UNIVERSITY OF EDINBURGH

SCHOOL OF MATHEMATICS

Generalised Regression Models

GRM: The Basics – Linear Statistical Modelling in R **Semester 1, 2022–2023**

In this document we study how R may be used to fit and analyse linear statistical models of the form:

$$\mathrm{E}(\mathbf{Y}) = X\beta \quad \text{and} \quad \mathrm{var}(\mathbf{Y}) = \sigma^2\mathbf{I},$$

where $\mathbf{Y}$ is a vector of responses, X is a design matrix, β is a vector of regression coefficients, and σ^2 is an unknown constant.

This modelling framework includes the types of models required for

- *Regression*; and
- *Analysis of Variance*.

The `lm` function is used to fit linear regression models in R, and thus to produce fitted model objects. These objects are then analysed by generic functions, such as `summary`, to extract relevant information.

Although analysis of variance models are special cases of the linear model used for regression, R provides the `aov` function for fitting such models. The advantage of using the `aov` function is that the generic functions for extracting information from model objects produce appropriate analyses for *analysis of variance*.

The help pages for `lm` and `aov` give information on these functions: examples of their use are given at the end of the help pages.

```
help(lm)
```

1 Modelling Framework

Linear statistical modelling in R is based on object-oriented features of the R language. R functions can recognize the type of an object and carry out the appropriate action for that type of object. Conveniently, the same functions can be used to extract information from different types of model object, of which `lm` and `aov` are examples. This provides a consistent interface to conduct a wide range of statistical modelling, with only very minor changes in the way we specify models and study fitted models, e.g., writing `aov` in place of `lm`.

Two of the most important features of R modelling are:

- *data frame* objects which contain the data for the model; and
- *model formula* objects which specify the *form* of the relationship between the response variable and the explanatory variables.

1.1 Data Frames

A data frame is a two-dimensional data object similar to a `matrix`. However, unlike a matrix, a data frame can contain columns of different data types, e.g., `numeric` and `character`. One way to think of a data frame is as a 'directory' of closely related columns.

```
n <- 25
x <- seq(0,10,length=n)                        # generate data
y <-  15 + 0.5*x + rnorm(n,mean=0,sd=.1)
cbind(x, y)                                    # print artificial data
```

Plot these data in a graphics window. What model are we generating the data from?

To create a data frame from vectors `x` and `y` type:

```
xy.df <- data.frame(xvar=x, yvar=y) # construct data frame
xy.df                               # print data frame
xy.df$xvar                          # extract and print col1
xy.df$yvar                          # extract and print col2
```

1.1.1 Factors

A `factor` variable is appropriate if the explanatory variable is categorical, e.g., a treatment code in a designed experiment.

```
treat <- factor(c('T1','T2','T1'))
treat
summary(treat)                  # tally categories
```

Sometimes we might want to ignore the continuous nature of a variable, and redefine it as a factor.

```
xfac <- factor(cut(x,5))        # 5 categories
xfac
summary(xfac)                   # tally categories
xy.df <- data.frame(xy.df,xfac=xfac) # add a col to xy.df
```

An `ordered` factor is a factor that contains information on the relative ordering of the categories.

```
xord <- ordered(xfac)           # order factor
xord
summary(xord)                   # tally categories
```

1.2 Model Formula: Notation for Models

A model formula is used to specify the type of model to be fitted: this approach is based on the notation suggested by Wilkinson and Rogers (1973, *Applied Statistics*).

1.2.1 Simple Formulae

A simple formula is given by

```
y ~ x                    # a formula
```

This states that `y` is modelled by `x` (and a constant `1`), e.g. $\mathrm{E}(Y) = \alpha + \beta x$.

A constant is always included in a model unless explicitly removed by placing `-1` in the formula, e.g.,

```
y ~ x - 1                # no constant term in model
```

A transformation of the response variable can be specified on the left hand side of the formula, e.g., the formula

```
log(y) ~ x
```

states that the logarithm of `y` is modelled by `x`, e.g., $\mathrm{E}(Z) = \alpha + \beta x$, where $Z = \log Y$.

Examples of model notation

Model formula	Meaning
`y ~ x`	`y` is modelled as `x`
`y ~ x1+x2+x3`	`y` is modelled as `x1` plus `x2` plus `x3`
`y ~ poly(x,2)`	`y` is modelled as a quadratic in `x`
`y ~ poly(x,m)`	`y` is modelled as `m` degree polynomial in `x`
`log(y) ~ x`	`log(y)` is modelled as `x`

1.2.2 Continuous Explanatory Variables

The formula

```
y ~ x1 + x2 + x3                    # several x variables
```

represents a linear model

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \varepsilon.$$

1.2.3 Categorical Explanatory Variables

Using a `factor` variable in R is equivalent to using 'dummy variables' (e.g. variables that can only take values of either 0 or 1). It is unnecessary to explicitly create dummy variables, but be careful that you know which parameterization is being used.

Two widely used parameterizations for factor effects (in models with an intercept term) are obtained by using

(i) a **corner-point** constraint (set the parameter associated with level 1 of a factor equal to zero), e.g. if the main effects are $\alpha_1, \ldots, \alpha_t$, then assume $\alpha_1 = 0$

(ii) a **sum-to-zero** constraint (set the sum of the main effects equal to zero), e.g. $\sum_{i=1}^{t} \alpha_i = 0$

The **corner-point** constraint is the default in R.

1.2.4 Interactions

Interactions may be included in models with the colon symbol (`:`).

Continuous variables: An 'interaction' term `x1:x2` can be included with

```
y ~ x1 + x2 + x1:x2
```

which represents a model of the form

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2 + \varepsilon$$

Factor variables: If `a` and `b` are factors, the formula

```
y ~ a + b + a:b
```

represents a two-way model of the form

$$Y = \mu + \alpha_i + \beta_j + (\alpha\beta)_{ij} + \varepsilon$$

Mixed continuous and factor variables: If `x` is continuous and `a` is a factor, the formula

```
y ~ a + x + a:x
```

represents a linear model of the form

$$Y = \alpha + \alpha_i + \beta x + \beta_i x + \varepsilon$$

An example is given by the Insulation data analysis in Section 4.3 of the course notes, in which the model corresponds to fitting two separate regression lines. With this parameterization we can test for parallel regression lines by testing the interaction term `a:x`, and then, if there is no interaction, to test for equality of regression lines we can test `a`.

Summary of model notation

Model formula	Meaning
$T \sim F$	T is modelled as F
$F_a + F_b$	Include both F_a and F_b in the model
$F_a - F_b$	Include all of F_a except what is in F_b in the model
$F_a : F_b$	Interaction between F_a and F_b
$F_a * F_b$	$F_a + F_b + F_a : F_b$
$F \wedge m$	All terms in F crossed to order m

Note that some of the notation we might want to use for calculations involving explanatory variables has a different meaning when used in model formulae. Therefore care is needed! To overcome this problem there is a function `I()` which can be used to over-ride the usual meaning of the following symbols in model formulae:

$$+ \qquad * \qquad \wedge \qquad -$$

For example,

```
y ~  x1 + x2 + I(x1*x2)
```

would include $(x_1 x_2)$ as an explanatory variable in a model.

Similarly,

```
y ~  x1 + I(x1^2)     or                y ~  I(x1+x2)
```

would include (x_1^2) or $(x_1 + x_2)$ respectively, in a model.

1.3 Model Fitting

To fit a regression model or an analysis of variance model use:

```
lm(formula,dataframe)                    # for regression
aov(formula,dataframe)                   # for ANOVA
```

For example, to fit a simple linear regression with an explanatory variable `x` and a response `y` with observations contained in a data frame `xy.df` use:

```
lm(y ~ x, data=xy.df)              # if xy.df is not attached
                                    # or
attach(xy.df)
lm(y ~ x)                          # if xy.df is attached
```

Using `lm(y ~ x, data=xy.df)` is better practice though.[1]

Similarly, to fit a one-way analysis of variance model with `y` as the response and `xfac` as a factor with observations contained in a data frame `xy.df` use:

```
aov(y ~ xfac)              # if xy.df is attached
```

To fit a model with two factors `a` and `b` and an interaction `a:b`, we would use:

```
aov(y ~ a + b + a:b)
```

which represents the two-way (factorial) analysis of variance model

$$\mathrm{E}(Y_{ij}) = \mu + \alpha_i + \beta_j + (\alpha\beta)_{ij}$$

where μ is the overall mean, α_i is the main effect of factor `a`, β_j is the main effect of factor `b`, and $(\alpha\beta)_{ij}$ is the interaction between the two factors.

1.4 Extracting Information from Model Objects

The following functions are used to extract relevant information from `model` objects.

Functions for extracting information from model objects

Function	Action
`summary`	Give details of model object
`plot`	Graphical displays of model object
`coefficients`	Extract coefficients
`formula`	Extract formula on which model is based
`fitted`	Extract fitted values
`predict`	Predicts from model
`residuals`	Extract residuals
`update`	Refit modified model
`anova`	Anova table
`AIC`	AIC, $2k - 2\log L(\widehat{\beta};\widehat{\sigma}^2)$, where k is the number of parameters
`deviance`	Extract model deviance (i.e., resid sum sqs for `lm` and `aov`)

[1] WARNING: In general, it is safer to avoid using `attach()` and explicitly use `data=xy.df` in the call to `lm`.

2 Examples: Linear Models

2.1 Simple Linear Regression

We have seen the following case studies from Chapter 1 of the notes:

- **Section 1.1** January flows on the Kootenai river
- **Section 1.2** Calibration of a flame photometer
- **Section 1.3** Comparison of two methods for measuring the bitterness of beer

from Workshop 1. The anlyses are given on Learn under:

Course Materials > **Feedback on Workshops and Problem Sheets**

2.2 Transformation to Linearity: Residual Analysis

The document

- Examples — Residual Analysis

has the following case studies from Chapter 1 of the notes:

- **Section 1.4** Forbes' data: Atmospheric pressure and the boiling point of water
- **Section 1.5** Estimating the weights of birds from the lengths of wing bones

In these case studies a transformation of the response is used to improve linearity. The examples with transformtions are

- **Section 1.6** Numbers of beetles killed by different doses of insecticide
- **Problem Sheet 1, Question 6** Testing plastic conduit pipe: numbers tested and the numbers which broke.

Transforming the data values, using logs helped with the outlier in the data set in Section 1.1, modelling riverflow:

- **Section 1.10** January flows on the Kootenai river (log transformation)

2.3 Linear model with many factors

See the Real Estate case study from Workshop 2 for an analysis of a linear model with many factors.

2.4 Polynomial Regression

See Adhesive data analysis from Section 1.7 of the course notes for an analysis of a polynomial regression model:

```
y <- c(35.1,39.7,40.3,42.0,44.1,47.0,46.0,48.1,49.9,44.3,45.1,46.2)
x <- c(12,12,12,24,24,24,36,36,36,48,48,48)
summary(lm(y~x+I(x^2)))
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 26.950000   3.418541   7.883 2.49e-05 ***
x            1.117778   0.259889   4.301  0.00199 **
I(x^2)      -0.015278   0.004264  -3.583  0.00590 **
```

2.5 Sinusoidal Regression

See Births data analysis from Section 1.8 of the course notes (see Problem Sheet 3) for an analysis of a sinusoidal regression model.

2.6 Testing Equality of Several Regression Lines

See Insulation data analysis from Section 1.9, and analysis from Section 4.3, of the course notes for an analysis of a several regression lines model. Are the lines parallel?

2.7 One-way Analysis of Variance

In an investigation of the pollution of a river, 1 litre specimens of water were collected from points along a five mile stretch of the river. Each specimen was taken at 6pm at a site chosen at random at the same depth of water. Five specimens were collected on each day of a single week and a pollution index, measured in suitable units, was obtained for each specimen with the following results:

M	**Tu**	**W**	**Th**	**F**	**Sa**	**Su**
145	210	195	140	195	45	120
40	185	150	155	230	40	55
40	30	205	90	115	195	50
120	60	110	160	235	65	80
80	55	160	95	225	145	45

You may assume that the values of the index are statistically independent and Normally distributed with common variance and expectations only depending on the day of the week. Investigate whether the distribution of these values is affected in any way by the day of the week.

The `factor` in this data set is `day` (which has 7 categories).

Model:

$$Y_{ij} = \mu + \alpha_i + \varepsilon_{ij}.$$

The α_i term is the day effect, and to test that there is no difference between days we use the null hypothesis $H_0 : \alpha_i = 0$ for all i. As usual we assume $\varepsilon_{ij} \sim N(0, \sigma^2)$; we can assess the validity of this assumption by investigation of residual plots.

```
classes <- c('M','Tu','W','Th','F','Sa','Su')
day <- factor(rep(classes,c(5,5,5,5,5,5,5)))       # factor day
day

y <- c(145,40,40,120,80,210,185,30,60,55,
       195,150,205,110,160,140,155,90,160,95,
       195,230,115,235,225,45,40,195,65,145,
       120,55,50,80,45)

river.df <- data.frame(day,y)
river.df

summary(river.df)

plot.design(river.df)                          # is M a weekend day?

river.aov <- aov(y ~ day,river.df)                      # fit model

summary(river.aov)                             # look at ANOVA table
plot(river.aov)                                # check fit of model

river.aov1 <- update(river.aov, . ~. -day)     # remove the factor day

anova(river.aov,river.aov1)                    # test nested models
coef(river.aov)                                # parameter estimates
```