Introduction to R and Model Specifications

Henrik Madsen

DTU Informatics

January 2012

This lecture

- Introduction to R - mostly by running R-scripts
- Libraries and information
- Reading Data and Data frames
- Getting subsets of data
- Model specifications in R
- How to get help

Libraries and information

- Homepage: http://www.r-project.org
- Important entry: http://cran.at.r-project.org/
- CRAN family of internet sites: http://CRAN.R-project.org
- R on ETH: stat.ethz.ch/R-manual/
- University of Oxford: http://www.stats.ox.ac.uk/pub/
- Manuals and help are installed with R

Packages and installation

- Important entry:
  http://stat.ethz.ch/R-manual/R-patched/doc/html/index.html
- Try eg. 'Packages' – 'stats' – 'StructTS'
- For packages in standard library:
  ```
   library('splines')
  ```
- For other packages you must first install:
  ```
  install.packages('tree',dependencies=TRUE)
  ```

Reading data from file

On Windows:

```
worms <-read.table("c:\\data\\worms.txt",header=T,row.names=1)
```

On Linux/Unix:

```
worms<-read.table("./data/worms.txt",header=T,row.names=1)
```

Typically once the file has been imported to R we want to do two things:

- Use `attach` to make the variables accessible by name within the R session, and
- Use `names` to see a list of the variable names

Also to see some information you might want to

- See the contents of the dataframe - just type its name
- Use `summary{worms}` to see a summary of the dataframe

Selecting Parts of a Dataframe

- To select all the rows of the first three columns:

```
worms[,1:3]
```

- To select the middle 11 rows for all columns:

```
worms[5:15,]
```

- To select only those rows which have `Area>3` and `Slope<4`:

```
worms[Area>3 & Slope<4,]
```

- Suppose we want the rows of the whole dataframe sorted by Area (the variable in column number one)

```
worms[order(worms[,1]),1:6]
```

- Alternatively, the dataframe can be sorted in descending order by Soil pH, with only Soil pH and Worm density as output:

```
worms[rev(order(worms[,4])),c(4,6)]
```

Specification of models

y: Dependent variable
x: Explanatory variable (continuous)
a: Explanatory variable (factor)

```
y ~ x    or    y ~ 1 + x
```

specifies the model

$$y_i = \mu + \beta x_i + e_i$$

and

```
y ~ -1 + x
```

implies no intercept.

```
y ~ a
```

specifies the model

$$y_{ij} = \alpha_j + e_{ij}; \quad i = 1, \ldots, n_j; \; j = 1, \ldots, k$$

the parameterization is however depend on the applied contrast.

Specification of models

- Additive 2-sided model:

  ```
  y ~ a1 + a2
  ```

  2-sided model with interaction

  ```
  y ~ a1 + a2 + a1:a2  or
  y ~ a1*a2
  ```

Specification of models

- Additive 2-sided model:

  ```
  y ~ a1 + a2
  ```

  2-sided model with interaction

  ```
  y ~ a1 + a2 + a1:a2  or
        y ~ a1*a2
  ```

- Hierarchical effects

  ```
  y ~ a1 + a2 %in% a1  or
        y ~ a1/a2
  ```

  a2 under a1 (alternatively: `y ~ a1 + a1:a2`).

Model specification (cont.)

The construction

```
a1*a2*a3
```

is understood by expanding

```
(1+a1):(1+a2):(1:a3)
```

as ordinary multiplication, ie.

```
(1 + a1 + a2 + a1:a2):(1 + a3)
```

and then

```
1 + a1 + a2 + a3 + a1:a2 + a1:a3 + a2:a3 + a1:a2:a3
```

Model specification (cont.)

Further the construction

```
(a1 + a2 + a3)^3
```

is the same as

```
a1*a2*a3
```

whereas

```
(a1 + a2 + a3)^2
```

corresponds to

```
a1*a2*a3 - a1:a2:a3
```

or

```
1 + a1 + a2 + a3 + a1:a2 + a1:a3 + a2:a3
```

Transformation of variables

In general we may write things like

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

However – Be careful using ^, /, and $*$ on continuous variables!!
Use the function `I()` instead, like in

```
log(y) ~ x1 + x2 + I(x1*x2) + I(x4/x5) + I((x6+x7)^2)
```

Analysis of Variance

- `summary(lm(...))` : Partial test
- `anova(lm(...))` : Sekvential test (alternatively `summary(aov())`).
- `anova(lm(...), ssType=3)` : SAS Type III test (partial), I, II and IV are also possible. Alternatively consider `drop1(aov())`
- `anova(fit.H0, fit.HA)` : Specific hypotheses.

Examples of more adv. R Model Formulae

- Tree-way ANOVA (not with three-way interaction):

  `y ∼ N*P*K-N:P:K`

- Analysis of Covariance

  `y ∼ x + gender`

  A common slope for y against x but with two intercepts, one for each gender.

- Split-plot ANOVA:

  `y ∼ a*b*c+Error(a/b/c)`

  A 3-way factorial setup, but three different error variances.

- Including multiple (polynomial) regression:

  `y ∼ poly(x,2)+z`

- Multiple regression

  `y ∼ (x+z+w)2̂`

  Fit three variables plus all their two-way interactions

- Non-parametric model

  `y ∼ s(x) + lo(z)`

  y is a function of smoothed x and loess z.

Tips for buiding multivariate models

- Consider multivariate relations using eg.

```
pairs(..)
```

- Then a good way to start is estimating non-parametric models:

```
model = gam(ozone ∼ s(rad) + s(temp) + s(wind)); plot(model)
```

- Use tree based methods to identify complex interactions, like:

```
model = tree(ozone ∼ .,data=ozone.pollution); plot(model)
```

Now a parametric model can be formulated.

Use R's possibilities for changing the model

- Use R's possibilities for updating or reducing the model:

```
model4 = update(model3, ~ .  - temp:wind);
summary(model4)
```

- When all terms are significant the model assumptions should be checked using eg.

```
plot(model6)
```

- Control of heteroscedasity etc. Transformation is a possible solution.

Error structure

Up to this point we have dealt with statistical analysis of data with gaussian errors. In practice, however, non-Gaussian erros are often seen:

- Poission errors, useful with count data.
- Binomial errors, useful with data on proportions.
- Gamma errors, useful with data showing constant coefficient of variation.
- Exponential errors, useful with data on time-to-death (survival analysis).

The error structure is defined by the **family** directive, and specified as a part of the model formula like:

```
glm( y ~ x + z, family = binomial)
```

Residuals

Standardized residuals (`stdres` in MASS):

$$e_i' = \frac{e_i}{s\sqrt{1-h_{ii}}}$$

Studentized residuals (`studres` in MASS):

$$e_i^* = \frac{y_i - \hat{y}_{(i)}}{\sqrt{Var[y_i - \hat{y}_{(i)}]}}$$

also called jack-knifed residuals. Found alternatively for linear models as

$$e_i^* = \frac{e_i'}{\sqrt{\frac{N-p-(e_i')^2}{N-p-1}}}$$

h_{ii} can be obtain using `lm.influence(...)$hat`.