3129
## F-statistic: 4.302 on 4 and 25 DF,  p-value: 0.008752
```

```r
# sum contrast
options(contrasts=c("contr.sum","contr.poly"))
contrasts(clipping)
```

```
##         [,1] [,2] [,3] [,4]
## control    1    0    0    0
```

```
## n25         0    1    0    0
## n50         0    0    1    0
## r10         0    0    0    1
## r5         -1   -1   -1   -1
```

```r
output.sum <- lm(biomass ~ clipping)
summary(output.sum)
```

```
##
## Call:
## lm(formula = biomass ~ clipping)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -103.333  -49.667    3.417   43.375  177.667
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  561.800     12.859  43.688  < 2e-16 ***
## clipping1    -96.633     25.719  -3.757 0.000921 ***
## clipping2     -8.467     25.719  -0.329 0.744743
## clipping3      7.533     25.719   0.293 0.772005
## clipping4     48.867     25.719   1.900 0.069019 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 70.43 on 25 degrees of freedom
## Multiple R-squared:  0.4077, Adjusted R-squared:  0.3129
## F-statistic: 4.302 on 4 and 25 DF,  p-value: 0.008752
```

```r
tapply(biomass,clipping,mean) - 561.8 # the estimates
```

```
##    control        n25        n50        r10         r5
## -96.633333  -8.466667   7.533333  48.866667  48.700000
```

```r
detach(comp)
# reset the default
options(contrasts = c("contr.treatment", "contr.poly"))
```

Aliasing

Occurs when there is no information available on which to base an estimate of a parameter value.

1. Intrinsic aliasing occurs when it's due to the structure of the model; for example, two variables are perfectly correlated, then including both into a model will result in one zero parameter estimate.

2. Extrinsic aliasing occurs when it's due to the nature of the data; for example, a certain combination of the factors have zero observations accidentally, then this particular combination will contribute do data to the response variable and then cannot be estiamted.

Polynomial contrasts

Orthogonal polynomial contrasts for a factor with four levels:

term	x_1	x_2	x_3	x_4
linear	-3	-1	1	3
quadratic	1	-1	-1	1
cubic	-1	3	-3	1

```
data <- read.table("poly.txt", header = TRUE)
attach(data)
names(data)
```

```
## [1] "treatment" "response"
```

```
is.factor(treatment)
```

```
## [1] TRUE
```

```
is.ordered(treatment) # is factor but not ordered
```

```
## [1] FALSE
```

```
contrasts(treatment) # contr.treatment
```

```
##        low medium none
## high     0      0    0
## low      1      0    0
## medium   0      1    0
## none     0      0    1
```

```
treatment <- ordered(treatment,levels=c("none","low","medium","high"))
levels(treatment)
```

```
## [1] "none"   "low"    "medium" "high"
```

```
contrasts(treatment) # contr.poly: orthogonal polynomial contrasts
```

```
##              .L   .Q         .C
## [1,] -0.6708204  0.5 -0.2236068
## [2,] -0.2236068 -0.5  0.6708204
## [3,]  0.2236068 -0.5 -0.6708204
## [4,]  0.6708204  0.5  0.2236068
```

```
model2 <- lm(response ~ treatment)
summary.aov(model2)
```

```
##             Df Sum Sq Mean Sq F value   Pr(>F)
## treatment    3  41.69  13.896    24.7 2.02e-05 ***
## Residuals   12   6.75   0.563
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary.lm(model2)
```

```
##
## Call:
## lm(formula = response ~ treatment)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -1.25  -0.50   0.00   0.50   1.00
```

```
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   4.8125     0.1875  25.667 7.45e-12 ***
## treatment.L   1.7330     0.3750   4.621 0.000589 ***
## treatment.Q  -2.6250     0.3750  -7.000 1.43e-05 ***
## treatment.C  -0.7267     0.3750  -1.938 0.076520 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.75 on 12 degrees of freedom
## Multiple R-squared:  0.8606, Adjusted R-squared:  0.8258
## F-statistic:  24.7 on 3 and 12 DF,  p-value: 2.015e-05
```

```
# fit polynomial regression model to the mean values of the response with the four ordered levels
yv <- as.vector(tapply(response, treatment, mean))
x <- 1:4
model <- lm(yv ~ x + I(x^2) + I(x^3))
summary(model)
```

```
##
## Call:
## lm(formula = yv ~ x + I(x^2) + I(x^3))
##
## Residuals:
## ALL 4 residuals are 0: no residual degrees of freedom!
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.0000         NA      NA       NA
## x            -1.7083         NA      NA       NA
## I(x^2)        2.7500         NA      NA       NA
## I(x^3)       -0.5417         NA      NA       NA
##
## Residual standard error: NaN on 0 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:    NaN
## F-statistic:   NaN on 3 and 0 DF,  p-value: NA
```

```
x <- 1:4
x2 <- x^2
x3 <- x^3
cor(cbind(x, x2,x3))
```

```
##            x        x2        x3
## x  1.0000000 0.9843740 0.9513699
## x2 0.9843740 1.0000000 0.9905329
## x3 0.9513699 0.9905329 1.0000000
```

```
t(contrasts(treatment)) # linear, quadratic and cubic
```

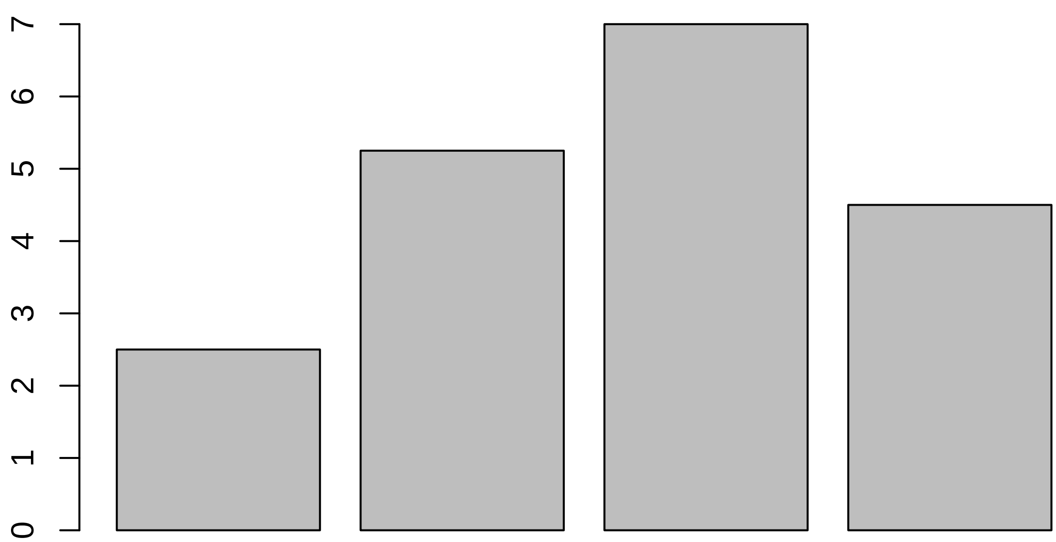
```
##          [,1]       [,2]       [,3]      [,4]
## .L -0.6708204 -0.2236068  0.2236068 0.6708204
## .Q  0.5000000 -0.5000000 -0.5000000 0.5000000
## .C -0.2236068  0.6708204 -0.6708204 0.2236068
```

```
# draw barplot to see how the curve looks like
y <- as.vector(tapply(response, treatment, mean))
```

```r
model <- lm(y ~ poly(x, 3))
summary(model)
```

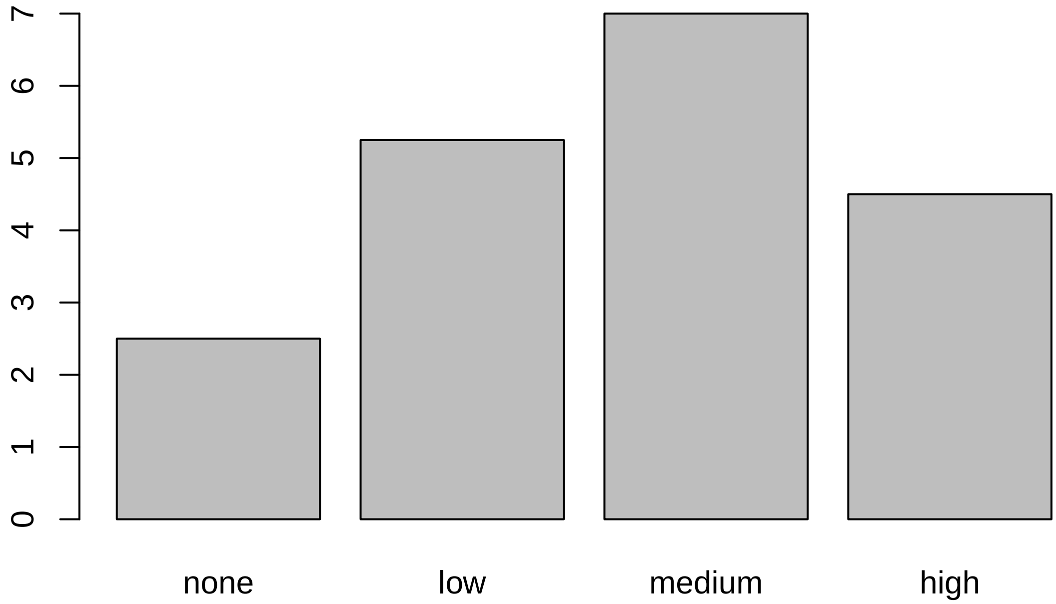
```
##
## Call:
## lm(formula = y ~ poly(x, 3))
##
## Residuals:
## ALL 4 residuals are 0: no residual degrees of freedom!
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   4.8125         NA      NA       NA
## poly(x, 3)1   1.7330         NA      NA       NA
## poly(x, 3)2  -2.6250         NA      NA       NA
## poly(x, 3)3  -0.7267         NA      NA       NA
##
## Residual standard error: NaN on 0 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:    NaN
## F-statistic:   NaN on 3 and 0 DF,  p-value: NA
```

```r
xv <- seq(1, 4, 0.1)
yv <- predict(model, list(x=xv))

(bar.x <- barplot(y))
```

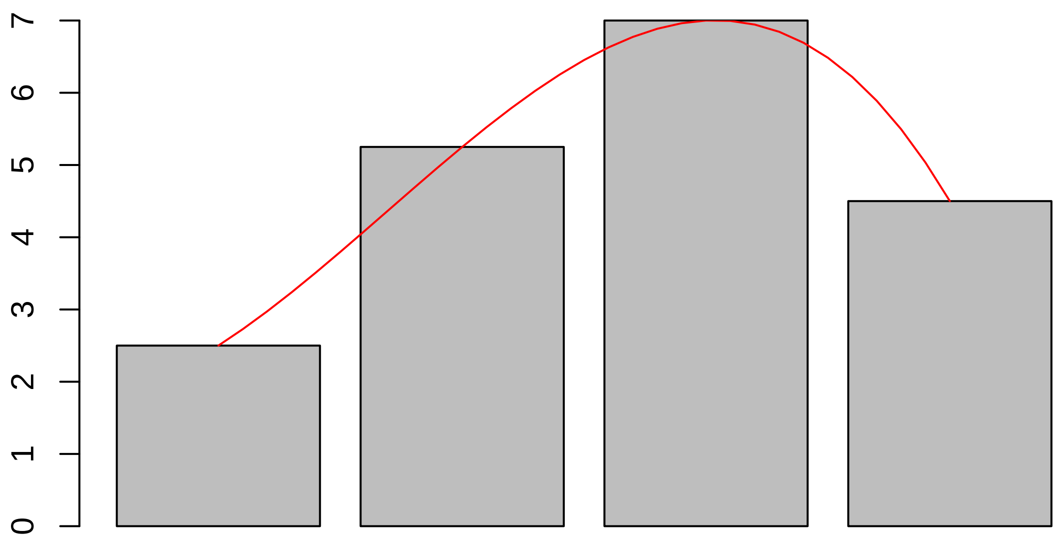



```
##      [,1]
## [1,]  0.7
## [2,]  1.9
## [3,]  3.1
## [4,]  4.3
```

```r
barplot(y, names = levels(treatment))
```




```
barplot(y)
xs <- -0.5 + 1.2 * xv
lines(xs,yv,col="red")
```




```
rm(y)
detach(data)
```

Summary for statistical modeling 1. Data description for possible errors and outliers

2. Model specification
3. Check if there is no pseudoreplication, or need to specify appropriate random effects
4. Fit models and do model cheking and model simplification.

Chapter 10 Regression

Important kinds of regression: 1. linear regression

2. polynomial regression for non-liearity relationship
3. piecewise regression for two or more adjacent straight lines
4. robust regression for less sensitivity to outliers
5. multiple regression for many explanatory variables
6. non-linear regression for a specified non-linear model
7. non-parametric regression for data when there is no obvious functional form.

The essence of regression analysis is using sample data to estimate paramter values and their standard errors.

Linear regression

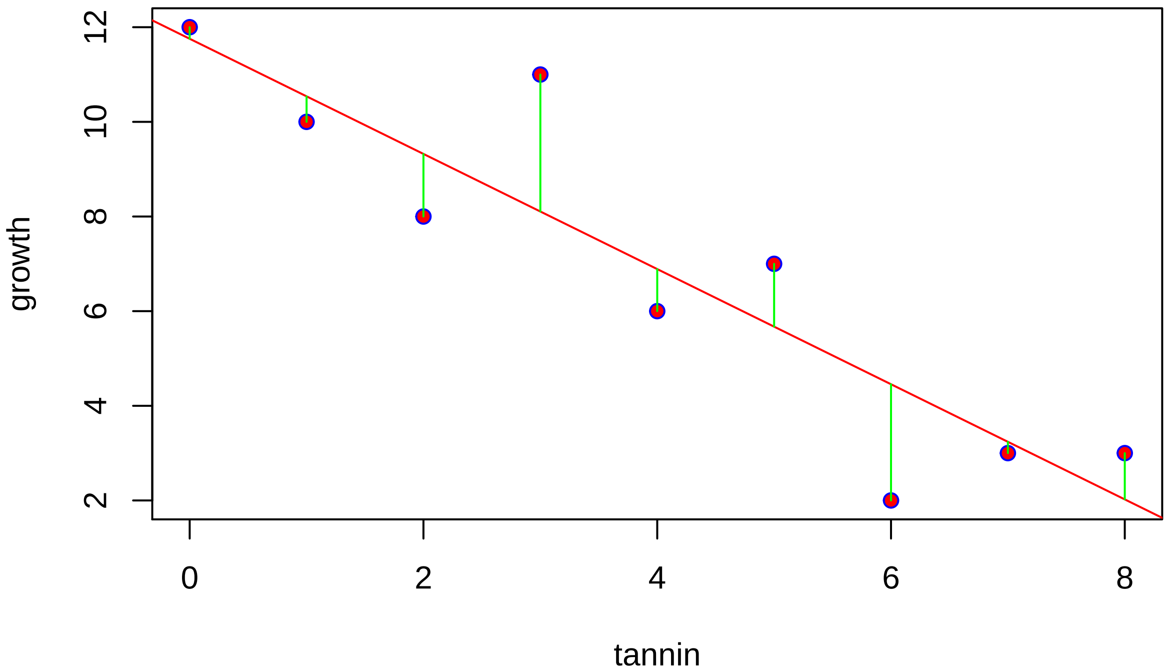
Assumptions:

- variance of y is constant
- x is measured with error
- residuals are normally distributed

```
reg.data <- read.table("regression.txt",header=T)
attach(reg.data)
names(reg.data)
```

```
## [1] "growth" "tannin"
```

```
# fit a linear regression
model <- lm(growth ~ tannin)
plot(tannin,growth, pch=21, col="blue", bg="red")
abline(model, col="red")
yhat <- predict(model,tannin = tannin)

join <- function(i)
 lines(c(tannin[i],tannin[i]),c(growth[i],yhat[i]),col="green")

# apply the function to all data points
sapply(1:9,join)
```




```
## [[1]]
## NULL
##
## [[2]]
## NULL
##
## [[3]]
## NULL
##
## [[4]]
## NULL
##
## [[5]]
## NULL
##
## [[6]]
## NULL
##
## [[7]]
## NULL
##
## [[8]]
## NULL
##
## [[9]]
## NULL
```

```
# sum of squares of residuals
# null model
ssy <- deviance(lm(growth ~ 1))
ssy
```

```
## [1] 108.8889
```

```r
# linear model
sse <- deviance(lm(growth  ~ tannin))
sse
```

```
## [1] 20.07222
```

```r
# then the R-square is
r_square <- (ssy - sse)/ssy
r_square
```

```
## [1] 0.8156633
```

```r
# absolute value correlation coefficient
r <- sqrt(r_square)
r # while the sign is determined by the sign of EXY
```

```
## [1] 0.9031408
```

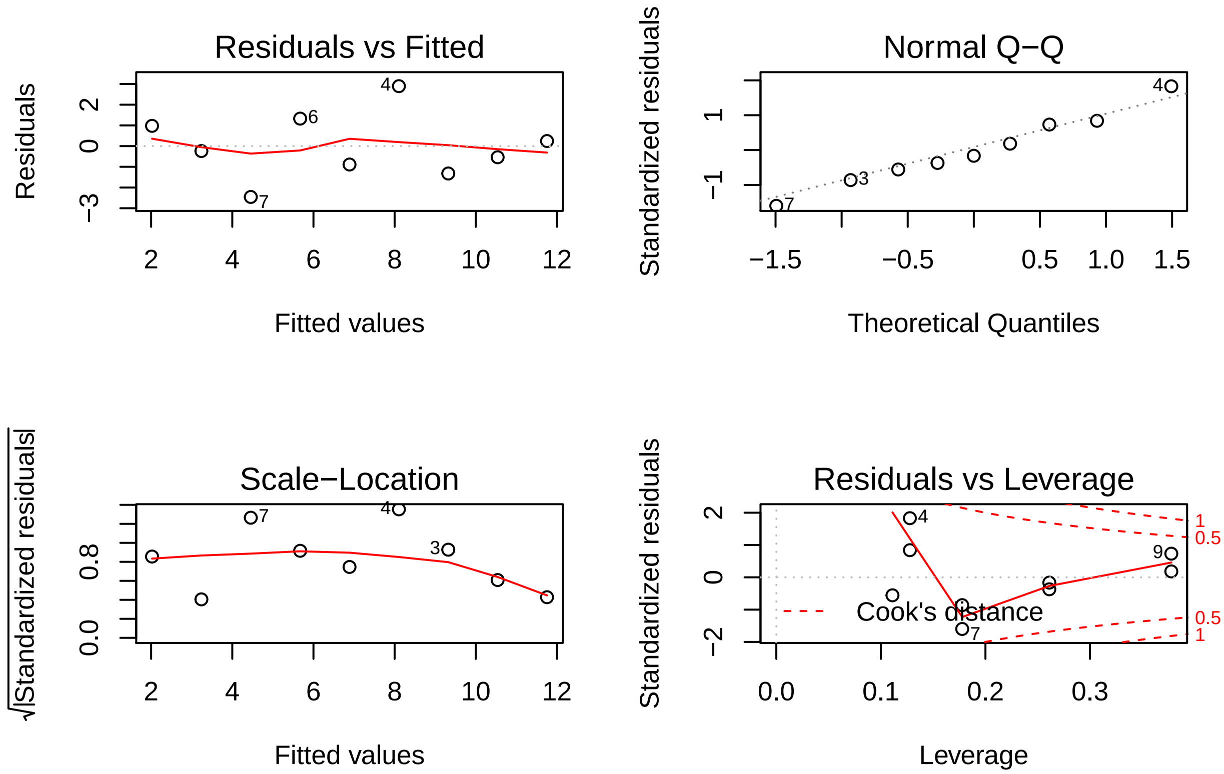
```r
# analysis of variance
anova(lm(growth ~ tannin))
```

```
## Analysis of Variance Table
##
## Response: growth
##           Df Sum Sq Mean Sq F value    Pr(>F)
## tannin     1 88.817  88.817  30.974 0.0008461 ***
## Residuals  7 20.072   2.867
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# confidence interval for coefficients
confint(model)
```

```
##                 2.5 %     97.5 %
## (Intercept)  9.294457 14.2166544
## tannin      -1.733601 -0.6997325
```

```r
# model checking
par(mfrow = c(2, 2))
plot(model)
```




```r
par(mfrow = c(1, 1))

model2 <- update(model,subset=(tannin != 6))
summary(model2)
```

```
##
## Call:
## lm(formula = growth ~ tannin, subset = (tannin != 6))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.4549 -0.9572 -0.1622  0.4572  2.6622
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  11.6892     0.8963  13.042 1.25e-05 ***
## tannin       -1.1171     0.1956  -5.712  0.00125 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.457 on 6 degrees of freedom
## Multiple R-squared:  0.8446, Adjusted R-squared:  0.8188
## F-statistic: 32.62 on 1 and 6 DF,  p-value: 0.001247
```

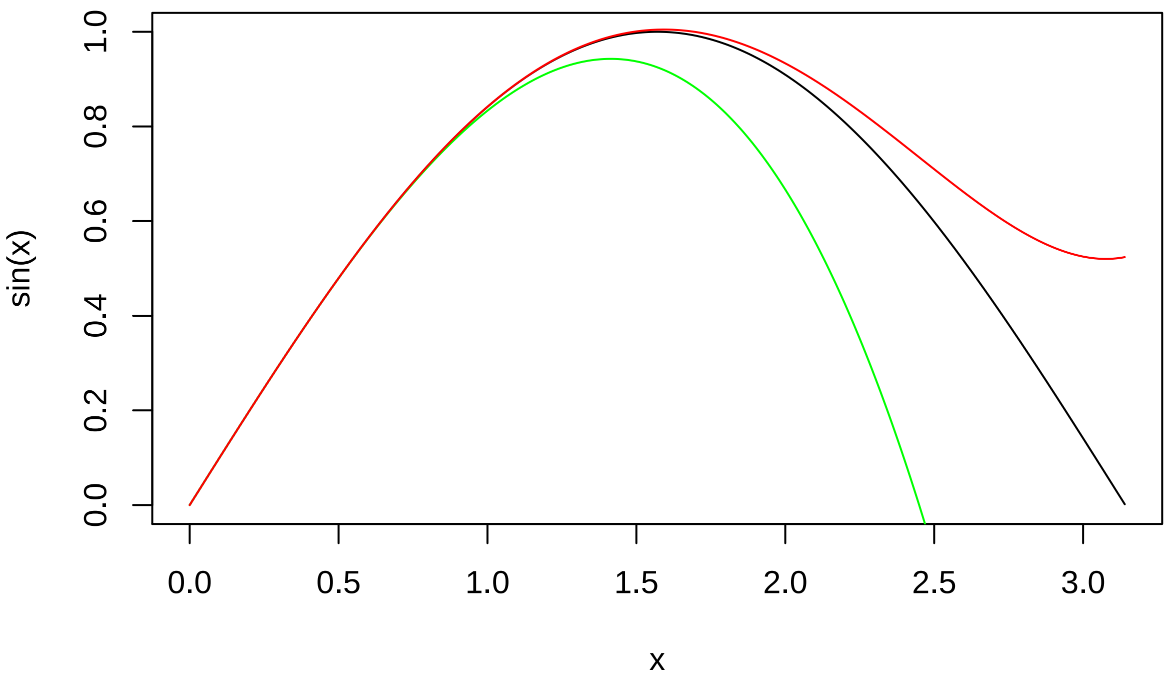
```r
detach(reg.data)
```

Polynomial approximations to elementary functions

Elementary functions expressed as Maclaurin series:

1. $sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$
2. $cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots$
3. $exp(x) = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$
4. $log(x+1) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \cdots$

```r
# approximation of sin(x)

x <- seq(0,pi,0.01)
y <- sin(x)

# the original sin(x) curve
plot(x,y,type="l",ylab="sin(x)")

# approximation by the first two terms
a1 <- x - x^3/factorial(3)
lines(x, a1, col = "green")

# by the first three terms
a2 <- x - x^3/factorial(3) + x^5/factorial(5)
lines(x, a2, col = "red")
```



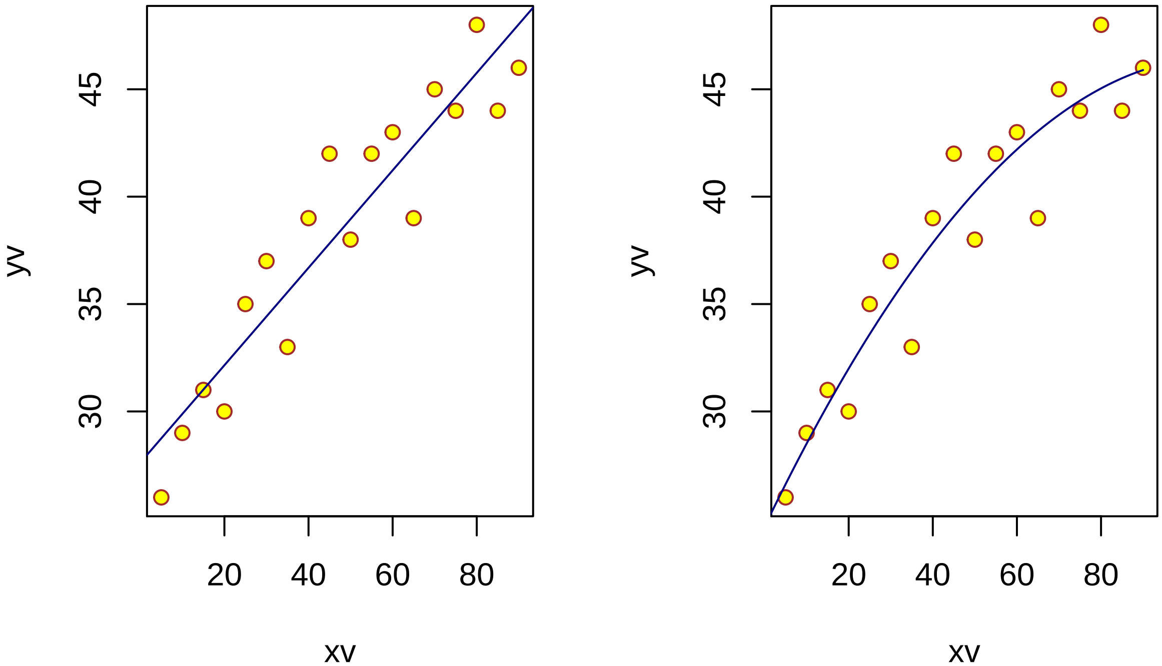

```r
# good appriximations for small values
rm(list = c("x", "y"))
```

Polynomial regression

```r
rm(list = c("xv", "yv")) # remove possible variables with the same names used below
poly <- read.table("diminish.txt",header=T)
attach(poly)
names(poly)
```

```
## [1] "xv" "yv"
```

```r
par(mfrow=c(1,2))

# linear model
model1 <- lm(yv ~ xv)
plot(xv, yv, pch=21, col = "brown", bg = "yellow")
abline(model1, col="navy")

# quadratic
model2 <- lm(yv ~ xv + I(xv^2))

plot(xv, yv, pch=21, col = "brown", bg = "yellow")
x <- 0:90
y <- predict(model2, list(xv = x))
lines(x, y, col = "navy")
```




```r
par(mfrow = c(1, 1))

# F test to see if the effect of the quadratic term is significant
anova(model1,model2) # significant
```

```
## Analysis of Variance Table
##
## Model 1: yv ~ xv
## Model 2: yv ~ xv + I(xv^2)
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     16 91.057
## 2     15 68.143  1    22.915 5.0441 0.0402 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
rm(list = c("x", "y")) # , "xv", "yv"))
detach(poly)
```

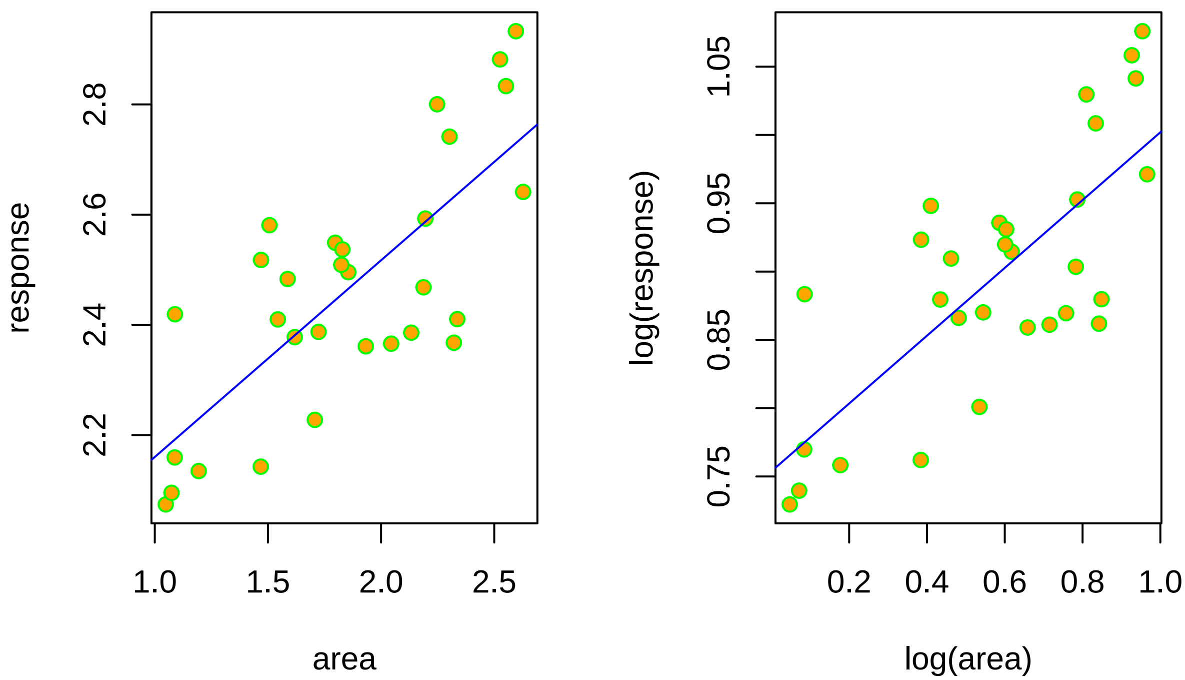
Linear regression after transformation

The most frequent transformations are logarithms, antilogs and reciprocals.

```r
power <- read.table("power.txt",header = TRUE)
attach(power)
names(power)
```

```
## [1] "area"     "response"
```

```r
par(mfrow = c(1, 2))
# plot on original scale
plot(area, response, pch = 21, col = "green", bg = "orange")
abline(lm(response ~ area), col = "blue")
# plot on log scale
plot(log(area), log(response), pch = 21, col = "green", bg = "orange")
abline(lm(log(response) ~ log(area)), col = "blue")
```

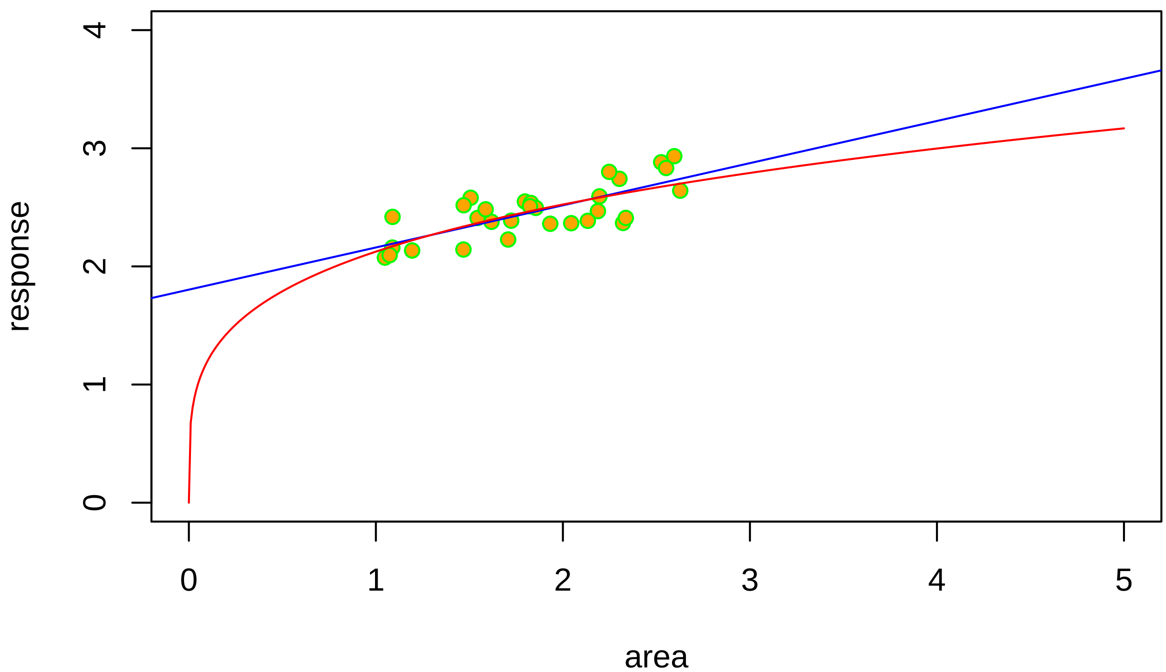



```r
par(mfrow = c(1, 1))

# linear model on original scale
model1 <- lm(response ~ area)

# log scale
model2 <- lm(log(response) ~ log(area))
summary(model2)
```

```
##
## Call:
## lm(formula = log(response) ~ log(area))
```

```
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -0.100937 -0.043289 -0.000562  0.046095  0.108453
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.75378    0.02613  28.843  < 2e-16 ***
## log(area)    0.24818    0.04083   6.079 1.48e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06171 on 28 degrees of freedom
## Multiple R-squared:  0.5689, Adjusted R-squared:  0.5535
## F-statistic: 36.96 on 1 and 28 DF,  p-value: 1.48e-06
```

```r
# a visual comparison of two models
plot(area, response, xlim = c(0, 5), ylim = c(0, 4), pch = 21, col = "green", bg = "orange")
abline(lm(response ~ area), col = "blue")
xv <- seq(0, 5, 0.01)
# log y = a + b logx , y = exp(a)*exp(logx^b) = exp(a) * x^b
yv <- exp(coef(model2)[1])*xv^coef(model2)[2]
lines(xv, yv, col = "red")
```




```r
# more data will be helpful for choosing models

rm(list = c("xv", "yv"))
detach(power)
```

Prediction following regression

Extrapolation : prediction beyond the measured range of the data

Interpolation : predition within the measured range of the data, can often be accurate and not affected by model choice.

```
attach(reg.data)
plot(tannin, growth, pch = 21, col = "blue", bg = "red")
model <- lm(growth ~ tannin)
abline(model, col = "blue")

coef(model)[2] # b1
```

```
##    tannin
## -1.216667
```

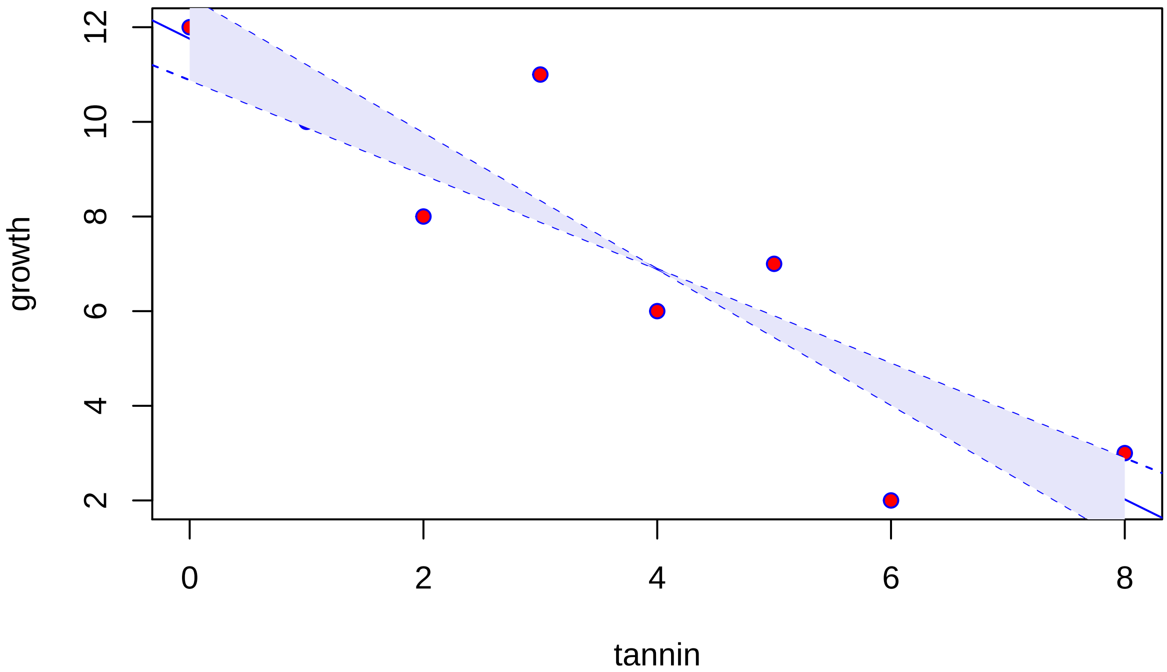
```
names(summary(model))
```

```
##  [1] "call"          "terms"         "residuals"     "coefficients"
##  [5] "aliased"       "sigma"         "df"            "r.squared"
##  [9] "adj.r.squared" "fstatistic"    "cov.unscaled"
```

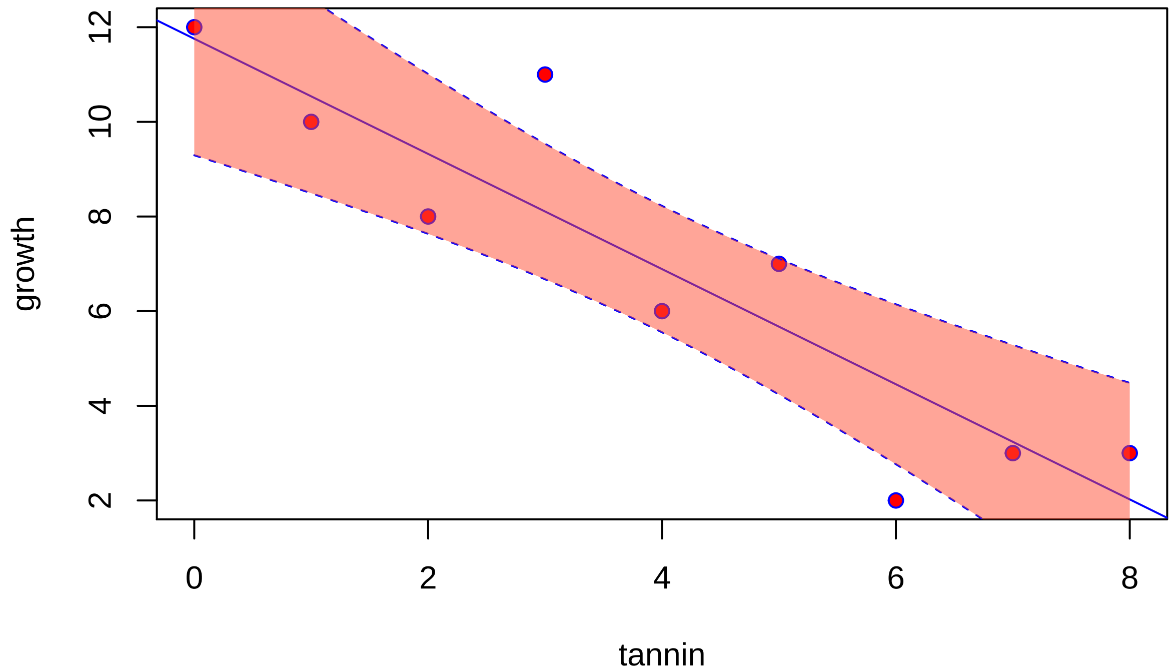
```
summary(model)[[4]][4] # standard error of b1
```

```
## [1] 0.2186115
```

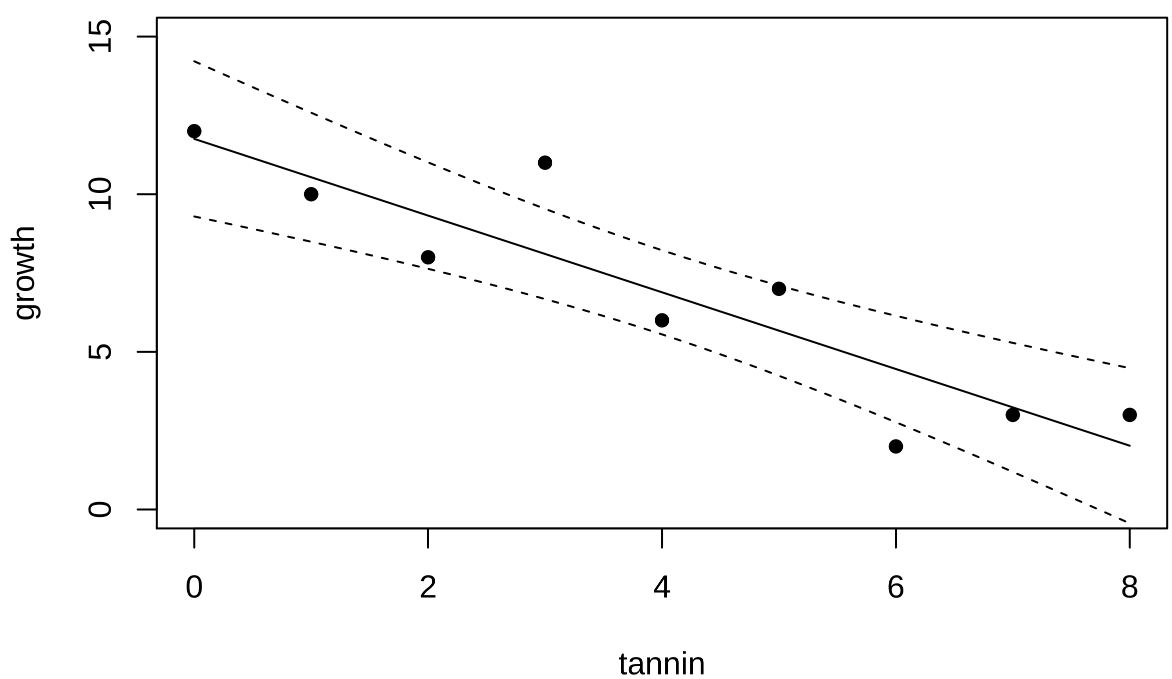
```
# add standard error lines for ONE standard error
se.lines <- function(model){
b1 <- coef(model)[2] + summary(model)[[4]][4]
b2 <- coef(model)[2] - summary(model)[[4]][4]

# model[[12]] is the original data set and [2] means the x values tannin
xm <- sapply(model[[12]][2], mean) # mean tannin value
ym <- sapply(model[[12]][1], mean) # mean response growth value
a1 <- ym - b1 * xm
a2 <- ym - b2 * xm
abline(a1, b1,lty=2,col="blue")
abline(a2,b2,lty=2,col="blue")
polygon(c(rev(tannin), tannin), c(rev(a1 + b1 * tannin), a2 + b2* tannin), col = "lavender", border = N
}

se.lines(model)
```




```
# add confidence intervals

ci.lines <- function(model){
xm <- sapply(model[[12]][2], mean) # the overall mean for tannin (x)
n <- sapply(model[[12]][2], length) # total number of data
ssx <- sum(model[[12]][2]^2)-sum(model[[12]][2])^2/n  # model[[12]] is original data
s.t <- qt(0.975, (n-2)) # construct the 95% confidence interval from t distribution
xv <- seq(min(model[[12]][2]), max(model[[12]][2]), length=100) # sequence depends on max and min x val
yv <- coef(model)[1] + coef(model)[2]*xv
se <- sqrt(summary(model)[[6]]^2*(1/n+(xv - xm)^2/ssx))  # summary(model)[[6]] for sigma
ci <- s.t*se
uyv <- yv+ci
lyv <- yv-ci
lines(xv,uyv,lty=2,col="blue")
lines(xv,lyv,lty=2,col="blue")
polygon(c(rev(xv), xv), c(rev(uyv), lyv),
        col = rgb(1, 0.3, 0.2, alpha = 0.5),
        border = NA) # use rgb to specify a color with transparency
# rgb(red, green, blue, alpha, names = NULL, maxColorValue = 1)
}

plot(tannin,growth,pch=21,col="blue",bg="red")
abline(model, col="blue")
ci.lines(model)
```




```
# speed up the intervals drawing by using int = "c" and matlines
plot(tannin, growth, pch=16, ylim=c(0, 15))
model <- lm(growth ~ tannin)

xv <- seq(0,8,0.1)
yv <- predict(model, list(tannin=xv), int="c")
matlines(xv, yv, lty=c(1, 2, 2), col = "black", border = NA)
```




```
detach(reg.data)
```

Testing for lack of fit in a regression

```r
data <- read.delim("lackoffit.txt")
names(data)
```

```
## [1] "conc" "rate"
```

```r
attach(data)
plot(conc, jitter(rate), pch = 16, col = "red",
     ylim = c(0, 8), ylab = "Rate")
abline(lm(rate ~ conc), col = "blue")


model.reg <- lm(rate ~ conc)
summary(model.reg)
```

```
##
## Call:
## lm(formula = rate ~ conc)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.96429 -0.90476  0.09524  0.27381  2.15476
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   6.7262     0.4559  14.755 7.35e-12 ***
## conc         -0.9405     0.1264  -7.439 4.85e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.159 on 19 degrees of freedom
## Multiple R-squared:  0.7444, Adjusted R-squared:  0.7309
## F-statistic: 55.33 on 1 and 19 DF,  p-value: 4.853e-07
```

```r
# pure error variance by setting each level a factor
fac.conc <- factor(conc)
model.aov <- aov(rate ~ fac.conc)
summary(model.aov)
```

```
##             Df Sum Sq Mean Sq F value   Pr(>F)
## fac.conc     6  87.81  14.635   17.07 1.05e-05 ***
## Residuals   14  12.00   0.857
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# lack of fit
anova(model.reg, model.aov)
```

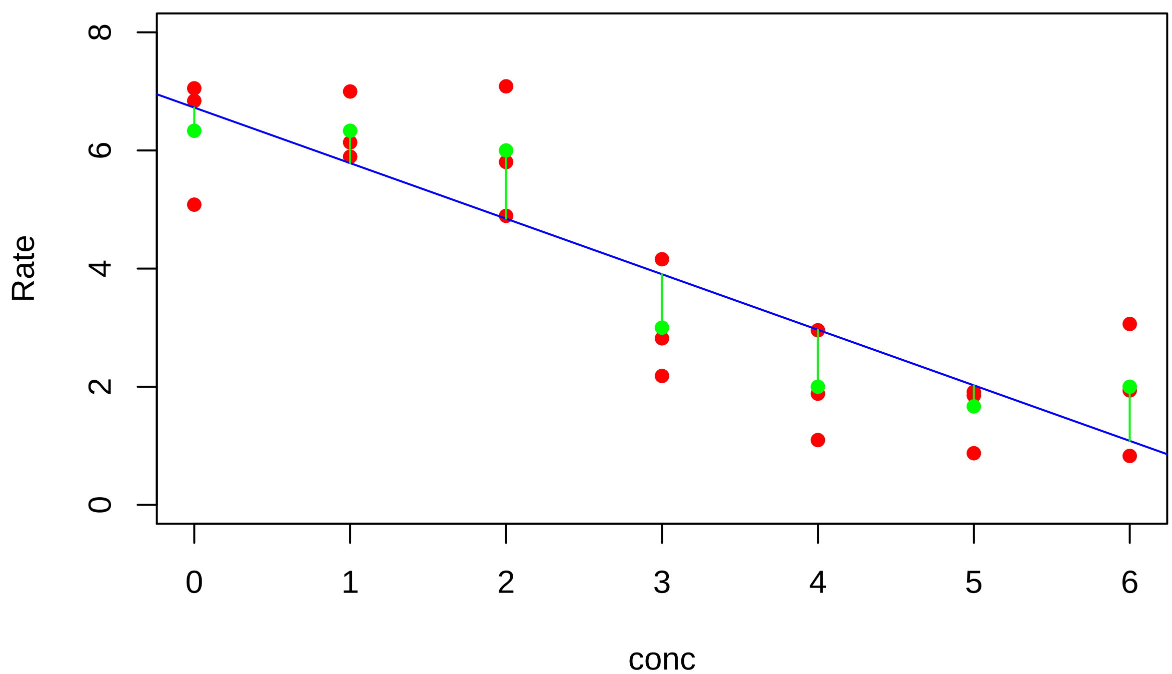
```
## Analysis of Variance Table
##
## Model 1: rate ~ conc
## Model 2: rate ~ fac.conc
##   Res.Df    RSS Df Sum of Sq      F  Pr(>F)
## 1     19 25.512
## 2     14 12.000  5    13.512 3.1528 0.04106 *
```

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

```r
# a single ANOVA table showing the lack-of-fit sum of squares by fitting both models
anova(lm(rate ~ conc + fac.conc))
```

```
## Analysis of Variance Table
##
## Response: rate
##           Df Sum Sq Mean Sq F value    Pr(>F)
## conc       1 74.298  74.298 86.6806 2.247e-07 ***
## fac.conc   5 13.512   2.702  3.1528   0.04106 *
## Residuals 14 12.000   0.857
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# a visual impression of this lack of fit

# means for each level
my <- as.vector(tapply(rate, fac.conc, mean))

for (i in 0:6) lines(c(i,i),c(my[i+1], predict(model.reg, list(conc = 0:6))[i+1]), col = "green")

points(0:6, my, pch = 16, col = "green")
```




```r
detach(data)
```

Bootstrap with regression , another way is jackknife

An alternative to estimate confidence intervals.

** Two ways of doing bootstrapping**:

1. sample cases with replacement, so some points are left off the graph while others appear more than

once in the dataframe.

2. calculate the residuals from the fitting regression model, and randomize which fitted y values get which residuals.

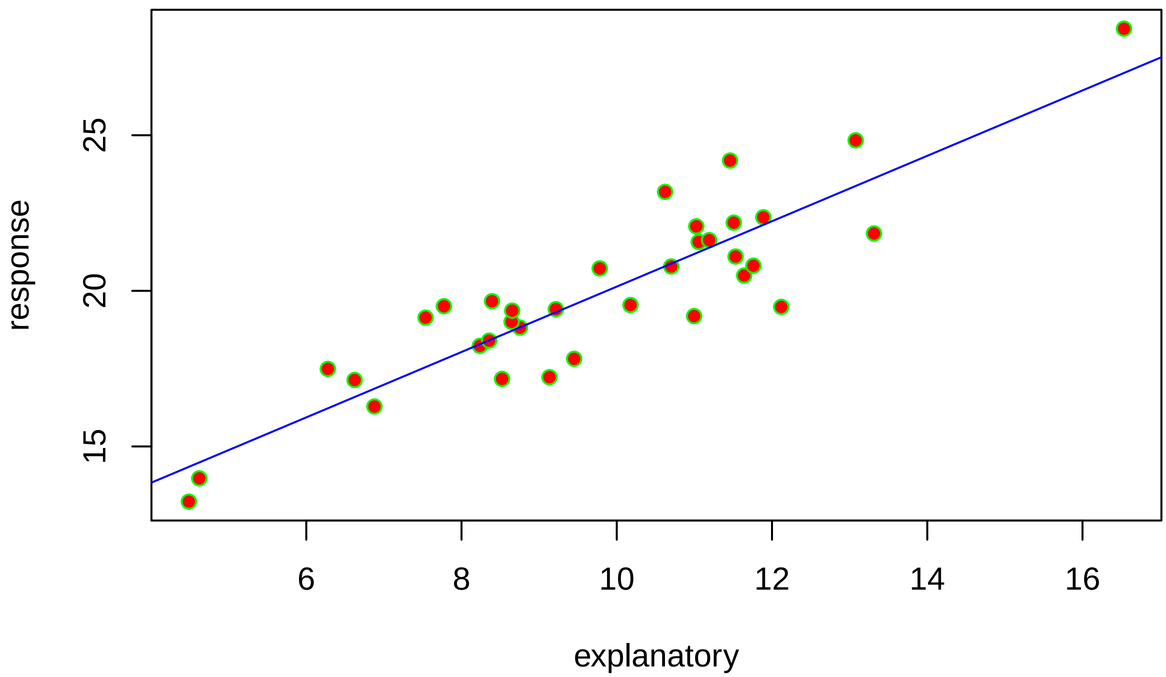
```
regdat <- read.table("regdat.txt", header = TRUE)
names(regdat)
```

```
## [1] "explanatory" "response"
```

```
rm(response)
```

```
## Warning in rm(response): object 'response' not found
```

```
attach(regdat)
plot(explanatory, response, pch = 21, col = "green", bg = "red")
model <- lm(response ~ explanatory)
abline(model, col = "blue")
```

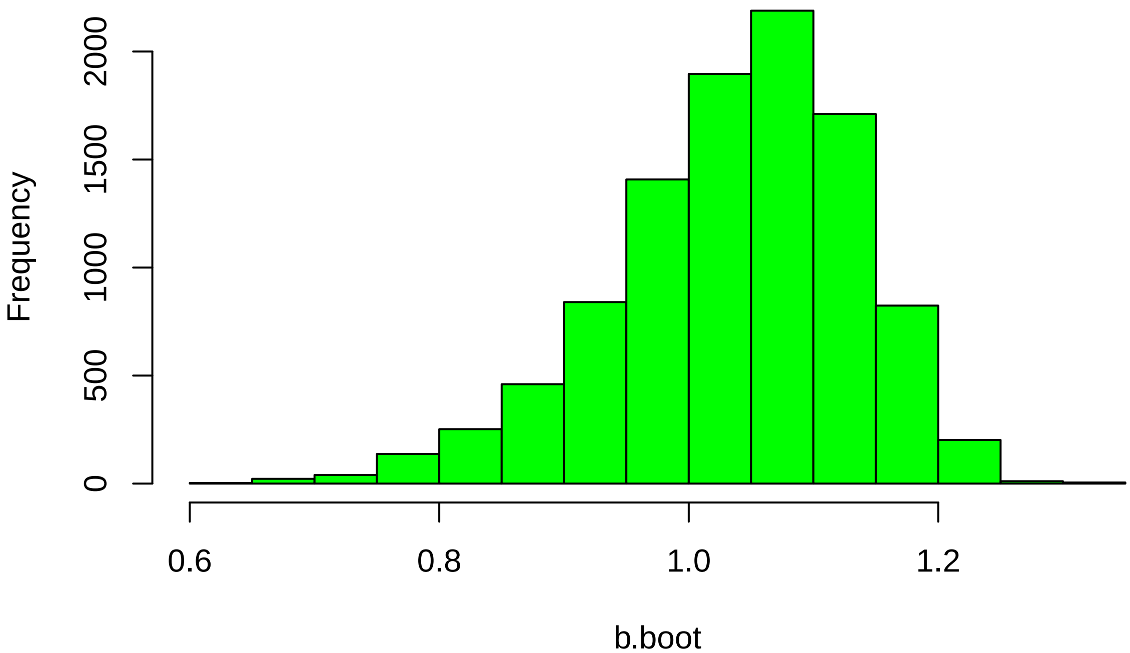



```
model
```

```
##
## Call:
## lm(formula = response ~ explanatory)
##
## Coefficients:
## (Intercept)  explanatory
##       9.630        1.051
```

```
# confidence interval from bootstrap
b.boot <- numeric(10000)
for(i in 1:10000){
  indices <- sample(1:length(response), replace = TRUE)
  xv <- explanatory[indices]
  yv <- response[indices]
  model <- lm(yv ~ xv)
```

```
  b.boot[i] <- coef(model)[2]
}

hist(b.boot, main = "", col = "green")
```




```
# 95% interval for the bootstrapped estimate of the slope
quantile(b.boot, c(0.025, 0.975))
```

```
##      2.5%     97.5%
## 0.8132099 1.1972981
```

```
# repeat the above exercise using the boot function

library(boot)
reg.boot <- function(regdat, index){
  xv <- explanatory[index]
  yv <- response[index]
  model <- lm(yv ~ xv)
  coef(model)
}

reg.model <- boot(regdat, reg.boot, R = 10000)
boot.ci(reg.model, index = 2) # index indicates the position of the variable of interest
```

```
## Warning in boot.ci(reg.model, index = 2): bootstrap variances needed for
## studentized intervals
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = reg.model, index = 2)
##
## Intervals :
```

```
## Level      Normal              Basic
## 95%   ( 0.874,  1.250 )   ( 0.904,  1.277 )
##
## Level     Percentile            BCa
## 95%   ( 0.824,  1.197 )   ( 0.830,  1.200 )
## Calculations and Intervals on Original Scale
```

```
# randomize the allocation of the residuals to fitted y values estimated from the original regression

model <- lm(response ~ explanatory)
fit <- fitted(model) # fitted values
res <- resid(model) # residuals

# function used for bootstrapping
residual.boot <- function(res, index){
y <- fit + res[index]
model <- lm(y ~ explanatory)
coef(model) }

res.model <- boot(res, residual.boot, R = 10000)
boot.ci(res.model, index = 2)
```

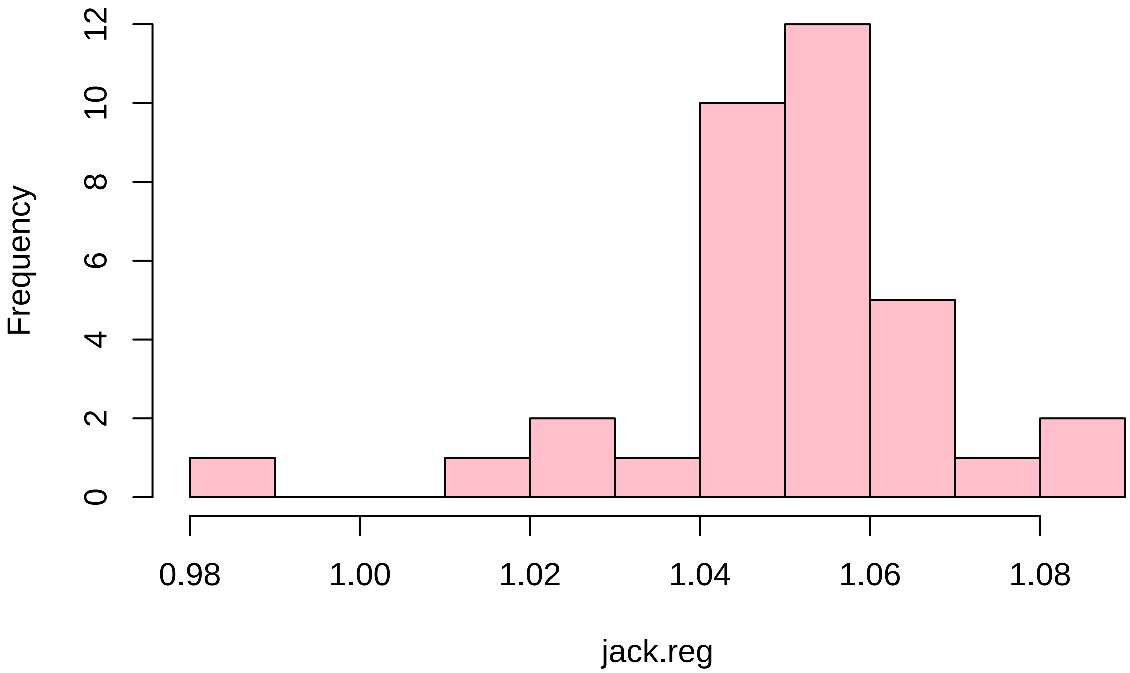
```
## Warning in boot.ci(res.model, index = 2): bootstrap variances needed for
## studentized intervals
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = res.model, index = 2)
##
## Intervals :
## Level      Normal              Basic
## 95%   ( 0.876,  1.222 )   ( 0.876,  1.220 )
##
## Level     Percentile            BCa
## 95%   ( 0.881,  1.225 )   ( 0.878,  1.221 )
## Calculations and Intervals on Original Scale
```

Jackknife

Each point in the data set is left out, one at a time and the paramter of interest is re-estimated. For more details, see reference.

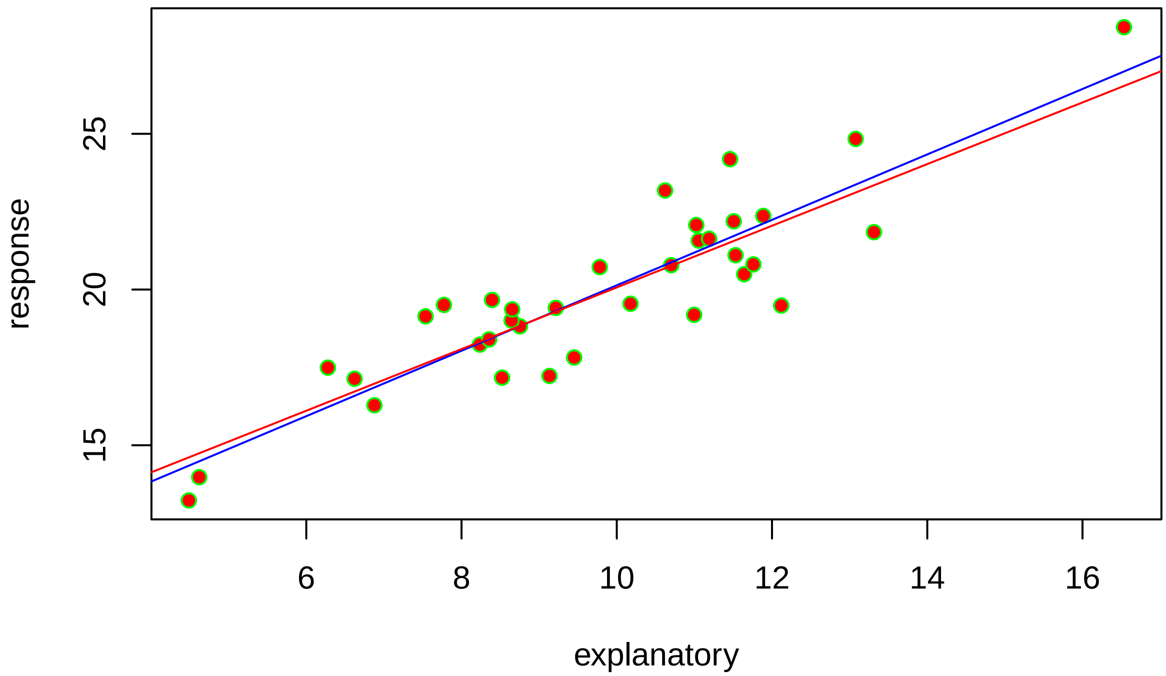
```
jack.reg <- numeric(length(response))


# carry out the regression 35 times leaving out one pair of x and y values

for(i in 1:35){
  model <- lm(response[-i] ~ explanatory[-i])
  jack.reg[i] <- coef(model)[2]
}

hist(jack.reg, main = "", col = "pink") # heavily left skewed
```




```
# check which point is the most influential
# in this case it's the point which caused the extreme left hand bar


# extract Cook's Distance by infmat[, 5]
model <- lm(response ~ explanatory)
which(influence.measures(model)$infmat[, 5] == max(influence.measures(model)$infmat[, 5]))
```

```
## 22
## 22
```

```
# plot the data and do regresion without this point
plot(explanatory ,response, pch = 21, col = "green", bg = "red")
abline(model, col = "blue")
abline(lm(response[-22] ~ explanatory[-22]), col = "red") # no big difference
```

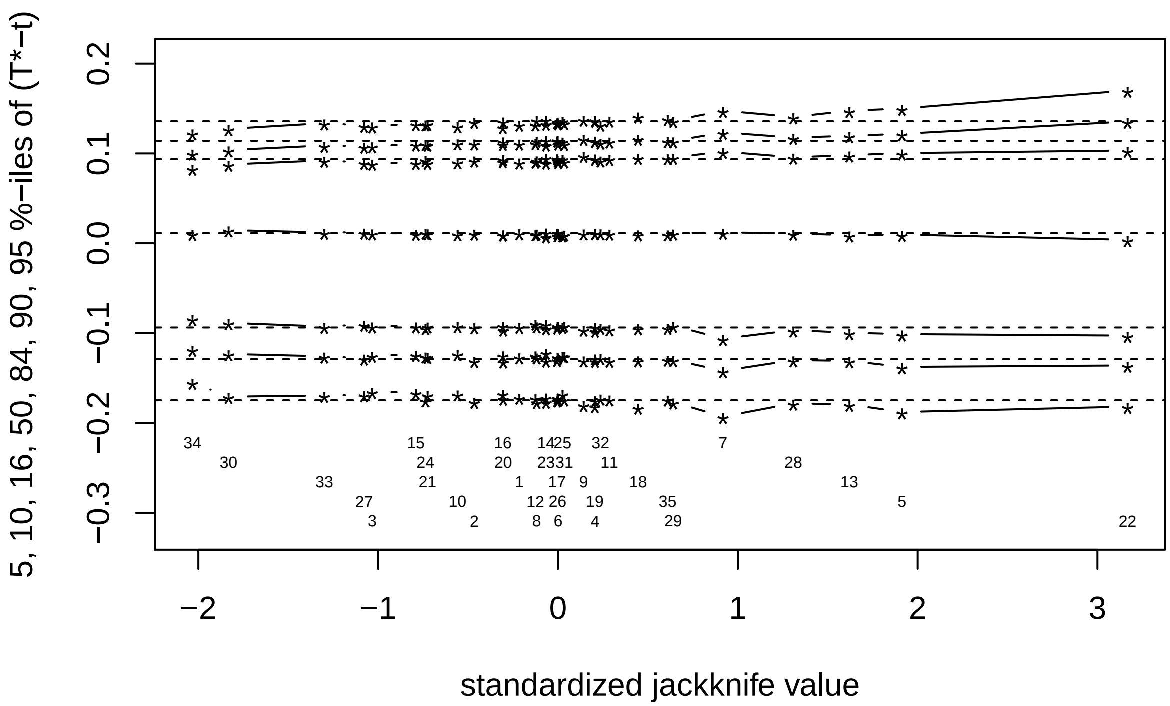



```r
detach(regdat)
```

Jackknife after bootstrap

`jack.after.boot` calculates the jacknife influence values from a bootstrap output object, and plots the corresponding jackknife after bootstrap plot.

```r
jack.after.boot(reg.model, index = 2)
```



```r
# reg.model <- boot(regdat, reg.boot, R = 10000) from above

# The centred jackknife quantiles for each observation are estimated from those bootstrap samples
# These are then plotted against the influence values.
# From the top downwards, the horizontal dotted lines show the 95th, 90th, 84th, 50th, 16th, 10th and 5
# the influence of point no. 22 shows up clearly (this time on the right-hand side),
# indicating that it has a strong positive influence on the slope,
# and the two left-hand outliers are identified as points nos 34 and 30.
```

Serial correlation in the residuals

`durbinWatsonTest`, the Durbin-Watson function is used for testing whether there is autocorrelation in the residuals from a linear model or a generalized linear model.

```r
library(car)
```

```
## Warning: package 'car' was built under R version 3.3.2

##
## Attaching package: 'car'

## The following object is masked from 'package:boot':
##
##     logit
```

```r
durbinWatsonTest(model)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1     -0.07946739      2.049899   0.886
##  Alternative hypothesis: rho != 0
```

```r
# no evidence of serial correlation
```

Piecewise regression

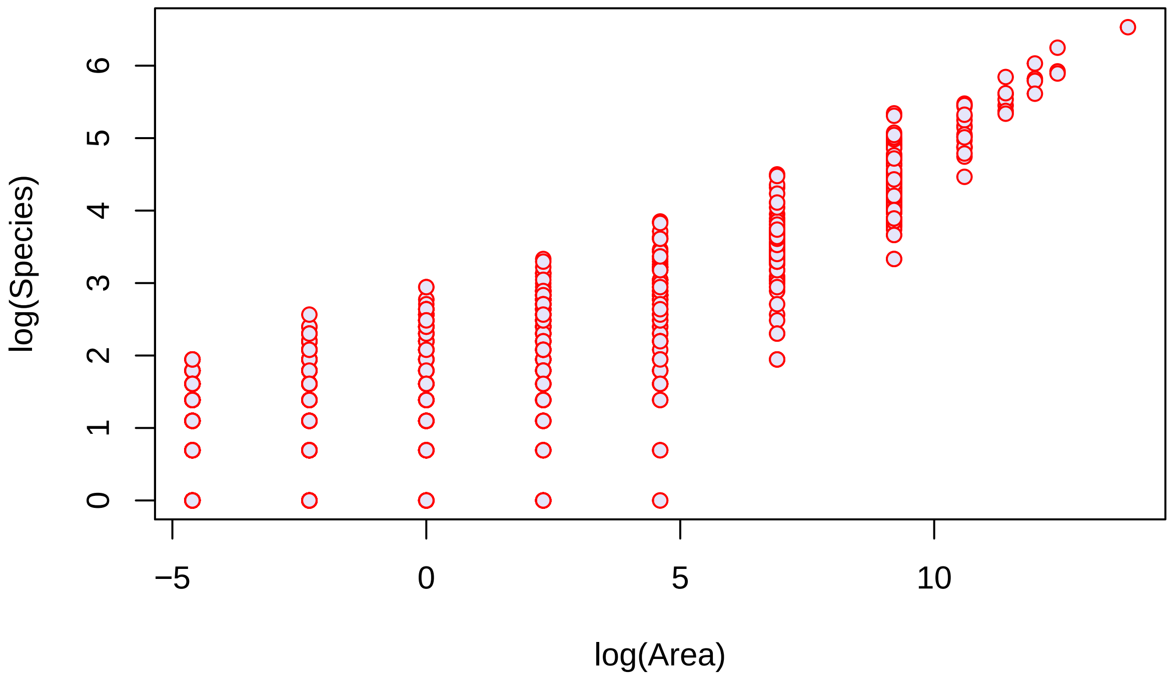
Two problems to be solved:

1. How many segments needed?
2. Where are the break points on the x axis?

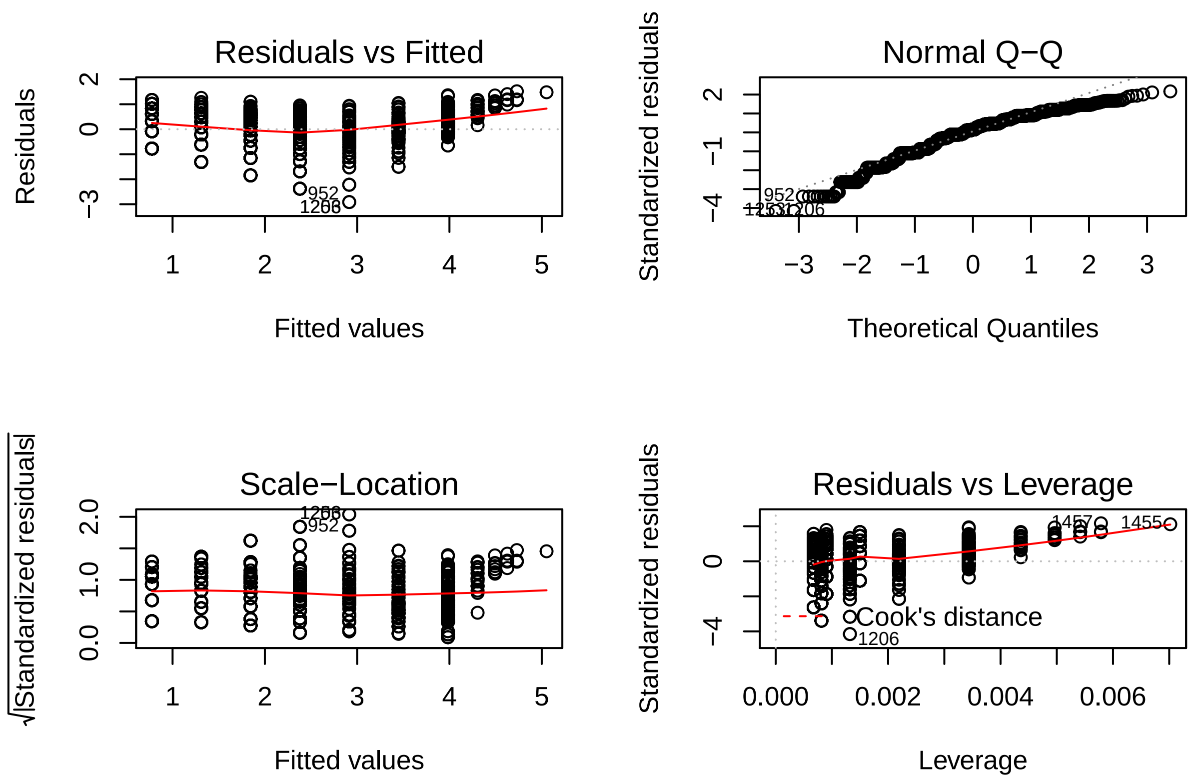
```r
data <- read.table("sasilwood.txt", header = TRUE)
attach(data)
names(data)
```

```
## [1] "Species" "Area"
```

```r
plot(log(Species) ~ log(Area), pch = 21, col = "red", bg = "lavender")
```




```r
# check the model
model1 <- lm(log(Species) ~ log(Area))
par(mfrow = c(2, 2))
plot(model1) # not good
```




```r
par(mfrow = c(1, 1))

# check where to break given that using two segments
```

```r
table(Area)
```

```
## Area
##   0.01    0.1      1     10    100   1000  10000  40000  90000 160000
##    346    345    259    239     88     67    110     18      7      4
## 250000  1e+06
##      3      1
```

```r
# include a logical statement as part of the model formula for piecewise regression

# check the break points of each Area and find the one with minimum standard error
Break <- sort(unique(Area))[3:11] # check the middle ones
Break
```

```
## [1]      1     10    100   1000  10000  40000  90000 160000 250000
```

```r
d <- numeric(length(Break))
for(i in 1:length(Break)){
  model <- lm(log(Species) ~ (Area < Break[i]) * log(Area) +
                 (Area >= Break[i]) * log(Area))
  d[i] <- summary(model)$sigma
}

# location to have break with smallest sigma
which(d == min(d))
```

```
## [1] 3
```

```r
Break[which(d == min(d))]
```

```
## [1] 100
```

```r
# fit model with corresponding break
model2 <- lm(log(Species) ~ log(Area) * (Area < 100) + log(Area) * (Area >= 100))

anova(model1, model2)
```

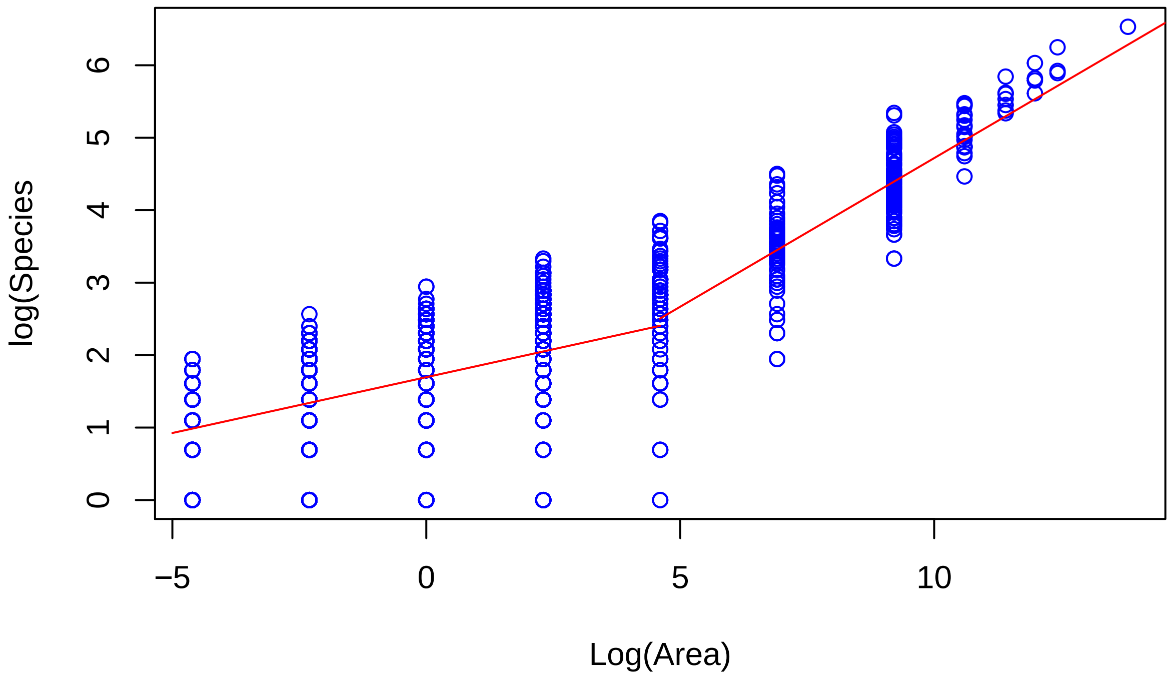
```
## Analysis of Variance Table
##
## Model 1: log(Species) ~ log(Area)
## Model 2: log(Species) ~ log(Area) * (Area < 100) + log(Area) * (Area >=
##     100)
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1   1485 731.98
## 2   1483 631.36  2    100.62 118.17 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
summary(model2)$coef
```

```
##                              Estimate Std. Error    t value      Pr(>|t|)
## (Intercept)                 0.6168168 0.13058554   4.723469  2.537983e-06
## log(Area)                   0.4101943 0.01654883  24.786903 9.081618e-114
## Area < 100TRUE              1.0785395 0.13245572   8.142642  8.117406e-16
## log(Area):Area < 100TRUE -0.2561147 0.01816373 -14.100333  1.834740e-42
```

```r
# visulization

a1 <- summary(model2)[[4]][1] + summary(model2)[[4]][3]
```

```r
a2 <- summary(model2)[[4]][1]
b1 <- summary(model2)[[4]][2] + summary(model2)[[4]][4]
b2 <- summary(model2)[[4]][2]

plot(log(Area), log(Species), col="blue", xlab = "Log(Area)", ylab = "log(Species")
lines(c(-5, 4.6), c(a1 + b1*-5, a1 + b1*4.6), col = "red")
lines(c(4.6, 15), c(a2 + b2*4.6, a2 + b2*15), col = "red")
```




```r
detach(data)
```

Multiple regression

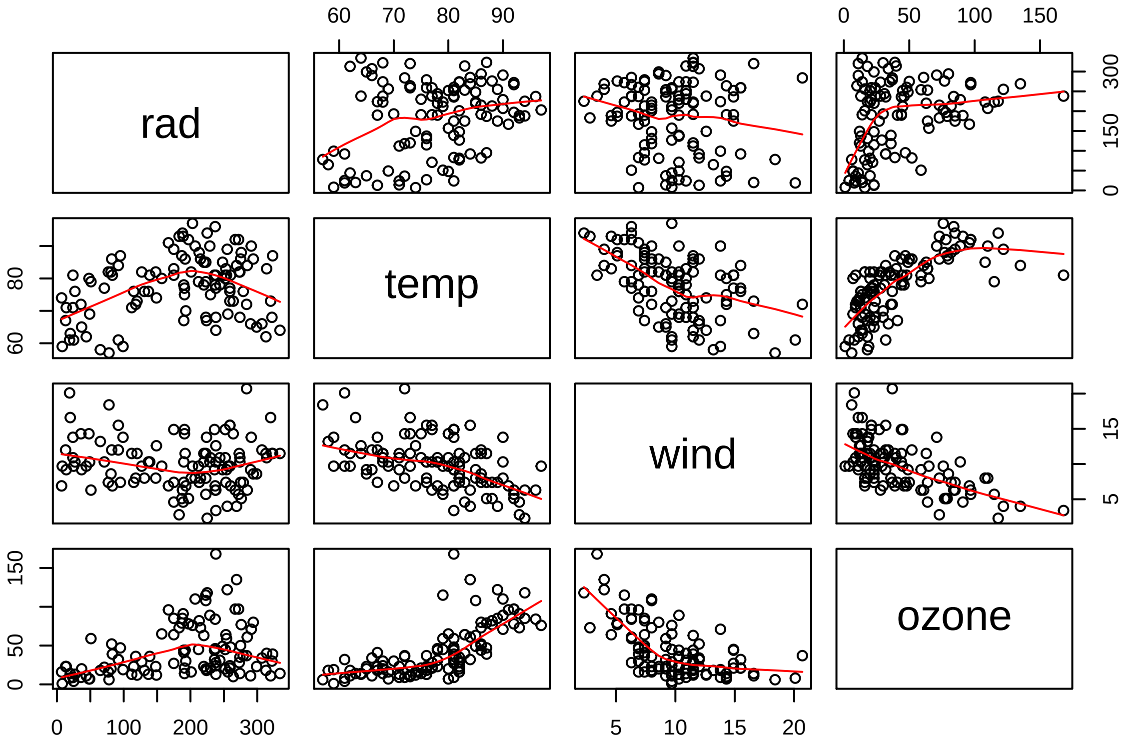
Possible problems:

- over fitting
- parameter proliferation due to curvature, interaction ...
- multi-colinearity
- non-independence of groups of measurements
- temporal or spatial correlation amongst the explanatory variables
- pseudoreplication

```r
ozone.pollution <- read.table("ozone.data.txt", header = TRUE)
attach(ozone.pollution)
names(ozone.pollution)
```

```
## [1] "rad"   "temp"  "wind"  "ozone"
```

```r
# scatterplot matrix for a visual check about correlations
pairs(ozone.pollution,panel = panel.smooth)
```

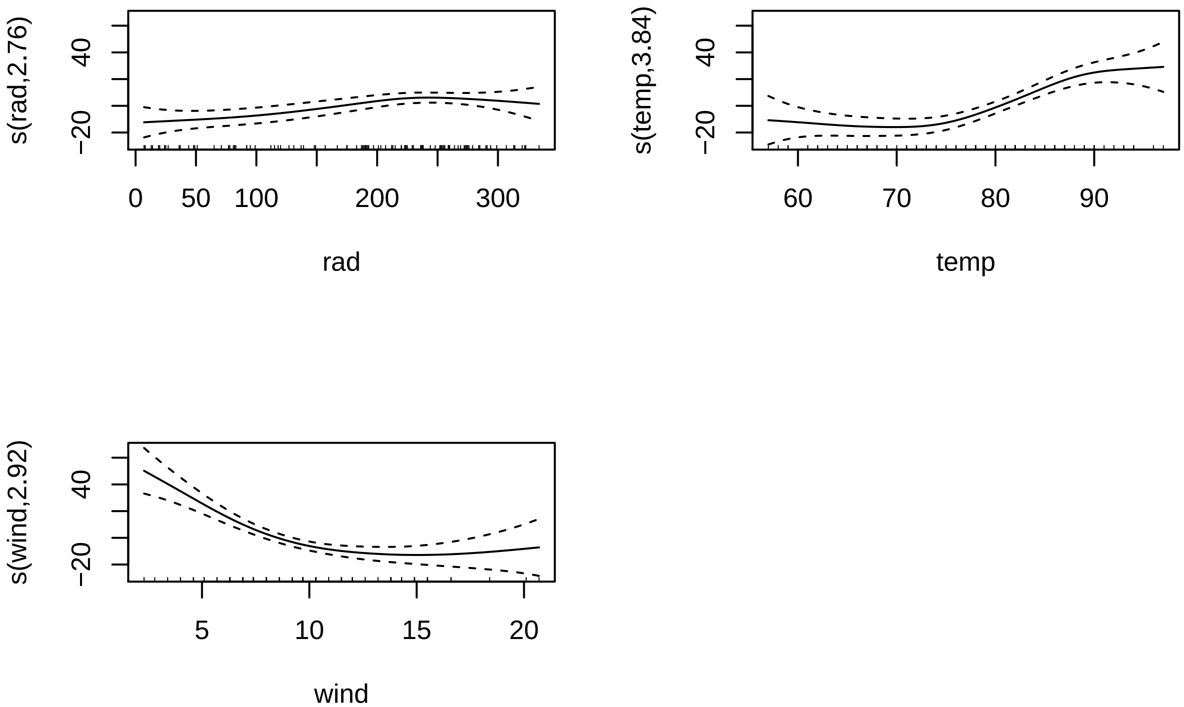



```r
# use non-parametric smoothers in generalized additive models
library(mgcv)
```

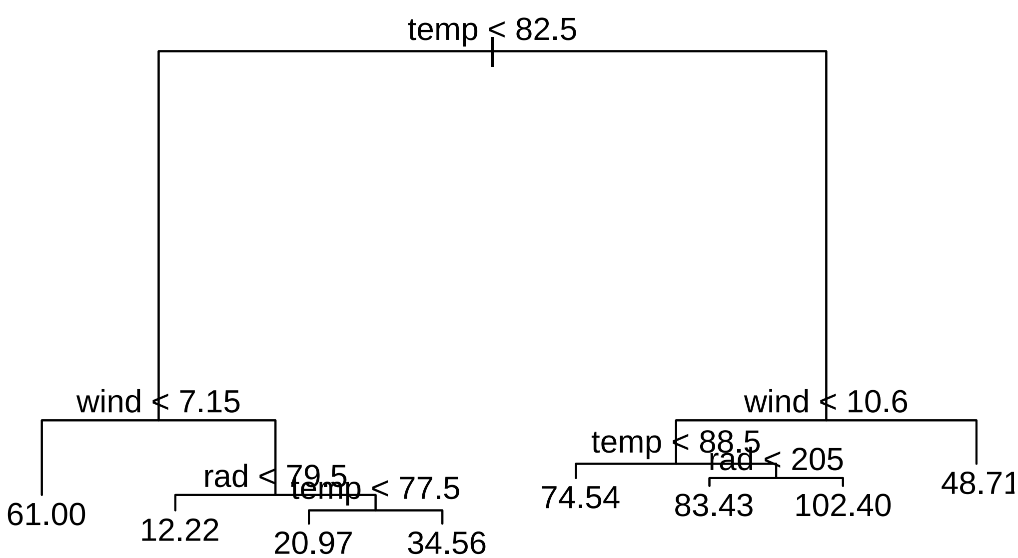
```
## Loading required package: nlme
```

```
## This is mgcv 1.8-16. For overview type 'help("mgcv-package")'.
```

```r
par(mfrow = c(2,2))
model <- gam(ozone ~ s(rad) + s(temp) + s(wind))
plot(model) # check if the confidence intervals are narrow or wide


# fit tree model to figure out which factor(s) is(are) important and see
# the interactions between variables
library(tree)
model <- tree(ozone ~ ., data = ozone.pollution)
par(mfrow=c(1,1))
```




```
plot(model)
text(model)
```




```
# initial complex model to begin with
w2 <- wind^2
t2 <- temp^2
r2 <- rad^2
tw <- temp*wind
wr <- wind*rad
tr <- temp*rad
wtr <- wind*temp*rad

model1 <- lm(ozone ~ rad + temp + wind + t2 + w2 + r2 + wr + tr + tw + wtr)
summary(model1)
```

```
##
## Call:
## lm(formula = ozone ~ rad + temp + wind + t2 + w2 + r2 + wr +
##     tr + tw + wtr)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -38.894 -11.205  -2.736   8.809  70.551
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5.683e+02  2.073e+02   2.741  0.00725 **
## rad         -3.117e-01  5.585e-01  -0.558  0.57799
## temp        -1.076e+01  4.303e+00  -2.501  0.01401 *
## wind        -3.237e+01  1.173e+01  -2.760  0.00687 **
## t2           5.833e-02  2.396e-02   2.435  0.01668 *
## w2           6.106e-01  1.469e-01   4.157 6.81e-05 ***
## r2          -3.619e-04  2.573e-04  -1.407  0.16265
## wr           2.054e-02  4.892e-02   0.420  0.67552
## tr           8.403e-03  7.512e-03   1.119  0.26602
## tw           2.377e-01  1.367e-01   1.739  0.08519 .
## wtr         -4.324e-04  6.595e-04  -0.656  0.51358
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 17.82 on 100 degrees of freedom
## Multiple R-squared:  0.7394, Adjusted R-squared:  0.7133
## F-statistic: 28.37 on 10 and 100 DF,  p-value: < 2.2e-16
```

```
# remove wtr
model2 <- update(model1, ~ .-wtr)
summary(model2)
```

```
##
## Call:
## lm(formula = ozone ~ rad + temp + wind + t2 + w2 + r2 + wr +
##     tr + tw)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -39.611 -11.455  -2.901   8.548  70.325
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5.245e+02  1.957e+02   2.680   0.0086 **
## rad          2.628e-02  2.142e-01   0.123   0.9026
## temp        -1.021e+01  4.209e+00  -2.427   0.0170 *
## wind        -2.802e+01  9.645e+00  -2.906   0.0045 **
## t2           5.953e-02  2.382e-02   2.499   0.0141 *
## w2           6.173e-01  1.461e-01   4.225 5.25e-05 ***
## r2          -3.388e-04  2.541e-04  -1.333   0.1855
## wr          -1.127e-02  6.277e-03  -1.795   0.0756 .
## tr           3.750e-03  2.459e-03   1.525   0.1303
## tw           1.734e-01  9.497e-02   1.825   0.0709 .
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 17.77 on 101 degrees of freedom
## Multiple R-squared:  0.7383, Adjusted R-squared:  0.715
## F-statistic: 31.66 on 9 and 101 DF,  p-value: < 2.2e-16
```

```r
model3 <- update(model2, ~ .-r2)
summary(model3)
```

```
##
## Call:
## lm(formula = ozone ~ rad + temp + wind + t2 + w2 + wr + tr +
##     tw)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -39.188 -11.387  -1.500   8.752  71.289
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept) 486.346603 194.333075   2.503  0.01392 *
## rad          -0.043163   0.208535  -0.207  0.83644
## temp         -9.446780   4.185240  -2.257  0.02613 *
## wind        -26.471461   9.610816  -2.754  0.00697 **
## t2            0.056966   0.023835   2.390  0.01868 *
## w2            0.599709   0.146069   4.106 8.14e-05 ***
## wr           -0.011359   0.006300  -1.803  0.07435 .
## tr            0.003160   0.002428   1.302  0.19600
## tw            0.157637   0.094595   1.666  0.09869 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 17.83 on 102 degrees of freedom
## Multiple R-squared:  0.7337, Adjusted R-squared:  0.7128
## F-statistic: 35.12 on 8 and 102 DF,  p-value: < 2.2e-16
```

```r
model4 <- update(model3, ~ .-tr)
summary(model4)
```

```
##
## Call:
## lm(formula = ozone ~ rad + temp + wind + t2 + w2 + wr + tw)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -41.379 -11.375  -2.217   8.921  71.247
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept) 514.401470 193.783580   2.655  0.00920 **
## rad           0.212945   0.069283   3.074  0.00271 **
## temp        -10.654041   4.094889  -2.602  0.01064 *
## wind        -27.391965   9.616998  -2.848  0.00531 **
## t2            0.067805   0.022408   3.026  0.00313 **
## w2            0.619396   0.145773   4.249 4.72e-05 ***
```

```
## wr           -0.013561   0.006089  -2.227  0.02813 *
## tw            0.169674   0.094458   1.796  0.07538 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 17.89 on 103 degrees of freedom
## Multiple R-squared:  0.7292, Adjusted R-squared:  0.7108
## F-statistic: 39.63 on 7 and 103 DF,  p-value: < 2.2e-16
```

```
model5 <- update(model4, ~ .-tw)
summary(model5)
```

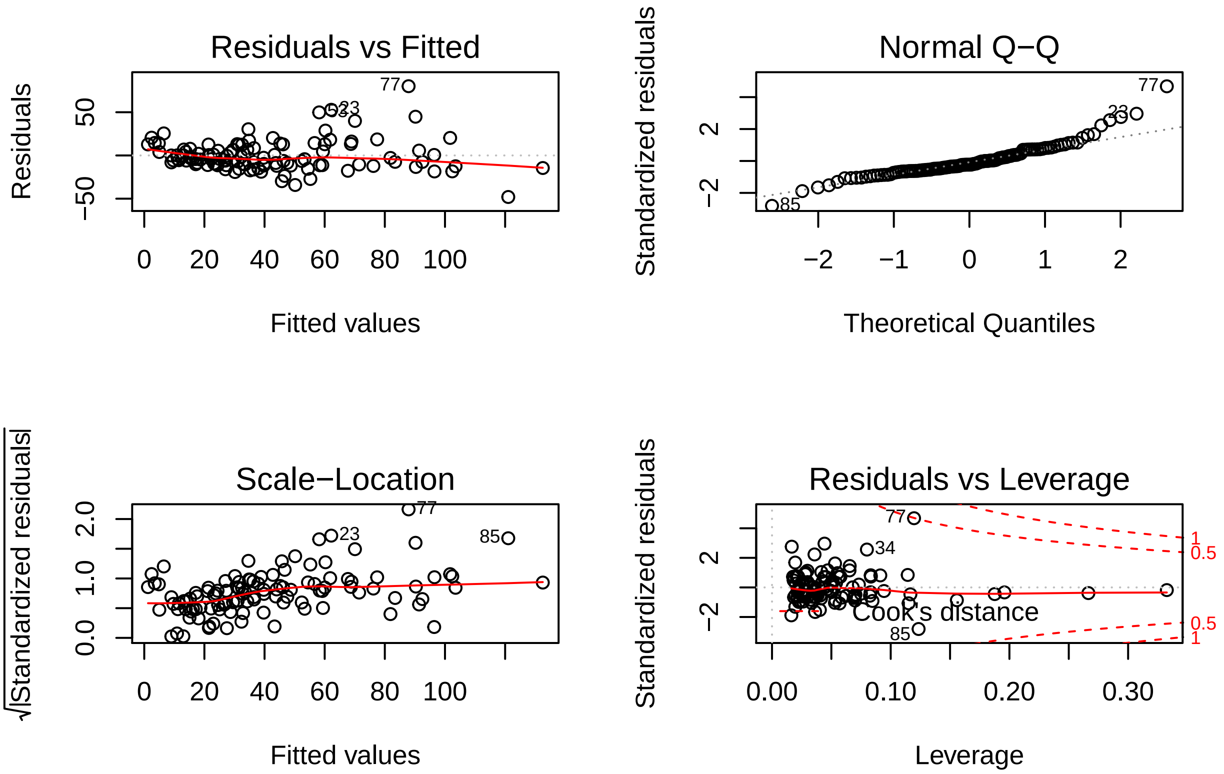
```
##
## Call:
## lm(formula = ozone ~ rad + temp + wind + t2 + w2 + wr)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -44.478 -10.735  -2.437   9.685  77.543
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept) 223.573855 107.618223   2.077 0.040221 *
## rad           0.173431   0.066398   2.612 0.010333 *
## temp         -5.197139   2.775039  -1.873 0.063902 .
## wind        -10.816032   2.736757  -3.952 0.000141 ***
## t2            0.043640   0.018112   2.410 0.017731 *
## w2            0.430059   0.101767   4.226 5.12e-05 ***
## wr           -0.009819   0.005783  -1.698 0.092507 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 18.08 on 104 degrees of freedom
## Multiple R-squared:  0.7208, Adjusted R-squared:  0.7047
## F-statistic: 44.74 on 6 and 104 DF,  p-value: < 2.2e-16
```

```
model6 <- update(model5, ~ .-wr)
summary(model6)
```

```
##
## Call:
## lm(formula = ozone ~ rad + temp + wind + t2 + w2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -48.044 -10.796  -4.138   8.131  80.098
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 291.16758  100.87723   2.886  0.00473 **
## rad           0.06586    0.02005   3.285  0.00139 **
## temp         -6.33955    2.71627  -2.334  0.02150 *
## wind        -13.39674    2.29623  -5.834 6.05e-08 ***
## t2            0.05102    0.01774   2.876  0.00488 **
## w2            0.46464    0.10060   4.619 1.10e-05 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 18.25 on 105 degrees of freedom
## Multiple R-squared:  0.713,  Adjusted R-squared:  0.6994
## F-statistic: 52.18 on 5 and 105 DF,  p-value: < 2.2e-16
```

```r
par(mfrow=c(2,2))
plot(model6)
```




```r
model7 <- lm(log(ozone) ~ rad+temp+wind+t2+w2+r2+wr+tr+tw+wtr)
summary(model7)
```

```
##
## Call:
## lm(formula = log(ozone) ~ rad + temp + wind + t2 + w2 + r2 +
##     wr + tr + tw + wtr)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.91943 -0.24169 -0.01742  0.28213  1.11802
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.803e+00  5.676e+00   0.494   0.6225
## rad          2.771e-02  1.529e-02   1.812   0.0729 .
## temp        -3.018e-02  1.178e-01  -0.256   0.7983
## wind        -9.812e-02  3.211e-01  -0.306   0.7605
## t2           6.034e-04  6.559e-04   0.920   0.3598
## w2           8.732e-03  4.021e-03   2.172   0.0322 *
```

```
## r2          -1.489e-05  7.043e-06  -2.114   0.0370 *
## wr          -2.001e-03  1.339e-03  -1.494   0.1382
## tr          -2.507e-04  2.056e-04  -1.219   0.2256
## tw          -1.985e-03  3.742e-03  -0.530   0.5971
## wtr          2.535e-05  1.805e-05   1.404   0.1634
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4877 on 100 degrees of freedom
## Multiple R-squared:  0.7116, Adjusted R-squared:  0.6827
## F-statistic: 24.67 on 10 and 100 DF,  p-value: < 2.2e-16
```

```r
model8 <- update(model7,~.-wtr)
summary(model8)
```

```
##
## Call:
## lm(formula = log(ozone) ~ rad + temp + wind + t2 + w2 + r2 +
##     wr + tr + tw)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.99582 -0.24838 -0.04271  0.32080  1.07835
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5.373e+00  5.398e+00   0.995   0.3219
## rad          7.896e-03  5.908e-03   1.336   0.1844
## temp        -6.230e-02  1.161e-01  -0.537   0.5927
## wind        -3.531e-01  2.660e-01  -1.327   0.1874
## t2           5.332e-04  6.571e-04   0.811   0.4191
## w2           8.340e-03  4.030e-03   2.069   0.0411 *
## r2          -1.624e-05  7.010e-06  -2.317   0.0225 *
## wr          -1.368e-04  1.731e-04  -0.790   0.4313
## tr           2.195e-05  6.783e-05   0.324   0.7469
## tw           1.784e-03  2.620e-03   0.681   0.4975
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4901 on 101 degrees of freedom
## Multiple R-squared:  0.7059, Adjusted R-squared:  0.6797
## F-statistic: 26.93 on 9 and 101 DF,  p-value: < 2.2e-16
```

```r
model9 <- update(model8,~.-tr)
summary(model9)
```

```
##
## Call:
## lm(formula = log(ozone) ~ rad + temp + wind + t2 + w2 + r2 +
##     wr + tw)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.96263 -0.24298 -0.04081  0.31953  1.09081
##
```

```
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5.516e+00  5.357e+00   1.030  0.30558
## rad          9.533e-03  3.036e-03   3.140  0.00221 **
## temp        -6.949e-02  1.134e-01  -0.613  0.54157
## wind        -3.574e-01  2.645e-01  -1.351  0.17966
## t2           6.029e-04  6.180e-04   0.976  0.33160
## w2           8.451e-03  3.998e-03   2.114  0.03697 *
## r2          -1.584e-05  6.865e-06  -2.307  0.02310 *
## wr          -1.517e-04  1.662e-04  -0.913  0.36341
## tw           1.846e-03  2.601e-03   0.710  0.47956
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4879 on 102 degrees of freedom
## Multiple R-squared:  0.7056, Adjusted R-squared:  0.6825
## F-statistic: 30.56 on 8 and 102 DF,  p-value: < 2.2e-16
```

```
model10 <- update(model9,~.-tw)
summary(model10)
```

```
##
## Call:
## lm(formula = log(ozone) ~ rad + temp + wind + t2 + w2 + r2 +
##     wr)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.89186 -0.26391 -0.03075  0.33076  1.09627
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.326e+00  2.907e+00   0.800  0.42544
## rad          8.875e-03  2.884e-03   3.077  0.00268 **
## temp        -9.290e-03  7.515e-02  -0.124  0.90185
## wind        -1.772e-01  7.366e-02  -2.405  0.01795 *
## t2           3.360e-04  4.892e-04   0.687  0.49375
## w2           6.389e-03  2.739e-03   2.333  0.02162 *
## r2          -1.515e-05  6.781e-06  -2.235  0.02761 *
## wr          -1.112e-04  1.557e-04  -0.714  0.47676
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4867 on 103 degrees of freedom
## Multiple R-squared:  0.7041, Adjusted R-squared:  0.684
## F-statistic: 35.02 on 7 and 103 DF,  p-value: < 2.2e-16
```

```
model11 <- update(model10,~.-t2)
summary(model11)
```

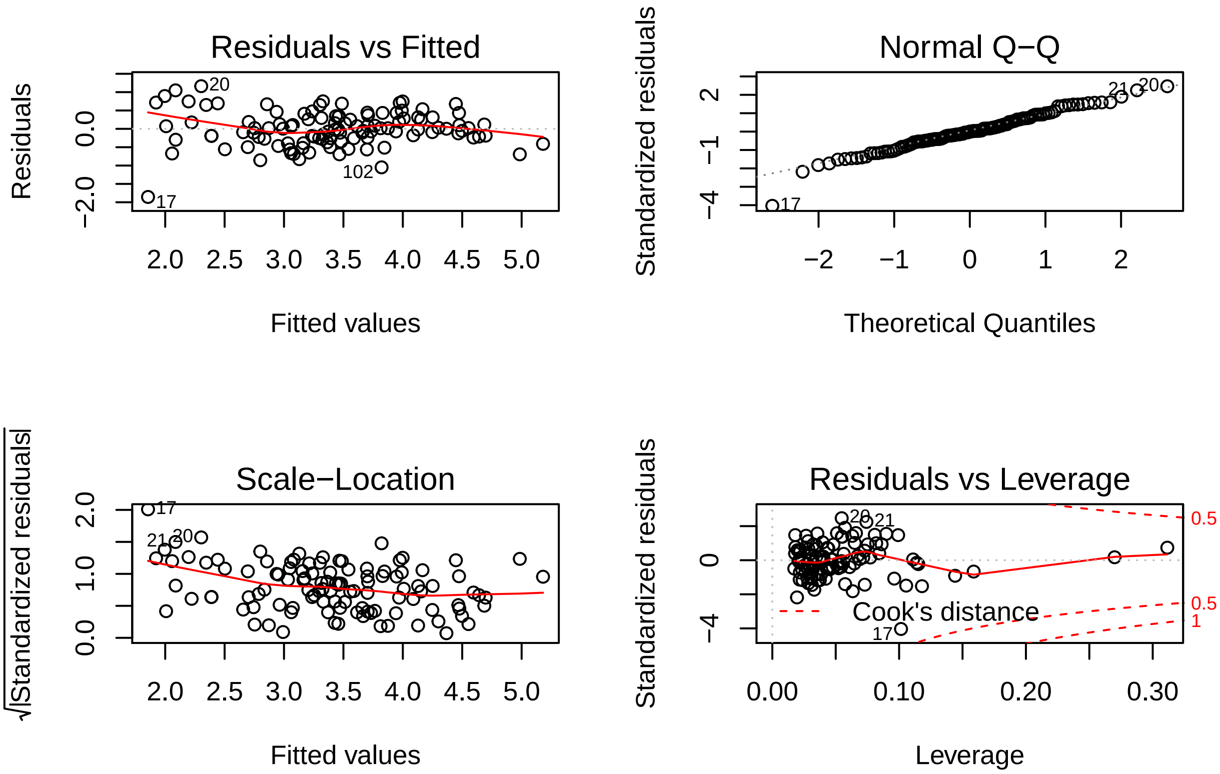
```
##
## Call:
## lm(formula = log(ozone) ~ rad + temp + wind + w2 + r2 + wr)
##
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -1.82031 -0.25479 -0.02779  0.33595  1.15024
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.989e-01  7.571e-01   0.527  0.59936
## rad          8.996e-03  2.871e-03   3.133  0.00225 **
## temp         4.214e-02  6.246e-03   6.746 8.79e-10 ***
## wind        -1.816e-01  7.320e-02  -2.481  0.01472 *
## w2           6.758e-03  2.679e-03   2.523  0.01316 *
## r2          -1.477e-05  6.740e-06  -2.191  0.03071 *
## wr          -1.368e-04  1.507e-04  -0.908  0.36615
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4855 on 104 degrees of freedom
## Multiple R-squared:  0.7028, Adjusted R-squared:  0.6856
## F-statistic: 40.98 on 6 and 104 DF,  p-value: < 2.2e-16
```

```
model12 <- update(model11,~.-wr)
summary(model12)
```

```
##
## Call:
## lm(formula = log(ozone) ~ rad + temp + wind + w2 + r2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.85551 -0.25578  0.00248  0.31349  1.16251
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.724e-01  6.350e-01   1.216 0.226543
## rad          7.466e-03  2.323e-03   3.215 0.001736 **
## temp         4.193e-02  6.237e-03   6.723 9.52e-10 ***
## wind        -2.211e-01  5.874e-02  -3.765 0.000275 ***
## w2           7.390e-03  2.585e-03   2.859 0.005126 **
## r2          -1.470e-05  6.734e-06  -2.183 0.031246 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4851 on 105 degrees of freedom
## Multiple R-squared:  0.7004, Adjusted R-squared:  0.6861
## F-statistic:  49.1 on 5 and 105 DF,  p-value: < 2.2e-16
```

```
plot(model12) # minimum adequate
```




```r
par(mfrow = c(1, 1))
detach(ozone.pollution)
```