TECHNISCHE
UNIVERSITÄT
WIEN

VIENNA
UNIVERSITY OF
TECHNOLOGY

R: A Free Software Project in Statistical Computing

Achim Zeileis
Institut für Statistik & Wahrscheinlichkeitstheorie

http://www.ci.tuwien.ac.at/~zeileis/

Acknowledgments

Thanks:

- Alex Smola & Machine Learning Group
- John Maindonald

Acknowledgements:

- Kurt Hornik & Friedrich Leisch
- Alexandros Karatzoglou

Overview

- What is R?
- Statistical computing with R
  - S language: programmable and object oriented
  - interfaces to other languages and environments
  - production-quality graphics
  - packaging mechanism for software delivery
- Reproducible research

What is R?

- R is an interactive computational environment for data analysis, inference and visualization.
- Developed for the Unix, Windows and Macintosh families of operating systems by an international team.
- Highly extensible through user-defined functions and a fast-growing list of add-on packages.
- Based on the award-winning S language which "has forever altered the way how people analyze, visualize and manipulate data ..." (ACM Software System Award 1998 to John Chambers).
- S-Plus is a commercial implementation of S.

What is R?

The "R Development Core Team" with members from New Zealand, Europe and North America:

Douglas Bates, John Chambers, Peter Dalgaard, Robert Gentleman, Kurt Hornik, Stefano Iacus, Ross Ihaka, Friedrich Leisch, Thomas Lumley, Martin Maechler, Guido Masarotto, Paul Murrell, Brian Ripley, Duncan Temple Lang, and Luke Tierney

But R would not be what it is today without the support of all those, who contributed by donating code, bug fixes and documentation.

What is R?

History of R

1991: Ross Ihaka and Robert Gentleman begin work on a project that will ultimately become R.
1993: First binary copies of R on Statlib.
1995: R release of sources under the GPL.
1997: R core group is formed.
1998: Comprehensive R Archive Network (CRAN).
1999: DSC meeting in Vienna, first R core meeting.
2000: R 1.0.0 is released.
2001: R Newsletter launched.
2002: R Foundation established.

Current release is 1.6.1

What is R?

Free software: Open source software like R does not cost money, but time to learn, and as time is money, this talk is mostly about free software in the sense of

R is open source, everybody can read the source code, hence you need not to rely on documentation to infer what the software really does. More importantly, everybody can use it, making research reproducible.

No owner? R is *not* in the public domain, you are given a license (GPL) to run the software.

S language

Fundamental design principle:

Everything in S is an object.

Or better, make that "should be" . . .

S language

Even the parse tree is an S object:

```
R> x <- parse(text = "sin(x + 2)")
R> x

expression(sin(x + 2))


R> x[[1]]

sin(x + 2)


R> x[[1]][[1]]

sin
```

Allows for computations on the language . . .

S language

Formulas:

- Models and plots are specified using a formula language.
- Model fitting functions return objects, these can be inspected, analyzed or visualized using methods for the corresponding class:
  - summary of model
  - extract residuals or fitted values
  - predict response for new inputs
  - get diagnostic plots
  - use in own functions

S language

`x, y`	. . . numeric variables
`a, b`	. . . categorical variables
`y ~ x`	. . . standard regression
`y ~ a`	. . . 1-way ANOVA
`y ~ a + x`	. . . main effects model
`y ~ a*x`	. . . main and interaction effects
`y ~ a*x + b*x`	. . . main and interaction effects

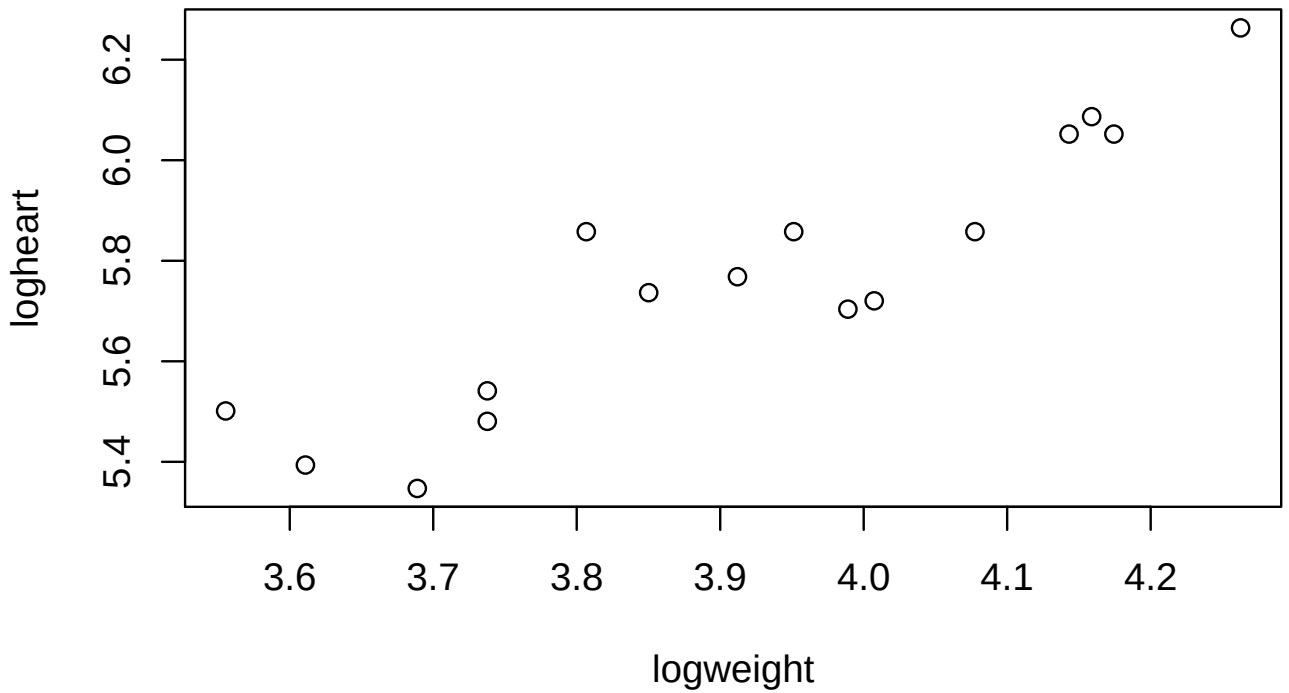
- Unified interface for model fitting functions.
- Contrasts for categorical variables can be customized.
- Functions for parsing the formula, extracting the response and design matrix.

Example: dolphins

```
R> load("dolphins.rda")
R> summary(dolphins)

   logweight        logheart      species
 Min.   :3.555   Min.   :5.347   delph:9
 1st Qu.:3.738   1st Qu.:5.531   styx :7
 Median :3.932   Median :5.752
 Mean   :3.916   Mean   :5.764
 3rd Qu.:4.094   3rd Qu.:5.906
 Max.   :4.263   Max.   :6.263

R> attach(dolphins)
R> class(species)

[1] "factor"
```

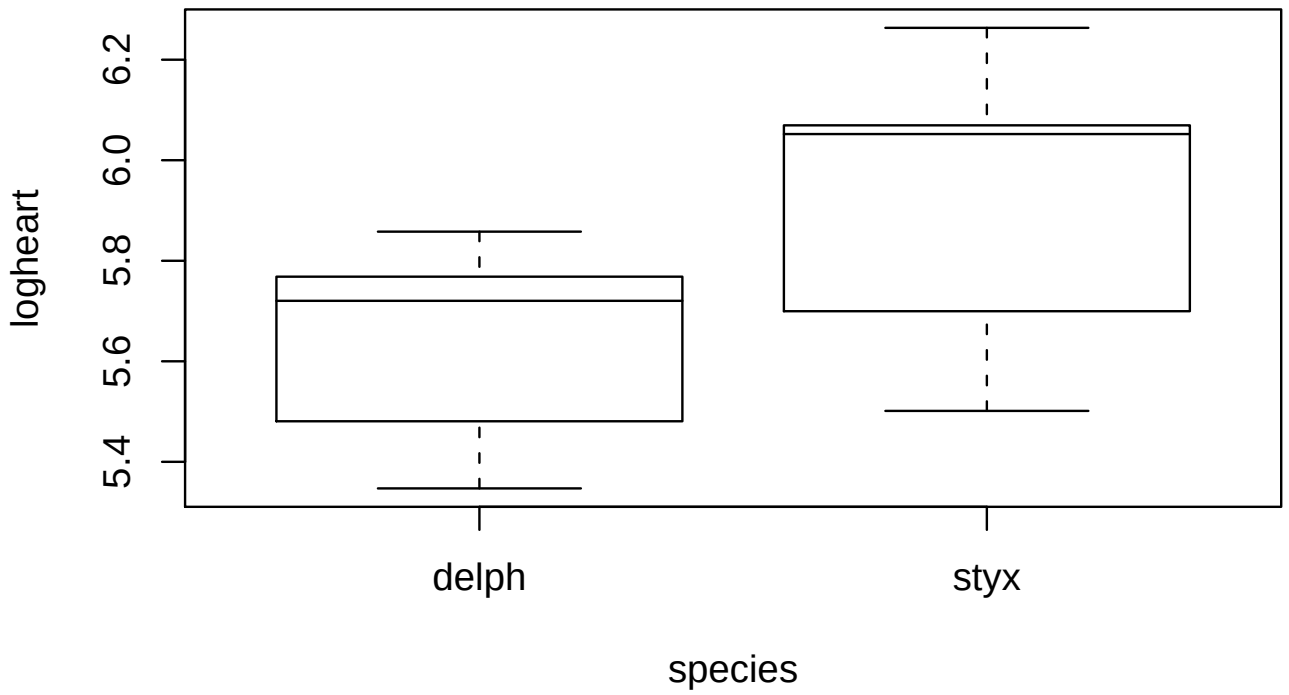
Example: dolphins

```
R> plot(logheart ~ logweight)
```



Example: dolphins

```
R> plot(logheart ~ species)
```



Example: dolphins

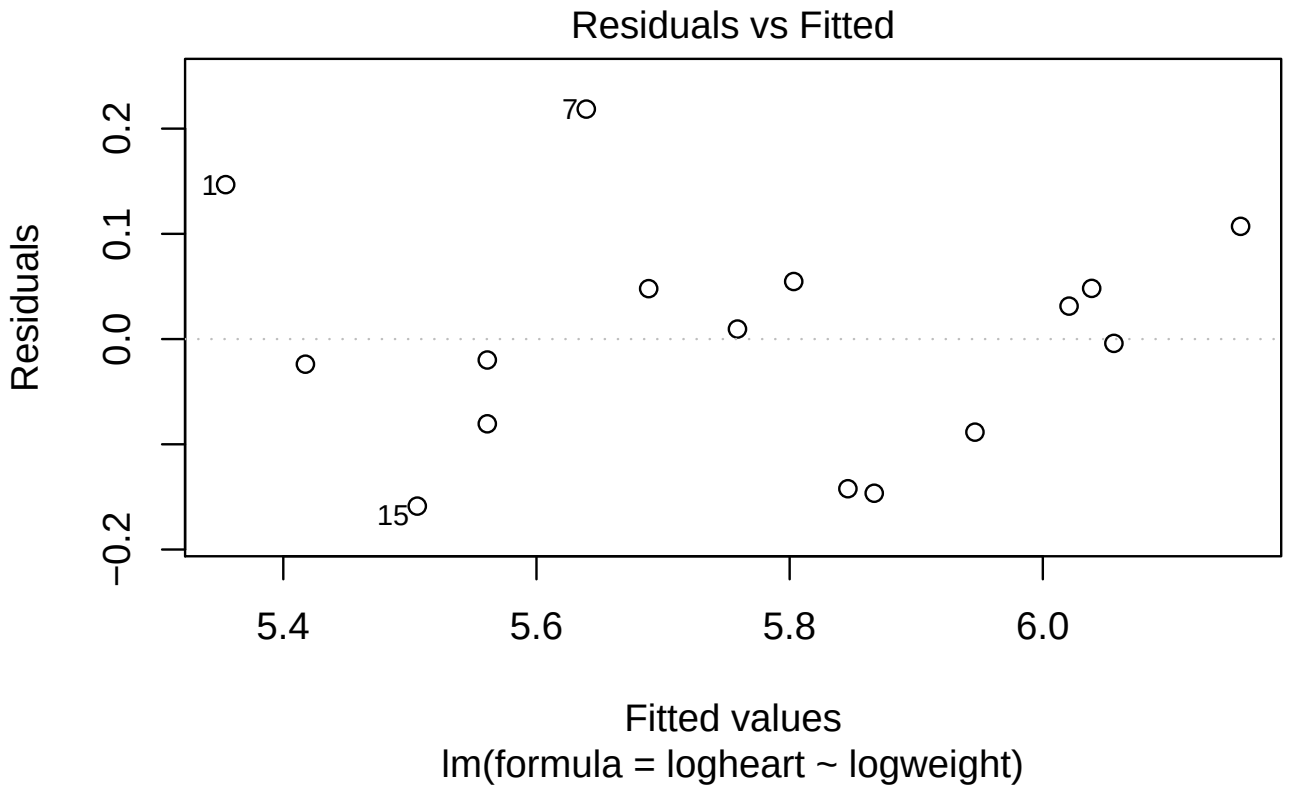
```
R> fm <- lm(logheart ~ logweight)
R> fm


Call:
lm(formula = logheart ~ logweight)

Coefficients:
(Intercept)    logweight
      1.325        1.133
```

Example: dolphins

```
R> summary(fm)

Call:
lm(formula = logheart ~ logweight)

Residuals:
      Min        1Q    Median        3Q       Max
-0.158735 -0.082494  0.002735  0.049814  0.218584

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   1.3245     0.5216   2.539   0.0236 *
logweight     1.1335     0.1330   8.523 6.51e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1107 on 14 degrees of freedom
Multiple R-Squared: 0.8384,        Adjusted R-squared: 0.8269
F-statistic: 72.63 on 1 and 14 DF,  p-value: 6.507e-07
```

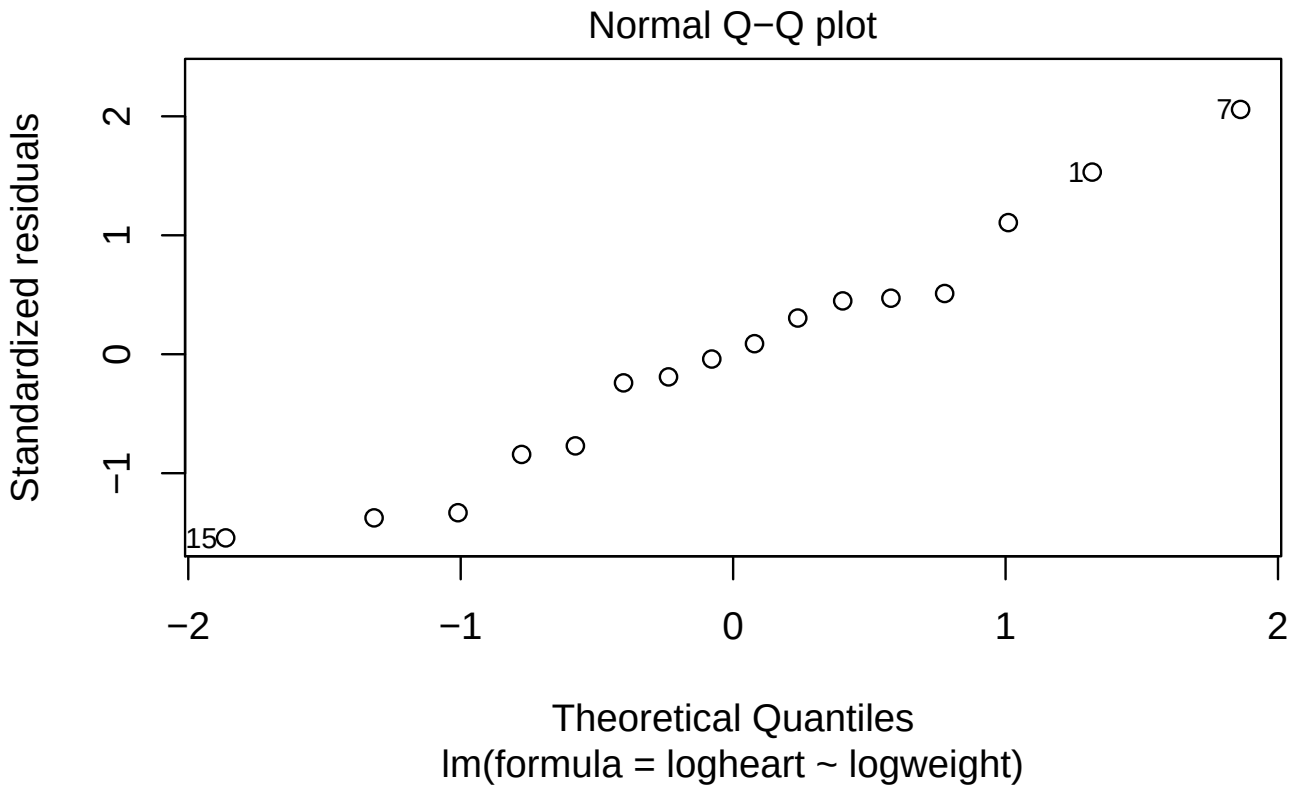
Example: dolphins

```
R> plot(fm)
```



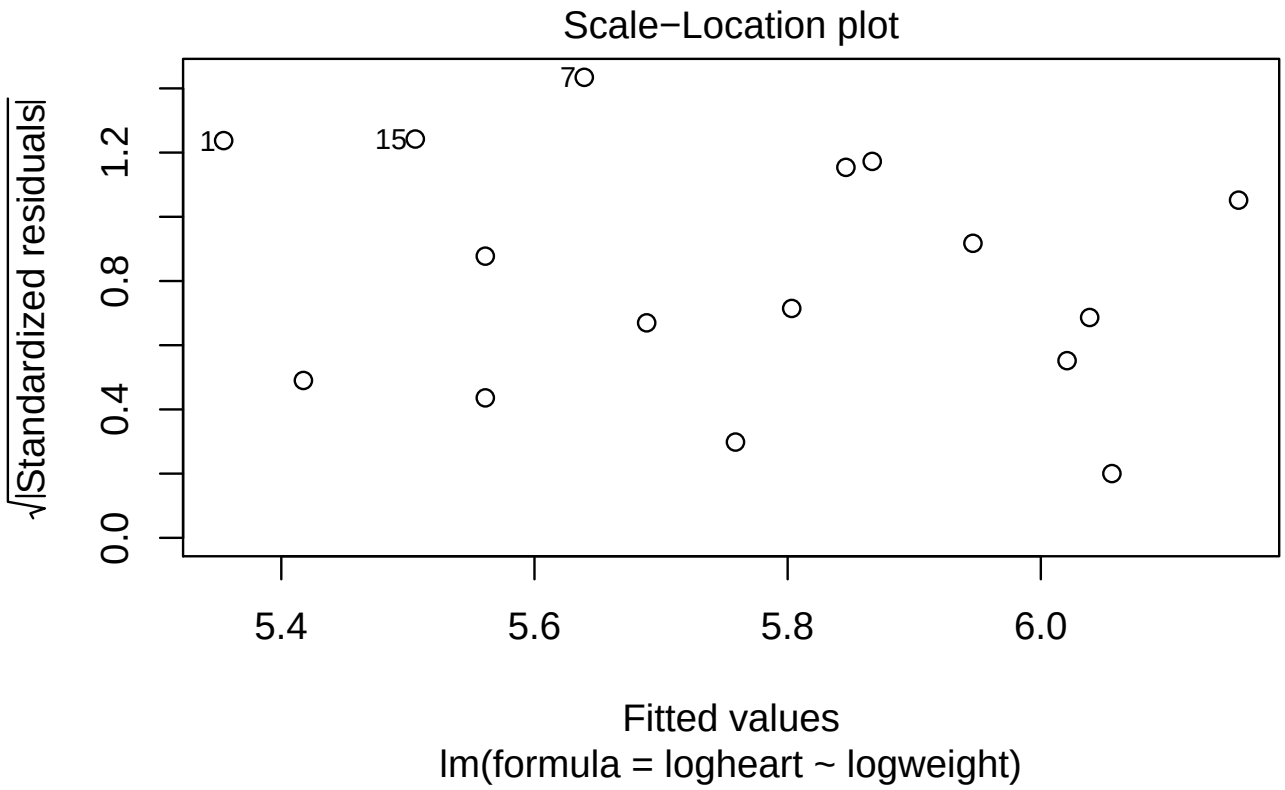
Example: dolphins

```
R> plot(fm)
```



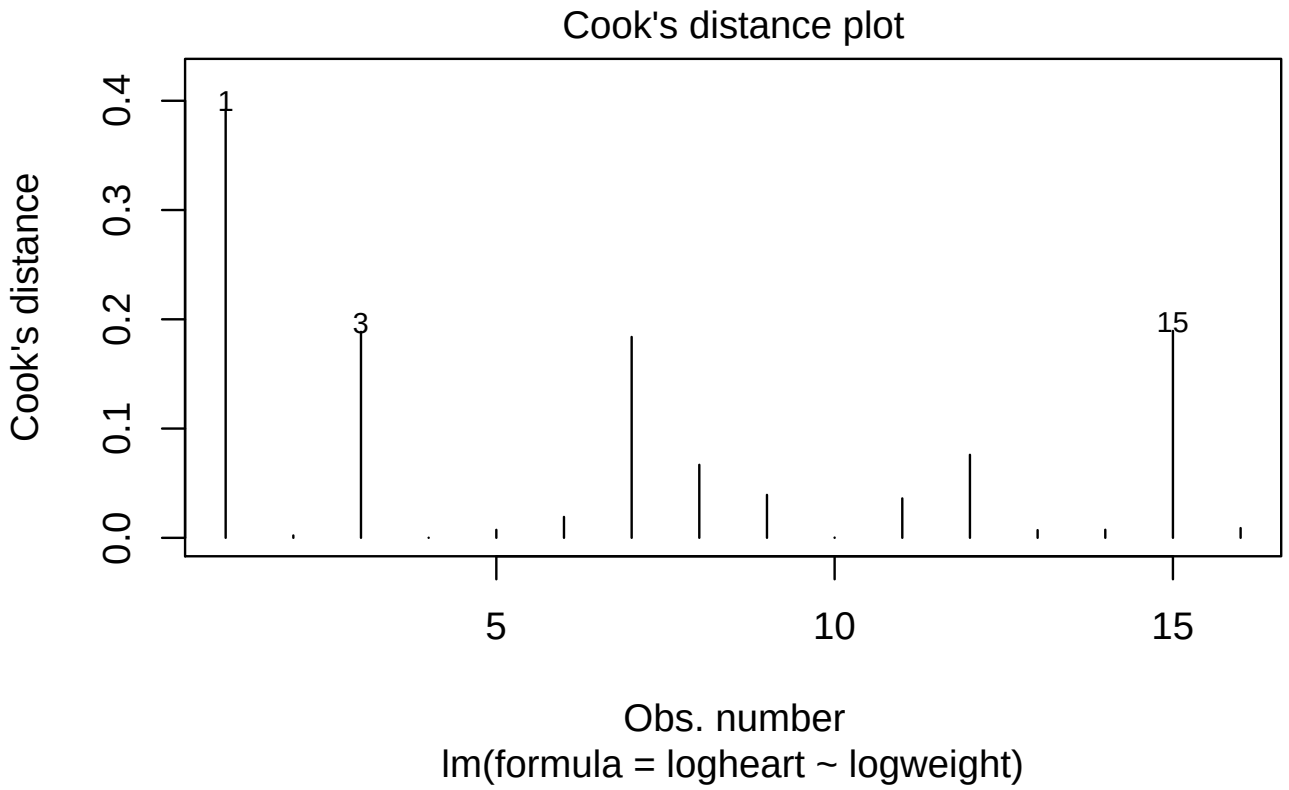
Example: dolphins

```
R> plot(fm)
```



Example: dolphins

```
R> plot(fm)
```

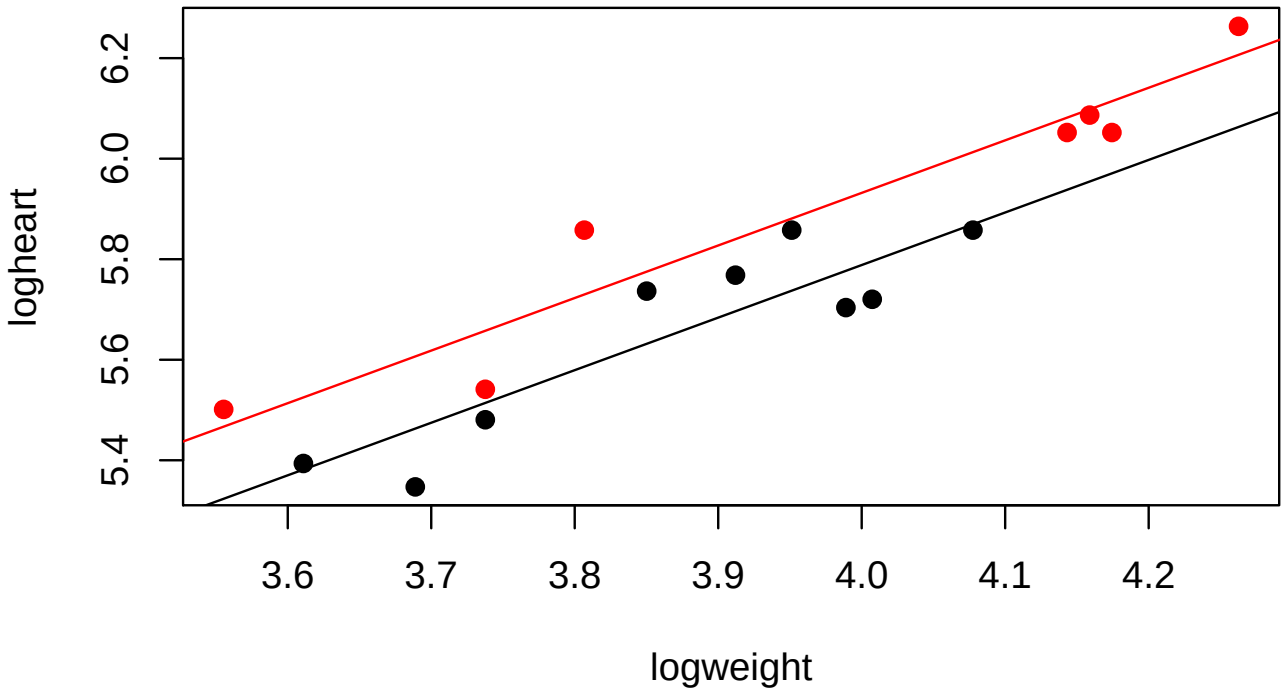


Example: dolphins

```
R> fm2 <- lm(logheart ~ logweight * species)
R> anova(fm2)


Analysis of Variance Table

Response: logheart
                  Df  Sum Sq Mean Sq  F value    Pr(>F)
logweight          1 0.89060 0.89060 112.6120 1.878e-07 ***
species            1 0.07581 0.07581   9.5854  0.009258 **
logweight:species  1 0.00095 0.00095   0.1204  0.734625
Residuals         12 0.09490 0.00791
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Example: dolphins

```
R> fm3 <- lm(logheart ~ logweight + species)
```



Example: spam mail

```
R> load("spam.rda")
R> index <- 1:nrow(spam)
R> ntest <- trunc(length(index)/3)
R> testindex <- sample(index, ntest)
```

Example: spam mail

```
R> load("spam.rda")
R> index <- 1:nrow(spam)
R> ntest <- trunc(length(index)/3)
R> testindex <- sample(index, ntest)


R> testset <- spam[testindex, ]
R> trainset <- spam[-testindex, ]
R> type.true <- testset[, 58]
R> testset <- testset[, -58]
```

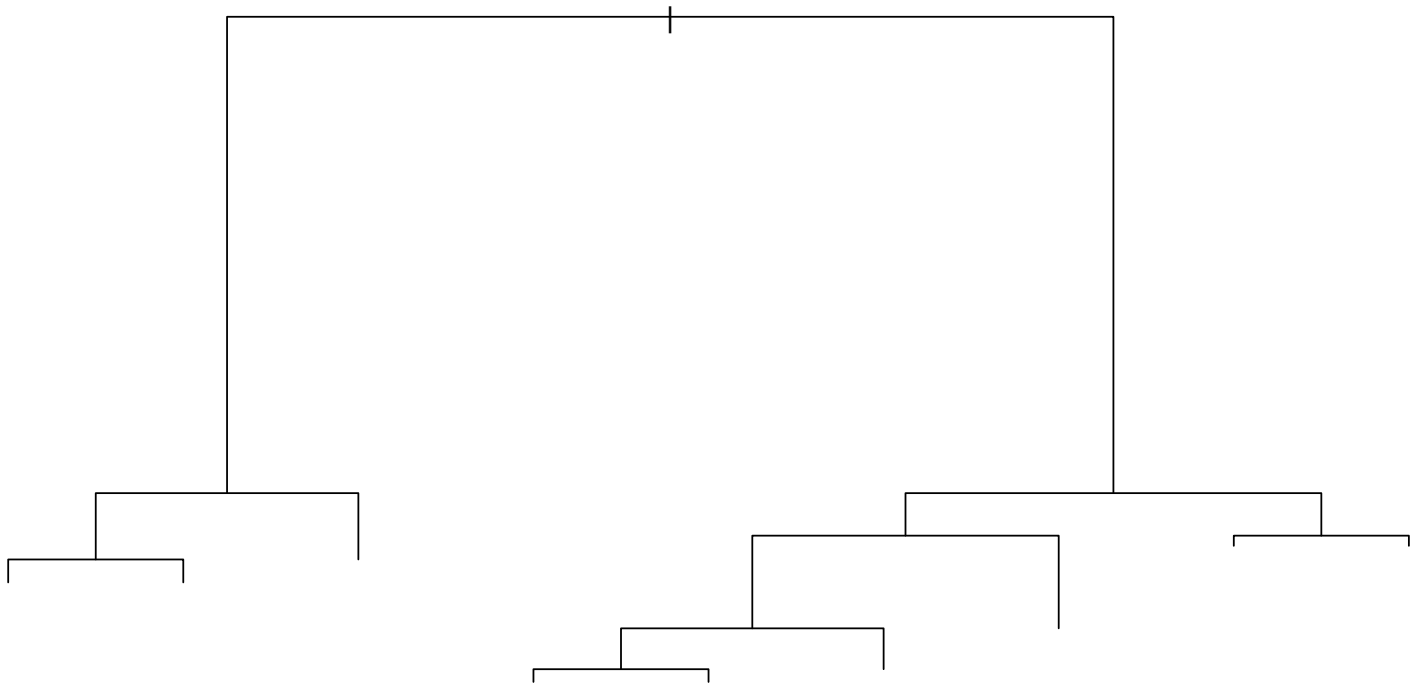
Example: spam mail

```
R> library(rpart)
R> rpart.model <- rpart(type ~ ., data = trainset)
```

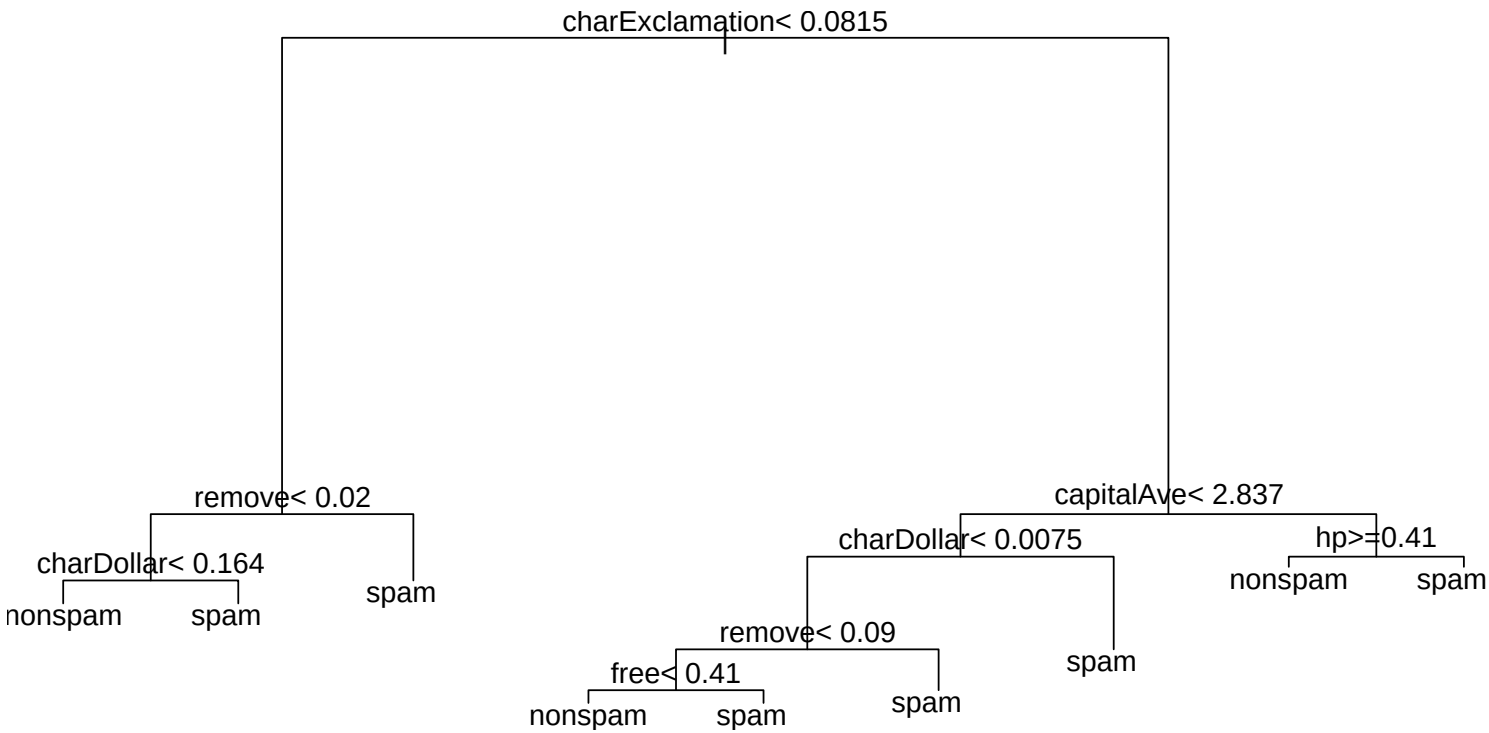
Example: spam mail

```
R> library(rpart)
R> rpart.model <- rpart(type ~ ., data = trainset)


R> plot(rpart.model)
R> text(rpart.model)
```

Example: spam mail



Example: spam mail



Example: spam mail

```
R> type.rpart <- predict(rpart.model, testset, type = "class")
```

Example: spam mail

```
R> type.rpart <- predict(rpart.model, testset, type = "class")

R> table(True = type.true, Pred = type.rpart)

         Pred
True      nonspam spam
  nonspam     858   59
  spam         90  526
```

Example: spam mail

```
R> type.rpart <- predict(rpart.model, testset, type = "class")

R> table(True = type.true, Pred = type.rpart)

         Pred
True      nonspam spam
  nonspam     858   59
  spam         90  526

R> (90 + 59)/ntest

[1] 0.09719504
```

Example: spam mail

```
R> library(randomForest)
R> rF.model <- randomForest(type ~ ., data = trainset)
R> type.rF <- predict(rF.model, testset)
```

Example: spam mail

```
R> library(randomForest)
R> rF.model <- randomForest(type ~ ., data = trainset)
R> type.rF <- predict(rF.model, testset)

R> table(True = type.true, Pred = type.rF)

         Pred
True      nonspam spam
  nonspam     892   25
  spam         57  559
```

Example: spam mail

```
R> library(randomForest)
R> rF.model <- randomForest(type ~ ., data = trainset)
R> type.rF <- predict(rF.model, testset)

R> table(True = type.true, Pred = type.rF)

         Pred
True      nonspam spam
  nonspam     892   25
  spam         57  559

R> (57 + 25)/ntest

[1] 0.05348989
```

Interfaces

Try not to reinvent the wheel by interfacing to other programs

- Execute code in virtually any other language (C, C++, Fortran, Java, Perl, ...).
- Use relational databases for storing your data (Oracle, ODBC, ...).
- Connections allow URLs, pipes or sockets to be treated like local files in many situations (read data, ...).

or interface from other programs

- R can be embedded in other languages or programs like Postgres databases or the Apache web server.

Packages

- R features a modern package management system (inspired by Linux package managers).
- Packages provide a convenient form of distributing data analytic methodology.
- An R source package typically contains
  - R code (and optionally C, C++ or Fortran code)
  - help pages with executable examples
  - datasets for examples
  - optional further documentation
- Packages can be automatically installed and updated over the Internet.

Packages

- The central network of package repositories CRAN currently provides more than 170 packages (and the list is growing fast). Binaries for Windows, Mac and several Linux distributions are provided by platform maintainers.
- You develop on your favorite platform, submit the package to CRAN (keeping the copyright), and the CRAN maintainers take care of all the other platforms, if possible.
- Several repositories can be used simultaneously, users can easily create their own package archives (and share them locally or globally).
- Many of the popular books on data analysis with S(-Plus) have corresponding R packages: MASS, boot, nlme, survival.

Packages

- Additionally
  - Omegahat:
    Web-based software, interfaces.

    http://www.omegahat.org/
  - Bioconductor:
    "Infrastructure" for the analysis and comprehension of genomic data.

    http://www.bioconductor.org/

Reproducible research

- Lawrence: "Online or Invisible?
  (*Nature*, 2001) shows that free online availability of papers increases the number of citations significantly.
- Equivalent "Online or Unused?" for data analytic techniques?
- We need to encourage (very strongly) writers of methodology to provide code that implements their methodology.
- The mathematical or theoretical description of a data analytic technique is a sufficient condition for being able to use it, but at least one implementation is necessary.
- Given the description of an excellent method and code for a good one, you choose . . . ?

Reproducible research

Are data and toolbox for favorite programming environment sufficient?

Reproducible research

Are data and toolbox for favorite programming environment sufficient?

Not really:

- Data preprocessing, weighting, meta-information and exact relation between different data sources, . . .
- Many analyses depend crucially on settings of hyperparameters or seed of random number generator.
- Recreation often hard (or at least time consuming) even given a complete description of the analysis.

Reproducible research

Are data and toolbox for favorite programming environment sufficient?

Not really:

- Data preprocessing, weighting, meta-information and exact relation between different data sources, . . .
- Many analyses depend crucially on settings of hyperparameters or seed of random number generator.
- Recreation often hard (or at least time consuming) even given a complete description of the analysis.

→ *code and text should be one tightly coupled bundle*

Reproducible research

Sweave: mixing LaTeX and S/R code.

Source code for next slide:

```
We compare the dolphins species using a Kruskal-Wallis test:

<<example>>=
kruskal.test(logheart ~ species)
@

which shows a significant difference between the two species.
```

Reproducible research

We compare the dolphins species using a Kruskal-Wallis test:

```
R> kruskal.test(logheart ~ species)

        Kruskal-Wallis rank sum test

data:  logheart by species
Kruskal-Wallis chi-squared = 3.8631, df = 1, p-value = 0.04936
```

which shows a significant difference between the two species.

Summary

R is a general purpose environment for data analysis with support for a wide variety of techniques including:

- linear and generalized linear models, nonlinear regression, random and mixed effects, multinomial logit
- classical parametric and nonparametric tests
- time series: ARIMA, GARCH, filter, spectrum
- clustering: hierarchical, partitioning, parametric mixtures, fuzzy
- regression and classification trees, bagging, random forests
- neural networks, support vector machines
- smoothing, generalized additive models, MARS
- bootstrap
- ...

Summary

- Using a language-based environment may feel uncomfortable if used to GUIs, but offers a lot more flexibility.
- Learn language while using the software.
- Full power of a modern programming language for implementing new ideas, working directly with statistical data types (nominal, ordinal, quantitative, missing values, . . .).
- Free software ideal for teaching, students do not need a license to use at home.

Summary

- Works on all common operating system platforms
- Open Source (GPL)

For more information...

http://www.R-project.org/

http://cran.R-project.org/