Statistical Modelling

Dr Tahani Coolen-Maturi

Last updated 2023-09-28

Contents

Computer Practicals

Welcome to the Statistical Modelling module (MATH2697) on the 2nd year undergraduate programme at Durham University.

These pages will contain the computer practicals for this course. We will be using R in this module, so you will need to download R and RStudio if you are going to the do the practicals on your own computer.

Data Sets:

All data sets for this course are provided in the R package `sm2data`. To install the `sm2data` package, first you need to install the package `remotes`. **This needs to be done only once on your computer.**

```r
install.packages("remotes")
remotes::install_github("tmaturi/sm2data")
```

Alternatively, you can download the data files from Blackboard Ultra, then use

```r
datafile <-  read.table("enter_name_of_path/datafile.rda", header=TRUE)
```

NEXT:

You may want to start with the **Hello R!** page.

Hello R!

Install R and RStudio

- Download R
- Download RStudio

The following is a crash-course in R.

Packages:

- Not all R functions that you may want to use are by default available.
- For instance, if you wish to use the `cloud` function in order to create a 3D scatterplot, you need to load the package `lattice` via `library(lattice)`. This works only for packages which are already installed. R comes with a number of pre-installed packages (type `library()` to see them).
- If you need to install another package, then you can do this via `install.packages("packagename")`.

Reading, viewing, and saving data:

- Data are typically read in from files via `read.table`, or from R packages via `data`. For instance,

```r
#read.table("file.dat", header = TRUE)
data(cement, package="MASS")
```

- We can learn more about our data

```r
# gives the first six rows of the `cement` data frame.
head(cement)
```

```r
# gives the names of the variables included in the data frame.
names(cement)
```

```r
# gives the dimension of the  data frame.
dim(cement)
```

- Individual variables can be accessed through the `$` symbol

```
# extract the y column from the cement data set
cement$y
```

- Write `cement` data to file `test.dat`:

```
write.table(cement, file= "test.dat", quote=FALSE)
```

- Save your `R` work space: `save.image()` (creates file ".RData") or `save.image("filename.RData")`; and restore via `load("filename.RData")`.

Vector and matrix operations:

- Creating vectors: we can create the vector $a = (1, 2, 7)^T$, and the longer vector $b = (1, 2, 7, 2, 4, 6, 0, 1, 7)^T$ as

```
a<- c(1,2,7)
b<- c(a,2,4,6,0,1,7)
```

- Creating a matrix with three columns:

```
A<- matrix(b, ncol=3, byrow=TRUE)
# here, `byrow=TRUE` means that the matrix is filled row by row
```

- Picking the third element of a vector:

```
a[3]
# here: 7
```

- Picking the element in the third row and second column of a matrix:

```
A[3,2]
# here: 1
```

- Picking the first two rows of A:

```
A[1:2,]
```

- To attache the column a to the right of matrix A; similar `rbind` to attach rows.

```
cbind(A,a)
```

- One gets the transpose, inverse, determinant, and trace of A, respectively.

```
t(A)
solve(A)
det(A)
sum(diag(A))
```

- To obtain the eigenvalues and eigenvectors of a matrix A.

```
eigen(A)
```

- Use %*% for matrix multiplication and * for component-wise multiplication.

```
B <- matrix(1:12, ncol = 3, nrow = 4)
B %*% A
```

- *Ordering:*

```
# provides the ordered values of b
sort(b)
# to gets the order statistic of b
# i.e. the j-th element of  `order(b)` tells at which position
# the $j$-th smallest value is found in b.
order(b)
# Naturally, `sort(b) =  b[order(b)]`
b[order(b)]
```

Simple statistical operations:

- `mean(a)` and `sd(a)` give the mean and standard deviation of vector a, respectively.
- `range(a)` gives a vector which contains the smallest and the largest value of a.
- `colMeans(A)` and `rowMeans(A)` give the column and row means, respectively, of a matrix A.
- `hist(a)` gives a histogram of the distribution of a.
- `var(Z)` and `cor(Z)` give the variance and correlation matrix of a data matrix Z, respectively.
- `qnorm(0.99,0,1)` gives the 99% quantile of a $N(0,1)$ distribution, i.e. the quantity usually denoted by $z_{0.01}$.
- For variables `x` and `y` saved in a data frame `D`, a simple linear model of type $y = a + b \cdot x + \epsilon$ is fitted via `lm1 <- lm(y ~  x, data = D)`.

- A scatterplot with regression line is obtained via `plot(D$x, D$y); abline(lm1)`
- *Simulation:* `runif(n, a, b)` generates n random numbers uniformly distributed on $[a, b]$, and `rnorm(n, m, s)` generates n normally distributed random numbers with mean m and standard deviation s.

Basic programming:

- The **if/then** command performs an *action* if the *condition* is met. For instance,

```
a <- 4
if (a==0){ stop("invalid divisor")} else {10/a}
```

- A **for** loop repeats an *action* for all elements of a *set*. For instance,

```
for (i in 1:10){  cat('This is loop', i,  '\n') }
# will produce 10 rows of text which report the number of the loop. The
# string  '\n' is borrowed from the C language and means to start a new line.
```

- **Functions** allow to prepare some code which can be used later with different function arguments. Functions can have more than one argument, which are separated by commas. Default values can be given behind a `=` symbol. For instance

```
max1 <- function(a, b = 1) {
  result <-  max(a, b)
  return(result)
}
max1(0.5)
max1(0.5,0)
```

- The **apply** function allows to carry out some operation onto all rows or columns of a matrix. For instance, if `A` is a $n \times p$ matrix, then

```
apply(A,1, sum)
```

would give a $n \times 1$ vector which contains the sums over each row, and

```
apply(A,2, mean)
```

would give the column means.

Plotting and data visualization:

- `plot(x,y)` gives a 2D scatterplot of y vs. x.
- `points(x0,y0)` adds additional points to an existing 2D scatterplot.
- `segments(x[i], y[i], x[j], y[j])` draws a straight line from $(x_i, y_i)^T$ to $(x_j, y_j)^T$.
- `pairs(Z)` gives a matrix plot of all 2D combinations of all variables in Z.
- `cloud(z~x+y)` (package **lattice**) gives a 3D scatterplot of z vs. x and y. All three vectors need to have the same length.
- `persp(x,y,S)` gives a 3D surface plot, where x and y are vectors of length n, and S a $n \times n$ matrix (containing the values $S(x, y)$). Try `persp(1:10,1:10, matrix(1:100,10,10))` to understand what happens.
- `contour(x,y,S)` works as `persp` but gives a contour plot instead.

Practical 1

To start, create a new R script by clicking on the corresponding menu button (in the top left); and save it somewhere in your computer. You can now write all your code into this file, and then execute it in R using the menu button Run or by simple pressing **ctrl + enter**.

Exploration

Scallop data set

Load the data set `scallop` from the `sm2data` library. You can alternatively download the data set file from Blackboard Ultra.

```r
#  This needs to be done only once on your computer
install.packages("remotes")
remotes::install_github("tmaturi/sm2data")
# if you have downloaded the data file from Blackboard Ultra then use
# scallops <-  read.table("enter_name_of_path/scallops.dat", header=TRUE)
```

```r
library(sm2data)
data(scallops)
```

Learn more about the data:

```r
dim(scallops)
names(scallops)
# Note: y=log(tcatch)
scallops[1:6,]
head(scallops)
# Both commands above do the same, namely to display the first 6 rows:
```

Exercise 0.1. Assign estimates of mean and variance to objects `m` and `Sigma`, respectively. Verify that `Sigma` is positive definite by looking at its eigenvalues (use `eigen(...)`).

Click for solution

```
## SOLUTION
# Compute the mean and display it.
m <- colMeans(scallops[,c("long","lat")])
m
```

```
##      long       lat
## -72.73215  39.91798
```

```
# Compute the variance and display it.
Sigma<- var(scallops[,c("long","lat")])
Sigma
```

```
##           long       lat
## long 0.2636210 0.2531269
## lat  0.2531269 0.3699811
```

```
# Compute the eigenvalues and eigenvectors.
eVs = eigen(Sigma)

eVs$values
```

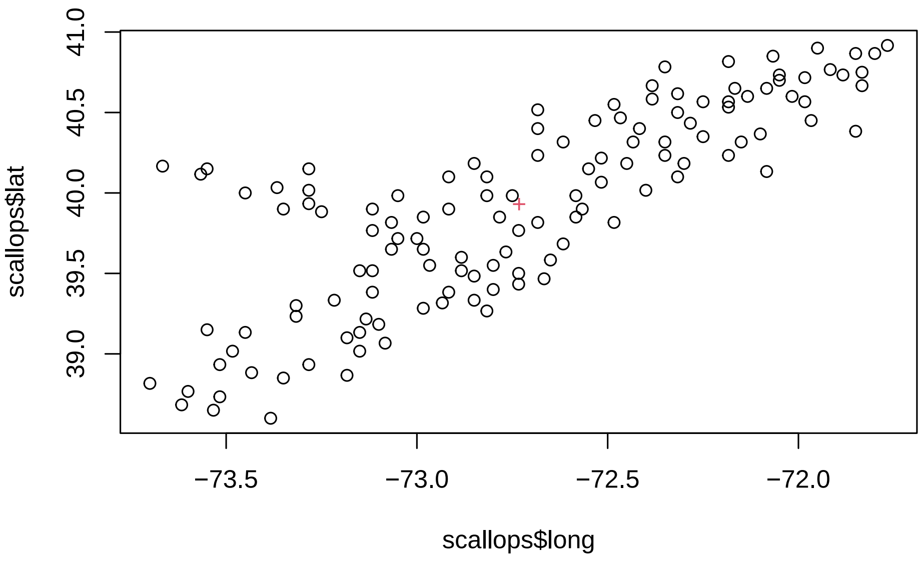
```
## [1] 0.57545402 0.05814811
```

```
# Note that all eigenvalues are > 0, meaning Sigma is positive definite.
```

Exercise 0.2. Make a scatterplot of the longitude and latitude data. Add the mean `m` to the plot. [Hint: Use the commants points for adding the mean.]

Click for solution

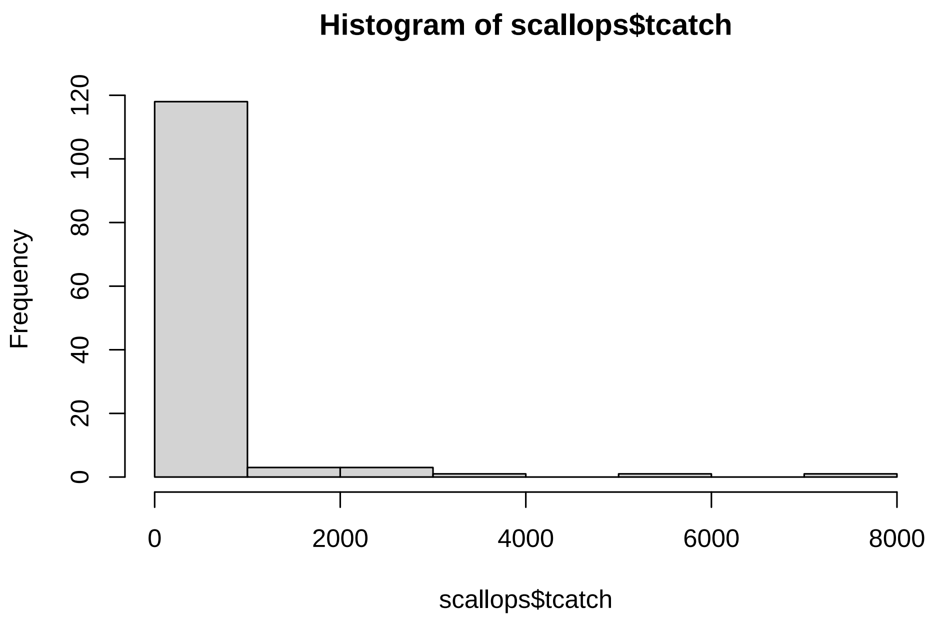
```
## SOLUTION
# Plot them.
plot(scallops$long, scallops$lat)

# The mean can then be added to the scatterplot via
points(m[1],m[2], col=2, pch="+")
```



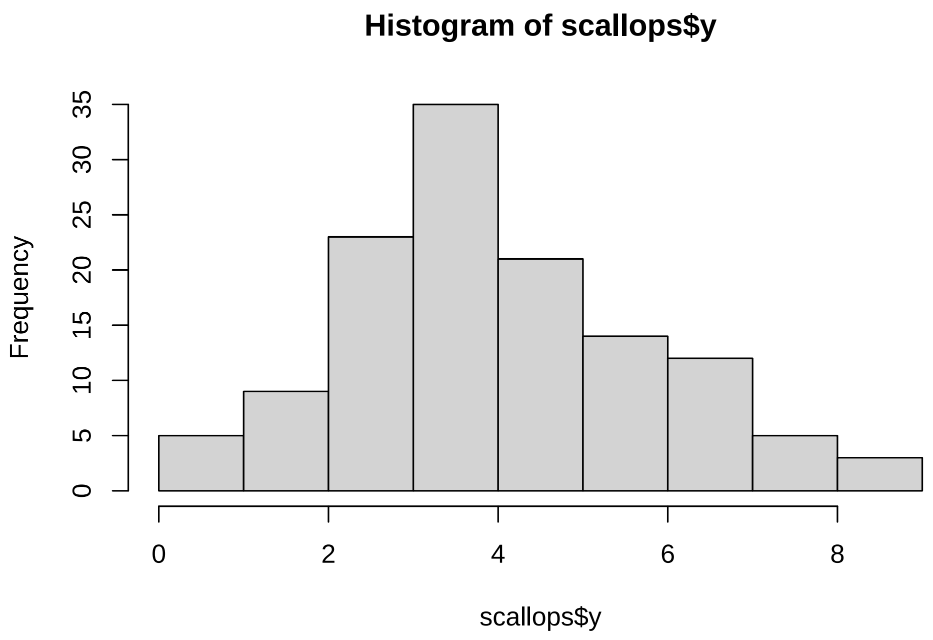
Exercise 0.3. Produce histograms of the abundances `tcatch`, and the log-abundances `y=log(tcatch)`. Give a reason why the latter appears more suitable for the use in statistical analysis.

Click for solution

```
## SOLUTION
# Plot the histograms.
hist(scallops$tcatch)
```



```r
hist(scallops$y)
```



```r
# We see that the distribution of log-abundances is far less skewed, and appears
# to be more 'normal'. Normality of the response is a standard assumption of
# regression models.
```

Exercise 0.4. We wish to fit a bivariate normal distribution with mean `m` and variance `Sigma` to the scallop locations. Create a matrix of bivariate normal density values (with mean `m` and variance `Sigma`) on a suitable two–dimensional grid which covers the longitudes and latitudes in the sample. *Hint*: note we cannot use independence here to multiply the densities.

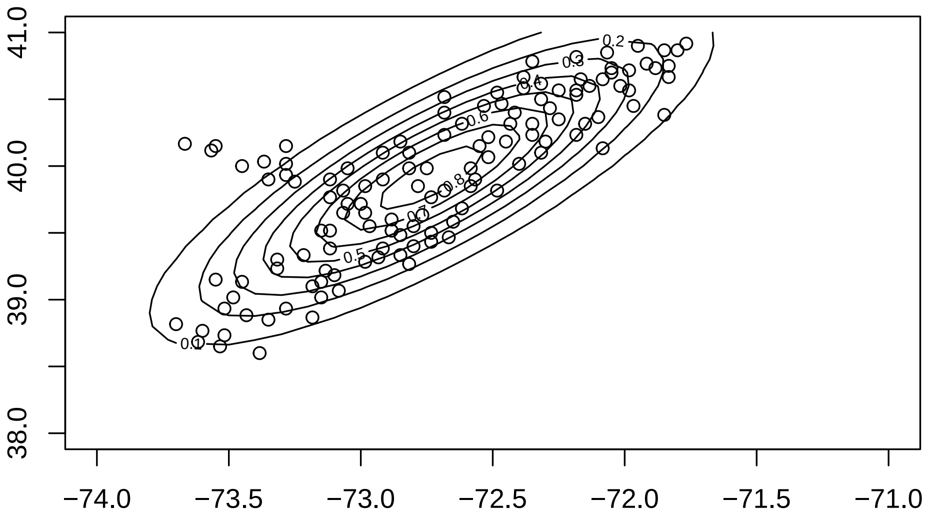
Click for solution

```
## SOLUTION
x<- seq(-74,-71, length=31)
y<- seq(38, 41, length=31)

dens <- matrix(0,31,31)

for (i in 1:31){
  for (j in 1:31){
   dens[i,j]<- 1/(2*pi*sqrt(det(Sigma)))*exp(-1/2* c(x[i]-m[1], y[j]-m[2])%*%solve(Sigma)%*%c(x[i
 }
}

# That is, we create two vectors (x and y) of length 31 and
# evaluate the bivariate density function at each pair.
# Note: the value 31 is arbitrary to create a smooth contour
# plot; try a smaller value, e.g. 15 or 20 and see what happens.
```

Exercise 0.5. Use `contour()` to create a contour plot of the fitted density, and `points()` in order to add the original data to it. Comment on the adequacy of the fitted BVN.

Click for solution

```
## SOLUTION
par(mfrow=c(1,1))
contour(x, y, dens)
points(scallops$long, scallops$lat)
```



```
# The BNV seems not to be perfectly adequate as it cannot account for (and gets biased
# the small branch in the NW.
# Relative to the fitted distribution, some points in the NW and the SE appear to be o
# but due to the reason above one has to be careful with this interpretation.
```

Mahalanobis distances

Engine data set

We consider the 46×3 data frame `engine` in the `sm2data` library. If you haven't installed library `sm2data` yet please install it now by running the the following commands in the RStudio console:

```
install.packages("remotes")
remotes::install_github("tmaturi/sm2data")
```

Load the data into `R`. The data consists of carbon monoxide (`CO`), hydrocarbon (`HC`) and nitrogen oxide (`NOX`) measurements (read the accompanying R Documentation: `?engine`).

```
library(sm2data)
data(engine)
```

Exercise 0.6. Calculate the mean $\hat{m}$ and the sample variance matrix $\hat{S}$ of the full data set, and save them into objects `M` and `S`, respectively. Write down $\hat{m}$ and $\hat{S}$, rounded to two decimal places. *Hint:* `R` can do the rounding for you, through function `round(...)`.

Click for solution

```
## SOLUTION
M    <- colMeans(engine)
S    <- var(engine)
round(M, digits=2)
```

```
##   CO   HC  NOX
## 7.96 0.55 1.33
```

```
round(S, digits=2)
```

```
##        CO    HC   NOX
## CO  27.67  0.80 -1.75
## HC   0.80  0.03 -0.05
## NOX -1.75 -0.05  0.23
```

Exercise 0.7. Produce a `pairs` plot of the data. Then produce a scatter plot of `CO` and `HC`. Which data points seem to deviate strongly from the other sample values? To identify these suspected points – the outliers – use the command

```
plot(engine$CO, engine$HC)
identify(engine$CO, engine$HC)
```

This will temporarily freeze the RStudio console. Now, move the mouse cursor to the plot. As you will see a cross will appear. Simply click on the points you think are outliers and when you are happy with your choice press the **Esc** button. The sample numbers will appear in the console. Repeat this for the pairs of variables (CO, NOX) and (HC,NOX). Write down the case numbers of the identified observations.

Click for solution

```
## SOLUTION
pairs(engine)

plot(engine$CO, engine$HC)
identify(engine$CO, engine$HC)

plot(engine$CO, engine$NOX)
identify(engine$CO, engine$NOX)

plot(engine$HC, engine$NOX)
identify(engine$HC, engine$NOX)

# Observations 34,35, and 39  appear to be possible outliers.

# A small number of Windows users may experience issues with the
# identify function; if this is the case, then run win.graph()
# before calling the scatterplot. Alternatively, use R GUI (basic
# R, not RStudio) to perform this exercise.
```

Exercise 0.8. Compute the squared Mahalanobis distances to the mean for all observations, and write down d_M^2 for the observation with case number 5. For the latter, also compute the Mahalanobis distance using its explicit mathematical formula, and verify whether the results agree.

Click for solution

```
## SOLUTION
d <- mahalanobis(engine, M, S)
d[5]
```

```
##        5
## 2.385444
```

```
eng5 <- as.numeric(engine[5,])
(eng5-M)%*%solve(S)%*%(eng5-M)
```

```
##          [,1]
## [1,] 2.385444
```

```
## Note: eng5-m is a vector (not a matrix), and so R will try to align
# its orientation by itself so that the matrix multiplication works.
```

Exercise 0.9. A more formal way of detecting outliers is by testing the hypothesis H_0 : *A point is not an outlier* vs. the alternative H_1 : *A point is an outlier* for a given level of statistical significance α. In practice in R, we reject H_0 if the Mahalanobis distance of a specific data point is greater than the $100\% \times (1-\alpha)$-percentile of a chi-square distribution with q degrees of freedom (where q is the number of variables). For all 46 cases, carry out a statistical test of this hypothesis. Give the case numbers for which this null hypothesis is rejected at the 5% and 2.5% level of significance, and
compare the result with your intuitive solution of **Exercise** 0.7. Write down a summary of your results. Hint: See `?qchisq`.

Click for solution

```
## SOLUTION
# 5% of significance
which(d>   qchisq(0.95,3))
# 30 34 35 39
#  Observation 30 might be surprising.

# 2.5% of significance
which(d>   qchisq(0.975,3))
# 34 35 39
```

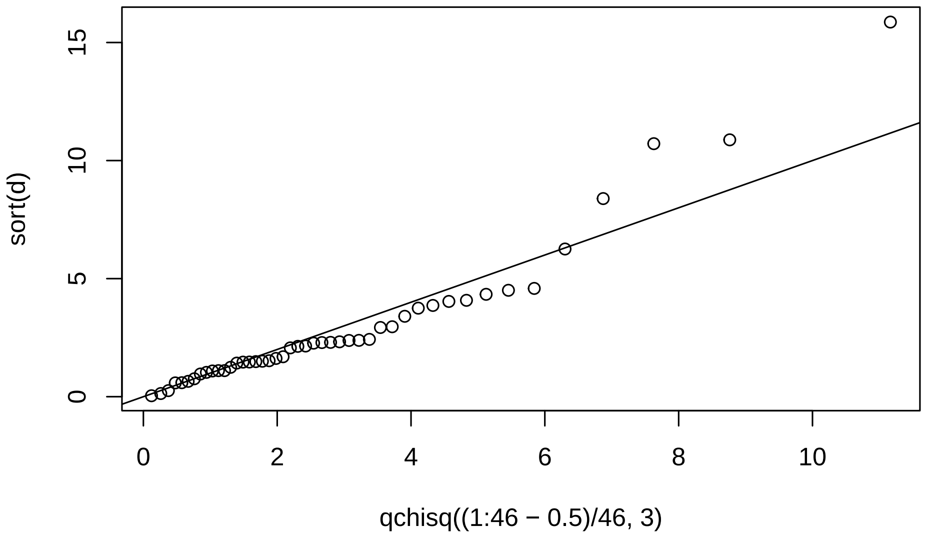
Exercise 0.10. Which assumption do you implicitly make in applying this outlier test based on the chi-square distribution? Use an adequate statistical tool to check the validity of your assumption. Summarize your results.

Click for solution

```
## SOLUTION
#  We assumed that the data are tri-variate normal.
# We check this through a chi^-quantile plot for the Mahalanobis distances

dim(engine)
```

```
## [1] 46  3
```

```
plot(qchisq( (1:46-0.5)/46,3),sort(d))

abline(a=0,b=1)
```



```
# clearly some deviation from the straight line,
# especially in the central part (observations are too small)
# and the boundary (observations are too large ==> outliers).
# the d_i  have also some discrete (stepwise) character.
```

Exercise 0.11. Repeat the analysis from **Exercise** 0.9, this time using the $100\% \times (1-\alpha/n)$-percentile, where n denotes sample size. This is the Bonferroni correction for the *multiple testing problem*.

Click for solution

```
## SOLUTION
# After Bonferroni correction:
which(d>   qchisq(1-0.05/46,3))
```

```
## named integer(0)
```

```
# none of the observations are identified as outliers
```

Practical 2

To start, create a new R script by clicking on the corresponding menu button (in the top left); and save it somewhere in your computer. You can now write all your code into this file, and then execute it in R using the menu button Run or by simple pressing **ctrl + enter**.

Normal equations

Engine data set

Load the data frame `engine` from the `sm2data` library. Suppose we wish to study the relation between the emissions of nitrogen monoxide and those of hydrocarbons and carbon monoxide. Consider the following linear model for this relationship:

$$\texttt{NOX} = \beta_1 + \beta_2\texttt{CO} + \beta_3\texttt{HC} + \epsilon. \tag{1}$$

```
# install.packages("remotes") #you need to do this once
# remotes::install_github("tmaturi/sm2data") #you need to do this once
library(sm2data)
data(engine)
?engine
```

Learn more about the data

```
head(engine)
```

```
##      CO   HC  NOX
## 1  5.01 0.50 1.28
## 2  8.60 0.46 1.17
## 3  4.95 0.41 1.16
## 4  7.51 0.44 1.08
## 5 14.59 0.72 0.60
## 6 11.53 0.83 1.32
```

```r
dim(engine)
```

```
## [1] 46  3
```

```r
names(engine)
```

```
## [1] "CO"  "HC"  "NOX"
```

Exercise 0.12. Construct the design matrix `X` and the matrix of response values `Y`. *Hint:* Use `X<-  as.matrix(cbind(..., ...))`, where the first `...` is a vector of 1's of appropriate length, and the second `...` corresponds to the first two columns of the `engine` data frame.

Click for solution

```r
## SOLUTION
X<-  as.matrix(cbind(rep(1,46), engine[,1:2]) )
Y<-  engine[,3]
```

Exercise 0.13. Solve the normal equations, that is, compute $(\mathtt{X}^T\mathtt{X})^{-1}\mathtt{X}^T\mathtt{Y}$ to obtain the estimated parameters.

Click for solution

```r
## SOLUTION
betahat<-solve(t(X)%*%X)  %*% t(X) %*%Y
betahat
```

```
##                    [,1]
## rep(1, 46)  1.54354381
## CO         -0.08876657
## HC          0.89127815
```

Exercise 0.14. Fit the same linear model in Eq. (1) using the built–in R function`lm`, with `NOX` as the response and `CO` and `HC` as predictors. Compare your results with the results from **Exercise** 0.13. *Hint:* Use `lm(...~...+..., data=engine)`, where `...` are the names of the involved variables.

Click for solution

```r
## SOLUTION
fit<-lm(NOX~CO+HC, data=engine)
fit
```

```
##
## Call:
## lm(formula = NOX ~ CO + HC, data = engine)
##
## Coefficients:
## (Intercept)           CO           HC
##     1.54354     -0.08877      0.89128
```

Inference and prediction

Engine data set

Exercise 0.15. Using the fitted linear model in Exercise 0.14. Save the vector of estimated regression parameters $\widehat{\beta}$ into a vector with name `beta`, and the estimated error standard deviation s into an object `s`. Then (using an adequate formula from the lecture notes), produce the estimated variance matrix of $\widehat{\beta}$ and save it into an object of name `Sigma`. From this, extract $SE(\hat{\beta}_2)$ and $SE(\hat{\beta}_3)$. Report these values here, and check via the linear model `summary` whether they are correct.

Click for solution

```
## SOLUTION
beta<- fit$coef
s<-summary(fit)$sigma
Sigma<- s^2*summary(fit)$cov.unscaled
Sigma
```

```
##              (Intercept)            CO          HC
## (Intercept)  0.063601963  0.0040631790 -0.16893795
## CO           0.004063179  0.0005361271 -0.01509929
## HC          -0.168937953 -0.0150992864  0.52403670
```

```
# SE(hatbeta1):
sqrt(0.0005361271)
```

```
## [1] 0.02315442
```

```
# SE(hatbeta2):
sqrt(0.52403670)
```

```
## [1] 0.7239038
```

```
# we can obtain the same results using
s * sqrt(diag(solve(t(X)%*%X)))
```

```
## rep(1, 46)          CO          HC
## 0.25219430 0.02315442 0.72390379
```

```
# Now compare that with the model summary
summary(fit)
```

```
##
## Call:
## lm(formula = NOX ~ CO + HC, data = engine)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.55611 -0.26214 -0.06695  0.19147  1.31380
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.54354    0.25219   6.120 2.44e-07 ***
## CO          -0.08877    0.02315  -3.834 0.000407 ***
## HC           0.89128    0.72390   1.231 0.224936
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3548 on 43 degrees of freedom
## Multiple R-squared:  0.4875, Adjusted R-squared:  0.4636
## F-statistic: 20.45 on 2 and 43 DF,  p-value: 5.745e-07
```

Exercise 0.16. Produce 95% confidence intervals $\hat{\beta}_j \pm t_{n-p,0.025} \times SE(\hat{\beta}_j)$ for β_j, $j = 2, 3$. *Hint:* For the quantile, use `qt(0.975, df)`, where `df`$= n - p$. Check for correctness of your result by applying the function `confint` used in lectures.

Click for solution

```
## SOLUTION
beta[2] +c(-1,1)*0.02315442 *qt(0.975, 43)
```

```
## [1] -0.13546191 -0.04207123
```

```
beta[3] +c(-1,1)*0.7239038 *qt(0.975, 43)
```

```
## [1] -0.568613  2.351169
```

```
confint(fit,2)
```

```
##         2.5 %      97.5 %
## CO -0.1354619 -0.04207124
```

```r
confint(fit,3)
```

```
##        2.5 %   97.5 %
## HC -0.568613 2.351169
```

Exercise 0.17. Carry out a statistical test of $H_0 : \beta_2 = 0$ vs. $H_1 : \beta_2 \neq 0$ by:

a. Considering the corresponding p–value in the `summary` output,
b. Computing explicitly the test statistic and comparing it with the critical value,
c. Considering the previously computed confidence interval.

Summarize your conclusion.

Click for solution

```r
## SOLUTION (a)
summary(fit)
```

```
##
## Call:
## lm(formula = NOX ~ CO + HC, data = engine)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.55611 -0.26214 -0.06695  0.19147  1.31380
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.54354    0.25219   6.120 2.44e-07 ***
## CO          -0.08877    0.02315  -3.834 0.000407 ***
## HC           0.89128    0.72390   1.231 0.224936
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3548 on 43 degrees of freedom
## Multiple R-squared:  0.4875, Adjusted R-squared:  0.4636
## F-statistic: 20.45 on 2 and 43 DF,  p-value: 5.745e-07
```

```r
# via p-value:   p=0.000407<0.05, so reject H0
```

```r
## SOLUTION (b)
# via test statistic
t = abs(-0.08877/0.02315)
t
```

```
## [1] 3.834557
```

```r
qt(0.975, 43)
```

```
## [1] 2.016692
```

```r
# 2.016692 < 3.834557, so reject H0
```

```r
## SOLUTION (c)
confint(fit,2)
```

```
##          2.5 %      97.5 %
## CO -0.1354619 -0.04207124
```

```r
# via CI: [-0.13546191, -0.04207123]  does not contain 0 so reject H0
```

Exercise 0.18. Carry out a statistical test of $H_0 : \beta_3 = 1$ vs. $H_1 : \beta_3 \neq 1$ by:

a. Computing explicitly the test statistic and comparing it with the critical value,
b. Considering the previously computed confidence interval.

Summarize your conclusion.

Click for solution

```r
## SOLUTION (a)
# via test statistic
t = abs((1-0.89128)/ 0.72390)
t
```

```
## [1] 0.1501865
```

```r
# 0.1501865 < 2.016692 so do not reject H0
```

```r
## SOLUTION (b)
confint(fit,3)
```

```
##         2.5 %   97.5 %
## HC -0.568613 2.351169
```

```r
# via CI: [-0.568613,  2.351169]  does contain 1 so do not reject H0.
```

Exercise 0.19. Since we know that the sampling distribution of s^2 is given by $s^2 \sim \sigma^2 \frac{\chi^2_{n-p}}{n-p}$, we are able to devise statistical tests for σ^2. So, assume for instance that we are interested in testing

$$H_0 : \sigma^2 \leq \sigma_0^2 \text{ versus } H_1 : \sigma^2 > \sigma_0^2.$$

Then we can define a test statistic, say V, as

$$V = s^2/\sigma_0^2 \times (n - p)$$

so that H_0 would be rejected if $V > \chi^2_{n-p,\alpha}$. Write a function `test.s` which takes the fitted model `lmobject` and the value `sigma0` as arguments, and produces the corresponding p-value as output. Apply your function using $\sigma_0 = 0.3$ and draw your conclusion for a significance level of 5%.

Click for solution

```
## SOLUTION
test.s<-function(lmobject, sigma0){
    n<-length(lmobject$res)
    p<-length(lmobject$coef)
    s<-summary(lmobject)$sigma
    V= s^2/sigma0^2*(n-p)
    pvalue<- 1-pchisq(V, n-p)
    return(pvalue)
}

summary(fit)
```

```
##
## Call:
## lm(formula = NOX ~ CO + HC, data = engine)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.55611 -0.26214 -0.06695  0.19147  1.31380
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.54354    0.25219   6.120 2.44e-07 ***
## CO          -0.08877    0.02315  -3.834 0.000407 ***
## HC           0.89128    0.72390   1.231 0.224936
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3548 on 43 degrees of freedom
## Multiple R-squared:  0.4875, Adjusted R-squared:  0.4636
## F-statistic: 20.45 on 2 and 43 DF,  p-value: 5.745e-07
```

```
test.s(fit,0.30)
```

```
## [1] 0.04300668
```

```
# H0 is (just) rejected, so we conclude sigma>0.3
```

Exercise 0.20. From a light–duty engine, one measures 12.2 g carbon monoxide and 0.4g hydrocarbons emitted per mile of usage. Use your fitted model to:

a. Predict the nitrogen oxide emitted per mile (g).
b. Obtain a 95% confidence interval for the expected emission.
c. Obtain a 95% prediction interval for the actual emission.

Hint: You can use commands of type

`predict(fit, newdata=data.frame("CO"=..., "HC"=...), interval=...)`

with `interval` taking the value `confidence` or `prediction`, to solve this question.

Click for solution

```
## SOLUTION (a)
predict(fit, newdata=data.frame("CO"=12.2, "HC"=0.4))
```

```
##         1
## 0.8171029
```

```
## SOLUTION (b)
predict(fit, newdata=data.frame("CO"=12.2, "HC"=0.4), interval="confidence")
```

```
##         fit       lwr      upr
## 1 0.8171029 0.3947171 1.239489
```

```
## SOLUTION (c)
predict(fit, newdata=data.frame("CO"=12.2, "HC"=0.4), interval="prediction")
```

```
##         fit         lwr      upr
## 1 0.8171029 -0.01373135 1.647937
```

Alternatively, of course, you may attempt a manual implementation.

Click for solution

```
## SOLUTION
# (a): prediction
x0<-c(1, 12.2,0.4)
y0hat<-as.numeric(x0%*%fit$coef)
y0hat
```

```
## [1] 0.8171029
```

```
# (b): CI
y0hat+c(-1,1)*qt(0.975, 43)*s*sqrt(as.numeric(x0%*%summary(fit)$cov.unscaled%*%x0))
```

```
## [1] 0.3947171 1.2394888
```

```
# (c): PI
y0hat+c(-1,1)*qt(0.975, 43)*s*sqrt(as.numeric(1+x0%*%summary(fit)$cov.unscaled%*%x0))
```

```
## [1] -0.01373135  1.64793720
```

Practical 3

To start, create a new R script by clicking on the corresponding menu button (in the top left); and save it somewhere in your computer. You can now write all your code into this file, and then execute it in R using the menu button Run or by simple pressing **ctrl + enter**.

Factor coding

Insects data set

The data below come from an experiment reported in *Beall (1942), Transformation of data from entomological field experiments, Biometrika.* Six different insect sprays were each applied to 12 plots and in each case the number of tobacco hornworms. Spray is a *factor*, with *levels* $A, \dots, F$ and $r = 12$ replicates per factor level. Note: we discussed this insects data set in the lecture.

Spray \ Number of insects												
A	10	7	20	14	14	12	10	23	17	20	14	13
B	11	17	21	11	16	14	17	17	19	21	7	13
C	0	1	7	2	3	1	2	1	3	0	1	4
D	3	5	12	6	4	3	5	5	5	5	2	4
E	3	5	3	5	3	6	1	1	3	2	6	4
F	11	9	15	22	15	16	13	10	26	26	24	13

Exercise 0.21. As in the lecture, carry out the coding of the spray factor, and fit the linear model. Display also the model `summary` and the design matrix (using, for instance, function `model.matrix`), and make sure you understand what you see. Which spray type is used as reference category?

Click for solution

```
## SOLUTION
# Our previous code for the insects data
insects0 <-
  c(
    10,7,20,14,14,12,10,23,17,20,14,13,
    11,17,21,11,16,14,17,17,19,21,7,13,
    0,1,7,2,3,1,2,1,3,0,1,4,
    3,5,12,6,4,3,5,5,5,5,2,4,
    3,5,3,5,3,6,1,1,3,2,6,4,
    11,9,15,22,15,16,13,10,26,26,24,13
  )

insects= data.frame("spray"= c(rep("A",12), rep("B",12), rep("C",12),rep("D",12), rep(
insects$spray = as.factor(insects$spray)

# Linear model
fit1.insects <- lm(insects ~ spray, data= insects)
summary(fit1.insects)
```

```
##
## Call:
## lm(formula = insects ~ spray, data = insects)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -8.333 -1.958 -0.500  1.667  9.333
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  14.5000     1.1322  12.807  < 2e-16 ***
## sprayB        0.8333     1.6011   0.520    0.604
## sprayC      -12.4167     1.6011  -7.755 7.27e-11 ***
## sprayD       -9.5833     1.6011  -5.985 9.82e-08 ***
## sprayE      -11.0000     1.6011  -6.870 2.75e-09 ***
## sprayF        2.1667     1.6011   1.353    0.181
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.922 on 66 degrees of freedom
## Multiple R-squared:  0.7244, Adjusted R-squared:  0.7036
## F-statistic:  34.7 on 5 and 66 DF,  p-value: < 2.2e-16
```

```
model.matrix(fit1.insects)
```

```
##    (Intercept) sprayB sprayC sprayD sprayE sprayF
## 1            1      0      0      0      0      0
## 2                1      0      0      0      0      0
## 3                1      0      0      0      0      0
## 4                1      0      0      0      0      0
## 5                1      0      0      0      0      0
## 6                1      0      0      0      0      0
## 7                1      0      0      0      0      0
## 8                1      0      0      0      0      0
## 9                1      0      0      0      0      0
## 10               1      0      0      0      0      0
## 11               1      0      0      0      0      0
## 12               1      0      0      0      0      0
## 13               1      1      0      0      0      0
## 14               1      1      0      0      0      0
## 15               1      1      0      0      0      0
## 16               1      1      0      0      0      0
## 17               1      1      0      0      0      0
## 18               1      1      0      0      0      0
## 19               1      1      0      0      0      0
## 20               1      1      0      0      0      0
## 21               1      1      0      0      0      0
## 22               1      1      0      0      0      0
## 23               1      1      0      0      0      0
## 24               1      1      0      0      0      0
## 25               1      0      1      0      0      0
## 26               1      0      1      0      0      0
## 27               1      0      1      0      0      0
## 28               1      0      1      0      0      0
## 29               1      0      1      0      0      0
## 30               1      0      1      0      0      0
## 31               1      0      1      0      0      0
## 32               1      0      1      0      0      0
## 33               1      0      1      0      0      0
## 34               1      0      1      0      0      0
## 35               1      0      1      0      0      0
## 36               1      0      1      0      0      0
## 37               1      0      0      1      0      0
## 38               1      0      0      1      0      0
## 39               1      0      0      1      0      0
## 40               1      0      0      1      0      0
## 41               1      0      0      1      0      0
## 42               1      0      0      1      0      0
## 43               1      0      0      1      0      0
## 44               1      0      0      1      0      0
## 45               1      0      0      1      0      0
## 46               1      0      0      1      0      0
## 47               1      0      0      1      0      0
## 48           1      0      0      1      0      0
## 49           1      0      0      0      1      0
## 50           1      0      0      0      1      0
## 51           1      0      0      0      1      0
## 52           1      0      0      0      1      0
## 53           1      0      0      0      1      0
## 54           1      0      0      0      1      0
## 55           1      0      0      0      1      0
## 56           1      0      0      0      1      0
## 57           1      0      0      0      1      0
## 58           1      0      0      0      1      0
## 59           1      0      0      0      1      0
## 60           1      0      0      0      1      0
## 61           1      0      0      0      0      1
## 62           1      0      0      0      0      1
## 63           1      0      0      0      0      1
## 64           1      0      0      0      0      1
## 65           1      0      0      0      0      1
## 66           1      0      0      0      0      1
## 67           1      0      0      0      0      1
## 68           1      0      0      0      0      1
## 69           1      0      0      0      0      1
## 70           1      0      0      0      0      1
## 71           1      0      0      0      0      1
## 72           1      0      0      0      0      1
## attr(,"assign")
## [1] 0 1 1 1 1 1
## attr(,"contrasts")
## attr(,"contrasts")$spray
## [1] "contr.treatment"
```

```
# Clearly, spray type A is the reference category
```

Exercise 0.22. Assume now that spray type F is to be used as reference category (this may be of interest, for instance, if this is the spray usually applied by the farmer). You can achieve this in R by defining a new factor

```
 insects$spray2<- relevel(insects$spray, ref = "F")
```

and then repeat the linear model fit using `spray2`. Do this, and also inspect again the `summary` and the `model.matrix`.

Click for solution

```
## SOLUTION
# If we want to use category F as reference category, we have to do the following:
```

```
insects$spray2<- relevel(insects$spray, ref = "F")

fit2.insects<- lm(insects ~ spray2, data= insects)
summary(fit2.insects)
```

```
##
## Call:
## lm(formula = insects ~ spray2, data = insects)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -8.333 -1.958 -0.500  1.667  9.333
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   16.667      1.132  14.721  < 2e-16 ***
## spray2A       -2.167      1.601  -1.353    0.181
## spray2B       -1.333      1.601  -0.833    0.408
## spray2C      -14.583      1.601  -9.108 2.79e-13 ***
## spray2D      -11.750      1.601  -7.339 4.04e-10 ***
## spray2E      -13.167      1.601  -8.223 1.05e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.922 on 66 degrees of freedom
## Multiple R-squared:  0.7244, Adjusted R-squared:  0.7036
## F-statistic:  34.7 on 5 and 66 DF,  p-value: < 2.2e-16
```

```
model.matrix(fit2.insects)
```

```
##    (Intercept) spray2A spray2B spray2C spray2D spray2E
## 1            1       1       0       0       0       0
## 2            1       1       0       0       0       0
## 3            1       1       0       0       0       0
## 4            1       1       0       0       0       0
## 5            1       1       0       0       0       0
## 6            1       1       0       0       0       0
## 7            1       1       0       0       0       0
## 8            1       1       0       0       0       0
## 9            1       1       0       0       0       0
## 10           1       1       0       0       0       0
## 11           1       1       0       0       0       0
## 12           1       1       0       0       0       0
## 13           1       0       1       0       0       0
## 14           1       0       1       0       0       0
## 15           1       0       1       0       0       0
## 16          1     0     1     0     0     0
## 17          1     0     1     0     0     0
## 18          1     0     1     0     0     0
## 19          1     0     1     0     0     0
## 20          1     0     1     0     0     0
## 21          1     0     1     0     0     0
## 22          1     0     1     0     0     0
## 23          1     0     1     0     0     0
## 24          1     0     1     0     0     0
## 25          1     0     0     1     0     0
## 26          1     0     0     1     0     0
## 27          1     0     0     1     0     0
## 28          1     0     0     1     0     0
## 29          1     0     0     1     0     0
## 30          1     0     0     1     0     0
## 31          1     0     0     1     0     0
## 32          1     0     0     1     0     0
## 33          1     0     0     1     0     0
## 34          1     0     0     1     0     0
## 35          1     0     0     1     0     0
## 36          1     0     0     1     0     0
## 37          1     0     0     0     1     0
## 38          1     0     0     0     1     0
## 39          1     0     0     0     1     0
## 40          1     0     0     0     1     0
## 41          1     0     0     0     1     0
## 42          1     0     0     0     1     0
## 43          1     0     0     0     1     0
## 44          1     0     0     0     1     0
## 45          1     0     0     0     1     0
## 46          1     0     0     0     1     0
## 47          1     0     0     0     1     0
## 48          1     0     0     0     1     0
## 49          1     0     0     0     0     1
## 50          1     0     0     0     0     1
## 51          1     0     0     0     0     1
## 52          1     0     0     0     0     1
## 53          1     0     0     0     0     1
## 54          1     0     0     0     0     1
## 55          1     0     0     0     0     1
## 56          1     0     0     0     0     1
## 57          1     0     0     0     0     1
## 58          1     0     0     0     0     1
## 59          1     0     0     0     0     1
## 60          1     0     0     0     0     1
## 61          1     0     0     0     0     0
## 62          1       0       0       0       0       0
## 63          1       0       0       0       0       0
## 64          1       0       0       0       0       0
## 65          1       0       0       0       0       0
## 66          1       0       0       0       0       0
## 67          1       0       0       0       0       0
## 68          1       0       0       0       0       0
## 69          1       0       0       0       0       0
## 70          1       0       0       0       0       0
## 71          1       0       0       0       0       0
## 72          1       0       0       0       0       0
## attr(,"assign")
## [1] 0 1 1 1 1 1
## attr(,"contrasts")
## attr(,"contrasts")$spray2
## [1] "contr.treatment"
```

Exercise 0.23. Use `predict` to find predicted values for spray type C, using each of the two models. Again, make sure you understand the result. Then proceed with producing 95% prediction intervals. Reflect critically on the usefulness of these intervals for this application.

Click for solution

```
## SOLUTION
predict(fit1.insects, newdata= data.frame(spray="C"), interval="prediction")
```

```
##        fit       lwr     upr
## 1 2.083333 -6.066732 10.2334
```

```
#      fit       lwr     upr
#1 2.083333 -6.066732 10.2334
predict(fit2.insects, newdata= data.frame(spray2="C"), interval="prediction")
```

```
##        fit       lwr     upr
## 1 2.083333 -6.066732 10.2334
```

```
#     fit       lwr     upr
#1 2.083333 -6.066732 10.2334

# The predictions are the same, as it should be (the coding does not change
# the properties of the model).

# The prediction intervals are rather too wide to be actually useful,
# and extend into the negative range which is implausible since the data
# are counts.  (The larger variability especially for spray types A and F
# has led to a relatively large overall standard error of about 4, which has
# inflated the width of the prediction intervals.)
```

Exercise 0.24. This question is optional and not examinable, but it may help you to understand better the concepts of different factor codings in general. It is recommended to return to this question at the end of the practical, if there is still time.

The coding scheme used above is called *dummy coding*. You can check by typing

```
options("contrasts")
```

what the coding scheme is — the default is `contr.treatment` which means just dummy coding. Previously, we mentioned another coding scheme *(effect coding)* which is characterized by $\sum_{i=1}^{a} \beta_i = 0$. You can change the coding scheme to effect coding by typing

```
options(contrasts=c("contr.sum", contrasts=TRUE))
```

As before, fit the model, look at `summary` and `model.matrix`, and obtain the prediction for spray type C. Again, convince yourself of the plausibility of the results. When you are completed, you can return to the dummy coding scheme via

```
options(contrasts=c("contr.treatment", contrasts=TRUE))
```

Click for solution

```
## SOLUTION
### this piece of code is not exam relevant

options("contrasts") # checks the current coding scheme. Default  "contr.teactment" wh
```

```
## $contrasts
##         unordered           ordered
## "contr.treatment"      "contr.poly"
```

```
options(contrasts=c("contr.sum", contrasts=TRUE)) # changing coding scheme to effect c
options("contrasts") # check what has changed
```

```
## $contrasts
##                contrasts
## "contr.sum"      "TRUE"
```

```
fit3.insects<- lm(insects ~ spray, data= insects) # fit model
summary(fit3.insects)
```

```
##
```

```
## Call:
## lm(formula = insects ~ spray, data = insects)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -8.333 -1.958 -0.500  1.667  9.333
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   9.5000     0.4622  20.554  < 2e-16 ***
## spray1        5.0000     1.0335   4.838 8.22e-06 ***
## spray2        5.8333     1.0335   5.644 3.78e-07 ***
## spray3       -7.4167     1.0335  -7.176 7.87e-10 ***
## spray4       -4.5833     1.0335  -4.435 3.57e-05 ***
## spray5       -6.0000     1.0335  -5.805 2.00e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.922 on 66 degrees of freedom
## Multiple R-squared:  0.7244, Adjusted R-squared:  0.7036
## F-statistic:  34.7 on 5 and 66 DF,  p-value: < 2.2e-16
```

```r
model.matrix(fit3.insects)
```

```
##    (Intercept) spray1 spray2 spray3 spray4 spray5
## 1            1      1      0      0      0      0
## 2            1      1      0      0      0      0
## 3            1      1      0      0      0      0
## 4            1      1      0      0      0      0
## 5            1      1      0      0      0      0
## 6            1      1      0      0      0      0
## 7            1      1      0      0      0      0
## 8            1      1      0      0      0      0
## 9            1      1      0      0      0      0
## 10           1      1      0      0      0      0
## 11           1      1      0      0      0      0
## 12           1      1      0      0      0      0
## 13           1      0      1      0      0      0
## 14           1      0      1      0      0      0
## 15           1      0      1      0      0      0
## 16           1      0      1      0      0      0
## 17           1      0      1      0      0      0
## 18           1      0      1      0      0      0
## 19           1      0      1      0      0      0
## 20           1      0      1      0      0      0
## 21           1      0      1      0      0      0
## 22          1    0    1    0    0    0
## 23          1    0    1    0    0    0
## 24          1    0    1    0    0    0
## 25          1    0    0    1    0    0
## 26          1    0    0    1    0    0
## 27          1    0    0    1    0    0
## 28          1    0    0    1    0    0
## 29          1    0    0    1    0    0
## 30          1    0    0    1    0    0
## 31          1    0    0    1    0    0
## 32          1    0    0    1    0    0
## 33          1    0    0    1    0    0
## 34          1    0    0    1    0    0
## 35          1    0    0    1    0    0
## 36          1    0    0    1    0    0
## 37          1    0    0    0    1    0
## 38          1    0    0    0    1    0
## 39          1    0    0    0    1    0
## 40          1    0    0    0    1    0
## 41          1    0    0    0    1    0
## 42          1    0    0    0    1    0
## 43          1    0    0    0    1    0
## 44          1    0    0    0    1    0
## 45          1    0    0    0    1    0
## 46          1    0    0    0    1    0
## 47          1    0    0    0    1    0
## 48          1    0    0    0    1    0
## 49          1    0    0    0    0    1
## 50          1    0    0    0    0    1
## 51          1    0    0    0    0    1
## 52          1    0    0    0    0    1
## 53          1    0    0    0    0    1
## 54          1    0    0    0    0    1
## 55          1    0    0    0    0    1
## 56          1    0    0    0    0    1
## 57          1    0    0    0    0    1
## 58          1    0    0    0    0    1
## 59          1    0    0    0    0    1
## 60          1    0    0    0    0    1
## 61          1   -1   -1   -1   -1   -1
## 62          1   -1   -1   -1   -1   -1
## 63          1   -1   -1   -1   -1   -1
## 64          1   -1   -1   -1   -1   -1
## 65          1   -1   -1   -1   -1   -1
## 66          1   -1   -1   -1   -1   -1
## 67          1   -1   -1   -1   -1   -1
## 68           1      -1      -1      -1      -1      -1
## 69           1      -1      -1      -1      -1      -1
## 70           1      -1      -1      -1      -1      -1
## 71           1      -1      -1      -1      -1      -1
## 72           1      -1      -1      -1      -1      -1
## attr(,"assign")
## [1] 0 1 1 1 1 1
## attr(,"contrasts")
## attr(,"contrasts")$spray
## [1] "contr.sum"
```

```
predict(fit3.insects, newdata= data.frame(spray="C"), interval="prediction")
```

```
##        fit       lwr     upr
## 1 2.083333 -6.066732 10.2334
```

```
options(contrasts=c("contr.treatment", contrasts=TRUE))
  # brings us back to the old coding scheme......
```

Factorial experiments

Missile data set

Montgomery (*Design and Analysis of Experiments*, Wiley, 1991) gives the results of an experiment involving a storage battery used in the launching mechanism of a shoulder-fired ground-to-air missile. Three material types can be used to make the battery plates. The objective is to make a battery that is relatively unaffected by the ambient temperature. As well as varying the material for the plates, the ambient temperature was varied. The response measured was the effective life of the battery in hours. The data are as follows:

Material \ Temperature (^{o}F)	Low	Medium	High
1	130, 155, 74, 180	34, 40, 80, 75	20, 70, 82, 58
2	150, 188, 159, 126	136, 122, 106, 115	25, 70, 58, 45
3	138, 110, 168, 160	174, 120, 150, 139	96, 104, 82, 60

Load the data frame `missile` contained in the `sm2data` library.

Now execute the following command

```r
library(sm2data)
data(missile)
missile$Temperature<-factor(missile$Temperature,levels=c("Low","Medium","High"))
```

Note: The `Temperature` factor in the `missile` data is encoded with levels `High`, `Medium`, `Low` (by default `R` uses alphabetic ordering) instead of the more natural `Low`, `Medium`, `High`. The command you are asked to execute rearranges the order of the factor levels to `Low`, `Medium`, `High` so that `Low` is now the `reference category` for the `Temperature` factor.

Exercise 0.25. Which type of experiment are we considering? Name some characteristics in terms of *design* characterizing this experiment.

Click for solution

```r
## SOLUTION
# 3x3 factorial design, complete and balanced.
```

We consider in what follows a (constrained) model of type

$$E(\text{life}|\mathcal{A},\mathcal{B}) = \mu + \tau_i^{\mathcal{A}} + \tau_j^{\mathcal{B}} + \tau_{ij}^{\mathcal{AB}} \tag{2}$$

where $i, j = 1, 2, 3$, with $\mathcal{A}$ corresponding to factor `temperature` and $\mathcal{B}$ to factor 'material'.

Note: For consistency with solutions that will be provided, in this practical please let `temperature` be the first covariate entering the model, then `material` and finally the interaction term.

Exercise 0.26. Fit the constrained model to the data. By looking at the R output, which levels (and combination of levels) of the considered factors do not appear in the output? Referring to model (2), formulate the constraints explicitly in mathematical notation.

Click for solution

```r
## SOLUTION
missile.lm<- lm(Battery.Life~Temperature*Material, data=missile)
summary(missile.lm)
```

```
##
## Call:
## lm(formula = Battery.Life ~ Temperature * Material, data = missile)
```

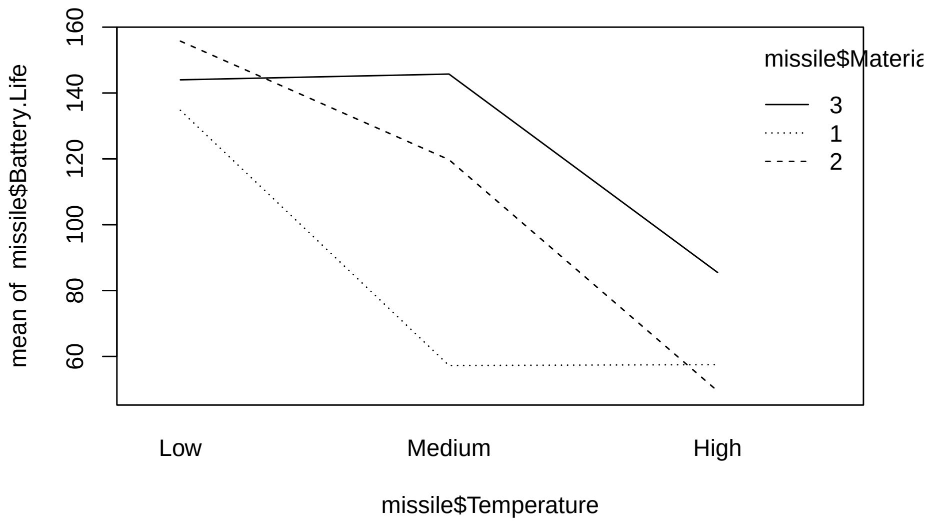
```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -60.750 -14.625   1.375  17.938  45.250
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   134.75      12.99  10.371 6.46e-11 ***
## TemperatureMedium             -77.50      18.37  -4.218 0.000248 ***
## TemperatureHigh               -77.25      18.37  -4.204 0.000257 ***
## Material2                      21.00      18.37   1.143 0.263107
## Material3                       9.25      18.37   0.503 0.618747
## TemperatureMedium:Material2    41.50      25.98   1.597 0.121886
## TemperatureHigh:Material2     -29.00      25.98  -1.116 0.274242
## TemperatureMedium:Material3    79.25      25.98   3.050 0.005083 **
## TemperatureHigh:Material3      18.75      25.98   0.722 0.476759
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 25.98 on 27 degrees of freedom
## Multiple R-squared:  0.7652, Adjusted R-squared:  0.6956
## F-statistic:    11 on 8 and 27 DF,  p-value: 9.426e-07
```

```
# t_1^A = t_1^B=0,  t_11^AB=0,  t_1j^AB= t_j1^AB=0, j=2,3.
```

Exercise 0.27. Produce a plot for a *two-way interaction*, using mean values for the response. Compare the plot with the parameter estimates for the model with interaction. Write some considerations.

Click for solution

```
## SOLUTION
interaction.plot(missile$Temperature, missile$Material, missile$Battery.Life)
```



```
# We see clearly in the plot the negative effect of medium and high temperatures,
# corresponding to a negative sign in the coefficients for TemperatureMedium  and
# TemperatureHigh.

# We further see a mostly postive effect of materials 2 and 3 as compared to the refer
# category (1), reflected in positive signs for the indicators Material2 and Material3

# Finally, the kinks for medium temperatures hint at some interaction, visbile in the
# significant Material3:TemperatureMedium interaction term.
#(That is, the strength of the effect that material3 has on the lifetime depends on th
# temperature, and is particularly strong for medium temperature).
```

Exercise 0.28. Using the model with interaction, calculate 99% confidence intervals for the *expected* life of batteries and corresponding 99% prediction intervals for the life of a *single* battery:

a. made with material 1 and used at low temperature,
b. made with material 2 and used at low temperature, and
c. made with material 1 and used at high temperature.
d. For at least one of (a), (b) or (c), do this *without* using function `predict()`.

Click for solution

```
## SOLUTION (a), (b) and (c)
new<-data.frame(Temperature=c("Low","Low","High"), Material=c("1", "2", "1"))

# Confidence interval
predict(missile.lm, newdata=new, interval="confidence", level=0.99)
```

```
##      fit      lwr      upr
## 1 134.75  98.7521 170.7479
## 2 155.75 119.7521 191.7479
## 3  57.50  21.5021  93.4979
```

```
# Prediction interval
predict(missile.lm, newdata=new, interval="prediction", level=0.99)
```

```
##      fit       lwr      upr
## 1 134.75  54.25624 215.2438
## 2 155.75  75.25624 236.2438
## 3  57.50 -22.99376 137.9938
```

```
## SOLUTION (d)
# For "Low", "2" by hand:
x0 <- c(1, 0, 0, 1, 0, 0, 0, 0, 0)

y0hat <- c(t(x0) %*% missile.lm$coef)
y0hat
```

```
## [1] 155.75
```

```
# 155.75

s <- summary(missile.lm)$sigma
s
```

```
## [1] 25.98486
```

```
# 25.98486
tcrit <- qt(0.995, 36 - 9)
tcrit
```

```
## [1] 2.770683
```

```
# 2.770683

# CI
y0hat + c(-1, 1) * c(s * tcrit * sqrt(t(x0) %*% summary(missile.lm)$cov.unscaled %*% x0))
```

```
## [1] 119.7521 191.7479
```

```
# [1]   119.7521 191.7479

# PI
y0hat + c(-1, 1) * c(s * tcrit * sqrt(1 + t(x0) %*% summary(missile.lm)$cov.unscaled %*% x0))
```

```
## [1]  75.25624 236.24376
```

```
# [1]   75.25624 236.24376
```

Practical 4

To start, create a new R script by clicking on the corresponding menu button (in the top left); and save it somewhere in your computer. You can now write all your code into this file, and then execute it in R using the menu button Run or by simple pressing **ctrl + enter**.

Analysis of variance

Missile data set

In this practical, we will continue with the data set **missile** from the previous practical.

Exercise 0.29. Display again the model `summary` of the fitted interaction model. Read and interpret from this the result of the overall F-test.

Click for solution

```
## SOLUTION
missile.lm<- lm(Battery.Life~Temperature*Material, data=missile)
summary(missile.lm)
```

```
##
## Call:
## lm(formula = Battery.Life ~ Temperature * Material, data = missile)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -60.750 -14.625   1.375  17.938  45.250
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    134.75      12.99  10.371 6.46e-11 ***
## TemperatureMedium              -77.50      18.37  -4.218 0.000248 ***
## TemperatureHigh                -77.25      18.37  -4.204 0.000257 ***
## Material2                          21.00      18.37   1.143 0.263107
## Material3                           9.25      18.37   0.503 0.618747
## TemperatureMedium:Material2        41.50      25.98   1.597 0.121886
## TemperatureHigh:Material2         -29.00      25.98  -1.116 0.274242
## TemperatureMedium:Material3        79.25      25.98   3.050 0.005083 **
## TemperatureHigh:Material3          18.75      25.98   0.722 0.476759
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 25.98 on 27 degrees of freedom
## Multiple R-squared:  0.7652, Adjusted R-squared:  0.6956
## F-statistic:    11 on 8 and 27 DF,  p-value: 9.426e-07
```

```
# F-statistic:    11 on 8 and 27 DF,  p-value: 9.426e-07
# clearly, the predictors explain jointly a significant proportion of the variation
```

Exercise 0.30. Use function `anova()` to perform (sequential) analysis of variance to investigate the significance of the effects included in the model. At $\alpha = 0.05$, what is the simplest appropriate model? Do your results change when changing the order of terms?

Click for solution

```
## SOLUTION
anova(missile.lm)
```

```
## Analysis of Variance Table
##
## Response: Battery.Life
##                      Df Sum Sq Mean Sq F value    Pr(>F)
## Temperature           2  39119 19559.4 28.9677 1.909e-07 ***
## Material              2  10684  5341.9  7.9114  0.001976 **
## Temperature:Material  4   9614  2403.4  3.5595  0.018611 *
## Residuals            27  18231   675.2
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# At the 5% level, we do need the interaction terms.
# Hence, Temperature*Material is the simplest appropriate model.
anova(lm(Battery.Life~Material*Temperature, data=missile))
```

```
## Analysis of Variance Table
##
## Response: Battery.Life
##                      Df Sum Sq Mean Sq F value    Pr(>F)
## Material              2  10684  5341.9  7.9114  0.001976 **
## Temperature           2  39119 19559.4 28.9677 1.909e-07 ***
## Material:Temperature  4   9614  2403.4  3.5595  0.018611 *
## Residuals            27  18231   675.2
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# does not depend on order of inclusion (balanced factorial design)
```

Exercise 0.31. Test (via a partial F-test) the hypothesis $H_0 = E(\text{life}|\text{material}) = \mu + \tau_i^{\mathcal{A}}$ vs $H_A = E(\text{life}|\text{material, temperature}) = \mu + \tau_i^{\mathcal{A}} + \tau_j^{\mathcal{B}}$ at a 5% significance level. What you conclude? *Hint*: Fit both models and then use `anova( , )`.

Click for solution

```
## SOLUTION
missile.material     <-lm(Battery.Life~Material,data=missile)
missile.materialtemp <-lm(Battery.Life~Material+Temperature,data=missile)
anova(missile.material,missile.materialtemp)
```

```
## Analysis of Variance Table
##
## Model 1: Battery.Life ~ Material
## Model 2: Battery.Life ~ Material + Temperature
##   Res.Df   RSS Df Sum of Sq      F    Pr(>F)
## 1     33 66963
## 2     31 27845  2     39119 21.776 1.239e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#1     33 66963
#2     31 27845  2     39119 21.776 1.239e-06 ***

# Clearly, the Temperature term needs to be included.
```

Exercise 0.32. Let's go back to the model with interaction. Using the values in the corresponding ANOVA table calculate R^2. Interpret the result and verify it by comparison with the R^2 given in the output from the `summary()` function.

Click for solution

```
## SOLUTION
anova(missile.lm)
```

```
## Analysis of Variance Table
```

```
##
## Response: Battery.Life
##                      Df Sum Sq Mean Sq F value    Pr(>F)
## Temperature           2  39119 19559.4 28.9677 1.909e-07 ***
## Material              2  10684  5341.9  7.9114  0.001976 **
## Temperature:Material  4   9614  2403.4  3.5595  0.018611 *
## Residuals            27  18231   675.2
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
sse <- 18231
ssr <- 39119+10684+9614
sst <- sse+ ssr
rsquared <- ssr/sst
rsquared
```

```
## [1] 0.7652097
```

```
#   0.7652097

# or we can use
av<-anova(missile.lm)
av
```

```
## Analysis of Variance Table
##
## Response: Battery.Life
##                      Df Sum Sq Mean Sq F value    Pr(>F)
## Temperature           2  39119 19559.4 28.9677 1.909e-07 ***
## Material              2  10684  5341.9  7.9114  0.001976 **
## Temperature:Material  4   9614  2403.4  3.5595  0.018611 *
## Residuals            27  18231   675.2
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
names(av)
```

```
## [1] "Df"      "Sum Sq"  "Mean Sq" "F value" "Pr(>F)"
```

```
av$"Sum Sq"
```

```
## [1] 39118.722 10683.722  9613.778 18230.750
```

```
sst<-sum(av$"Sum Sq")
ssr<- sum(av$"Sum Sq"[1:3])
sse<-av$"Sum Sq"[4]
rsquared <- ssr/sst
rsquared
```

```
## [1] 0.7652098
```

```
# [1] 0.7652098


# now from the ouput of the summary function
summary(missile.lm)
```

```
##
## Call:
## lm(formula = Battery.Life ~ Temperature * Material, data = missile)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -60.750 -14.625   1.375  17.938  45.250
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   134.75      12.99  10.371 6.46e-11 ***
## TemperatureMedium             -77.50      18.37  -4.218 0.000248 ***
## TemperatureHigh               -77.25      18.37  -4.204 0.000257 ***
## Material2                      21.00      18.37   1.143 0.263107
## Material3                       9.25      18.37   0.503 0.618747
## TemperatureMedium:Material2    41.50      25.98   1.597 0.121886
## TemperatureHigh:Material2     -29.00      25.98  -1.116 0.274242
## TemperatureMedium:Material3    79.25      25.98   3.050 0.005083 **
## TemperatureHigh:Material3      18.75      25.98   0.722 0.476759
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 25.98 on 27 degrees of freedom
## Multiple R-squared:  0.7652, Adjusted R-squared:  0.6956
## F-statistic:    11 on 8 and 27 DF,  p-value: 9.426e-07
```

```
# Multiple R-squared: 0.7652,     Adjusted R-squared: 0.6956
```

Mallows' C_I

Cement data set

Read in the Portland **cement** data discussed in the lecture notes. Please use the code as follows to ensure that the ordering and names of variables correspond to those used in the lecture.

```
#install.packages("MASS")
library(MASS)
```

```
data(cement, package="MASS")
cement=cement[,c(5,1,2,3,4)]
names(cement)= c("heat", "aluminate", "tri.silicate", "ferite", "di.silicate")
```

Exercise 0.33. Fit a linear model using `heat` as response, and the four chemical compounds as predictors. Save the fitted model, which will serve as our `full` model in what follows, into an object `cement.fit.full`. Extract, for later use, the residual standard error from this model, and save it into an object `s`.

Click for solution

```
## SOLUTION
cement.fit.full = lm(heat~ aluminate + tri.silicate + ferite + di.silicate, data = ceme
s<- summary(cement.fit.full)$sigma
```

Exercise 0.34. Produce a few sequential ANOVA tables for these data (using several orders of inclusion) to get impression on the relative importance of the variables. Summarize your conclusions.

Click for solution

```
## SOLUTION
anova(cement.fit.full)
```

```
## Analysis of Variance Table
##
## Response: heat
##              Df  Sum Sq Mean Sq  F value    Pr(>F)
## aluminate     1 1450.08 1450.08 242.3679 2.888e-07 ***
## tri.silicate  1 1207.78 1207.78 201.8705 5.863e-07 ***
## ferite        1    9.79    9.79   1.6370    0.2366
## di.silicate   1    0.25    0.25   0.0413    0.8441
## Residuals     8   47.86    5.98
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova( lm(heat~ ferite + di.silicate+aluminate + tri.silicate, data=cement))
```

```
## Analysis of Variance Table
##
## Response: heat
##              Df  Sum Sq Mean Sq  F value    Pr(>F)
## ferite        1  776.36  776.36 129.7624 3.184e-06 ***
## di.silicate   1 1763.66 1763.66 294.7812 1.347e-07 ***
## aluminate     1  124.90  124.90  20.8763  0.001828 **
## tri.silicate  1    2.97    2.97   0.4968  0.500901
## Residuals     8   47.86    5.98
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova( lm(heat~ di.silicate+aluminate + tri.silicate+ ferite, data=cement))
```

```
## Analysis of Variance Table
##
## Response: heat
##              Df  Sum Sq Mean Sq  F value    Pr(>F)
## di.silicate   1 1831.90 1831.90 306.1859 1.161e-07 ***
## aluminate     1  809.10  809.10 135.2350 2.722e-06 ***
## tri.silicate  1   26.79   26.79   4.4776   0.06724 .
## ferite        1    0.11    0.11   0.0182   0.89592
## Residuals     8   47.86    5.98
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# We see that (of course) the SS and significances do change when changing the order of
# inclusion; however for all orders of inclusion it is the case that 2 or 3  predictors are
# sufficient to explain the variation in the response. It also appears that aluminate +
# tri.silicate appear to explain generally more variation than the other two compounds
```

Exercise 0.35. Produce a function `CI` which implements the quantity C_I. Your function should have the shape

```
CI<-function(model, var){
  pI <- length(model$coef)
  n  <-...
  RSSI <- ...
  ci <- ...
  return(ci)
}
```

where `model` is a fitted model object, and `var` the error variance obtained from the *full* model.

Click for solution

```
## SOLUTION
CI<-function(model, var){
  pI <- length(model$coef)
  n  <- length(model$residuals)
  RSSI <- sum((model$residuals)^2)
  ci <- RSSI/var+2*pI-n
  return(ci)
}
```

Exercise 0.36. Apply your function `CI` to as many of the 16 submodels of the full model as you are able to do, in order to reproduce the results in the lecture notes. There are multiple ways of doing this, for instance the model $1 + x_1 + x_3$ could be fitted via any of

```
cement.fit.1.3 = lm(heat~  aluminate+ferite,   data = cement)
cement.fit.1.3 = lm(heat~.,   data = cement[,c(1,2,4)])
```

Click for solution

```
## SOLUTION
# for instance,
cement.fit.1.3 = lm(heat~  aluminate+ferite,   data = cement)
cement.fit.1.3 = lm(heat~.,   data = cement[,c(1,2,4)])
CI(cement.fit.1.3,s^2)
```

```
## [1] 198.0947
```

```
# [1] 198.0947 (this agrees with the result in the lecture notes)

cement.fit.1 = lm(heat~  aluminate,   data = cement)
cement.fit.1 = lm(heat~.,   data = cement[,c(1,2)])
CI(cement.fit.1,s^2)
```

```
## [1] 202.5488
```

```
# [1] 202.5488     (this agrees with the result in the lecture notes)
```

Exercise 0.37. Contemplate more efficient ways of carrying out this analysis (you will probably not have time to put your thoughts into practice).

Click for solution

```
## SOLUTION
# As a first simplification, it would be useful to define the set of all possible subs
# (that is, the `power set' of the predictors) and then to write a loop which computes
# each element of this power set (This is in in practice a bit more tricky then it loo

# An even better strategy would be to discard models automatically which appear implau
# or unlikely to be optimal, given the information collected so far.
# This will lead into stepwise methods (see the relevant section in the lecture notes)
```