R: A Free Software Project in Statistical Computing

Achim Zeileis

http://statmath.wu-wien.ac.at/~zeileis/

Achim.Zeileis@R-project.org

Overview

- A short introduction to some letters of interest
  - R, S, Z
- Statistics and computing
  - applied vs. computational statistics vs. stat. computing
  - statistical software
- The R project
- Basic functionality
- Empirical application: diabetes in native populations
  - exploratory analysis
  - linear regression
  - tree models
- Packages
- Summary

Some letters

R is an interactive computational environment for data analysis, inference and visualization.

S is a language for data analysis and graphics, implemented in the commercial software S-PLUS and the open-source software R.

Z (aka Achim Zeileis) is a statistician at WU Wien spending a considerable share of his time using and developing R:

- for research,
- for applied data analysis,
- for course administration,
- for Web page generation, CD covers, mp3 administration . . . (but that's another story).

Some letters: 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.
- Released under the GPL (General Public License), similar to the open-source operating system Linux.
- Highly extensible through user-defined functions and a fast-growing list of add-on packages.
- Based on the S language but with a new underlying implementation.

Some letters: S

- S is a language for data analysis and graphics developed by John Chambers and co-workers at Bell Labs (of AT&T, now Lucent Technologies).

- Exclusively licensed (and eventually sold) to Insightful Corp. as the basis for the commercial statistics system S-PLUS.

- Award-winning language which “has forever altered the way how people analyze, visualize and manipulate data…” (ACM Software System Award 1998 to John Chambers).

Some letters: Z

Short bio:

1996–2000	University studies in Statistics, Universität Dortmund, Germany
2000	Dipl.-Stat. ($\sim$ M.Sc.) in Statistics
2000–2003	Research Assistant, SFB "Adaptive Information Systems and Modelling in Economics and Management Science", Wirtschaftsuniversität Wien, Austria
2003	Dr. rer. nat. ($\sim$ Ph.D.) in Statistics
since 2003	Assistant Professor, Department of Statistics & Mathematics, Wirtschaftsuniversität Wien, Austria

Some letters: Z

Interests as a student:

- Year 1: some interest in exploratory data analysis – tried to learn SAS – failed miserably – decided software is not for me.
- Year 2–3: learned a lot about theoretical concepts in statistics – did applied work only if necessary – used various commercial software packages: SPSS, Minitab, S-PLUS, GLIM, some SAS again, ...
- Year 4: needed software for seminars and thesis: implement concepts, run simulations, apply to real data – discovered open-source software R – switched from Windows to Linux, from Word to LaTeX, from everything else to R – never looked back.

Some letters: Z

Interests as a researcher:

- Statistical computing,
- Applied econometrics
  (in particular: testing, monitoring and dating of structural changes),
- Statistical learning
  (in particular: tree-based models),
- Visualization & significance testing.

Statistics and computing

A large spectrum of statistics involve the use of computers and software programs. Different parts of this spectrum with varying emphasis are the following.

Applied statistics:

- task: understand structures in data and the underlying mechanisms.
- required: software that provides appropriate statistical techniques for application to real data.
- tools: preprocessing, inference, visualization, reporting.
- extensibility: modify existing tools, customize analysis, define work flows, automatization.

Statistics and computing

Computational statistics:

- task: solve computing-intensive statistical problems.
- examples: difficult optimization problems, Markov chain Monte carlo algorithms.
- required: efficient implementation/programming language.

Statistics and computing

Statistical computing:

- task: turn theoretical concepts into software
- examples: implement a new statistical model so that it can be easily applied to new data sets.
- required: flexible software system with programming language.
- tools: object orientation, compiled code, re-usable components.

Of course, these areas are not well separated but overlapping! Software is always the bridge between theoretical concepts and statistical practice.

Statistics and computing: Software

There is a broad range of statistical software packages, some of the most popular are:

Excel most popular spreadsheet, only very basic statistical functionality, some programming possible in Visual Basic.

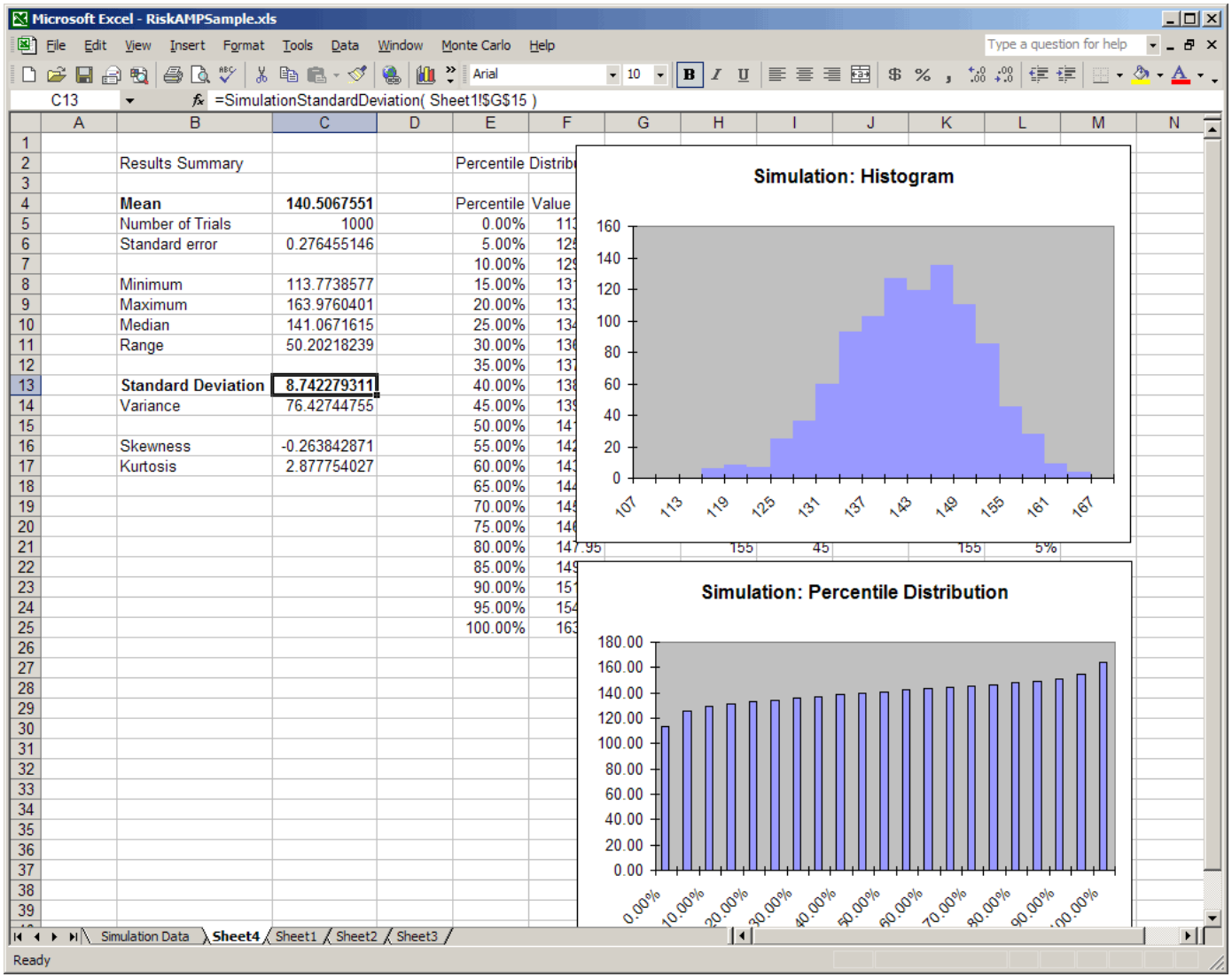
SPSS "statistical" spreadsheet, standard statistical tool box (some emphasis on social sciences), programming possible in SPSS language.

SAS comprehensive package with numerous interfaces, some emphasis on business solutions, programming in SAS macro language.

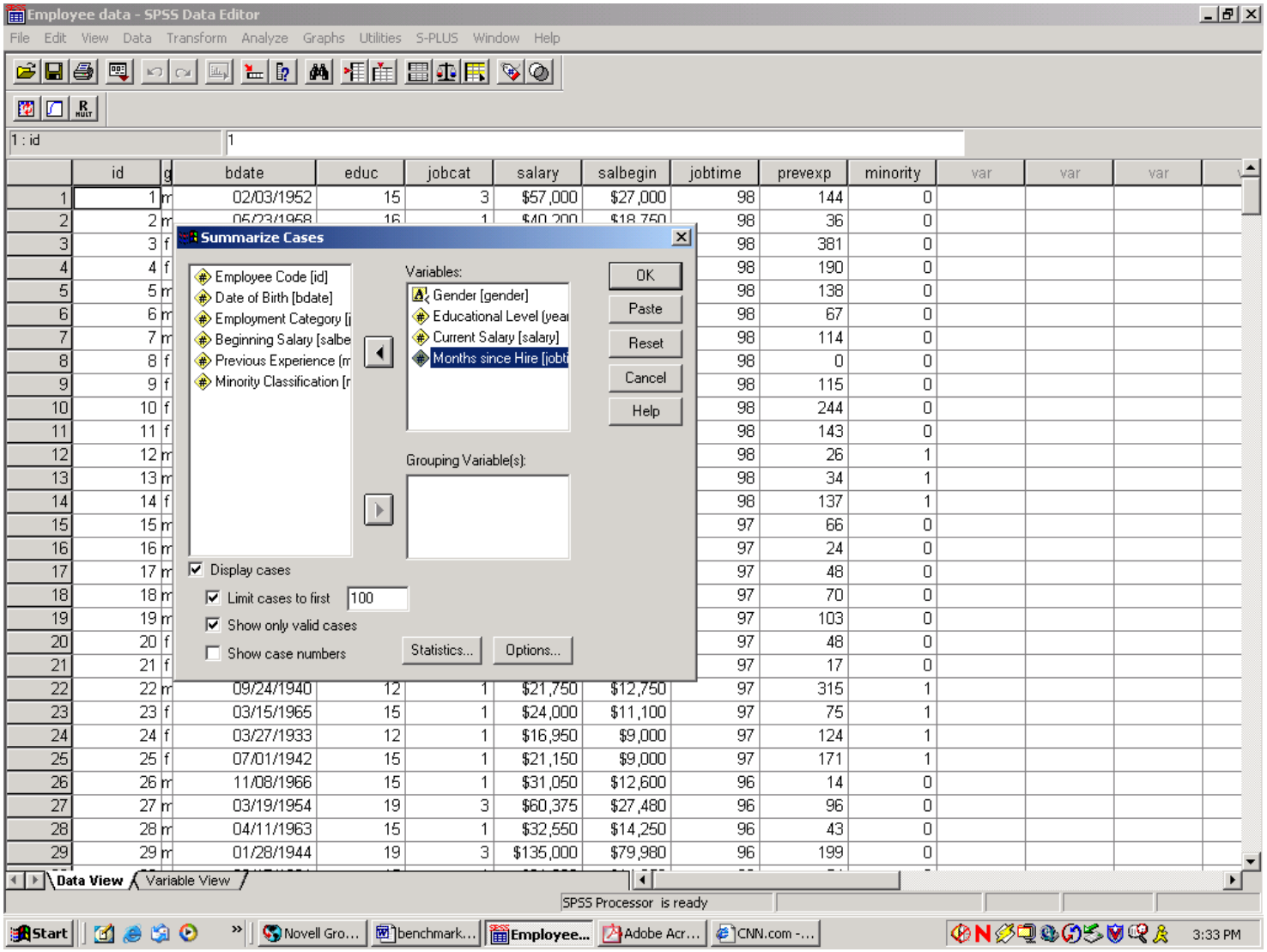
S-PLUS built on top of S, adds graphical user interface (GUI), spreadsheet, various extensions.

R command line interface (CLI), highly extensible, only specialized GUIs available.

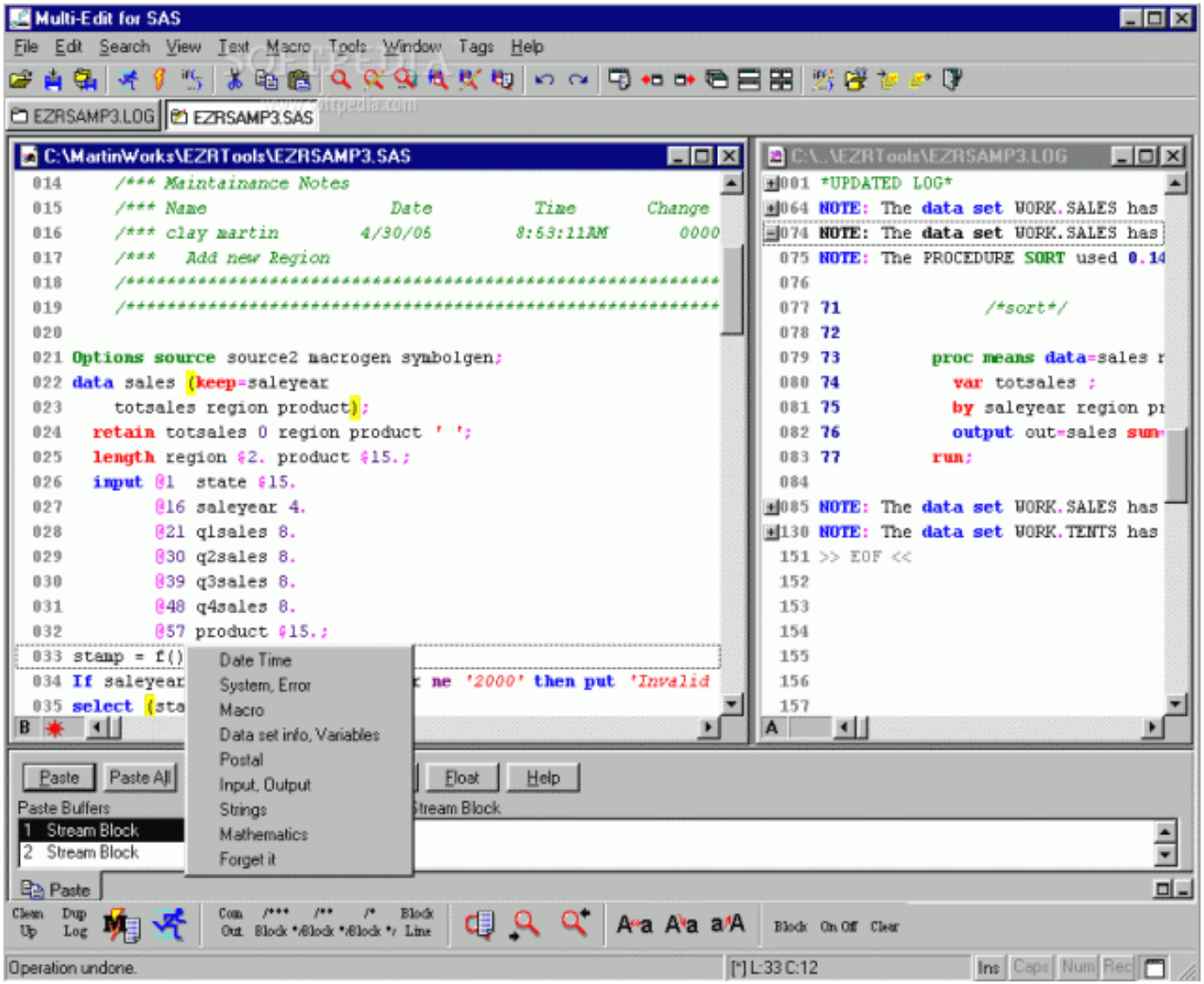
Statistics and computing: Excel



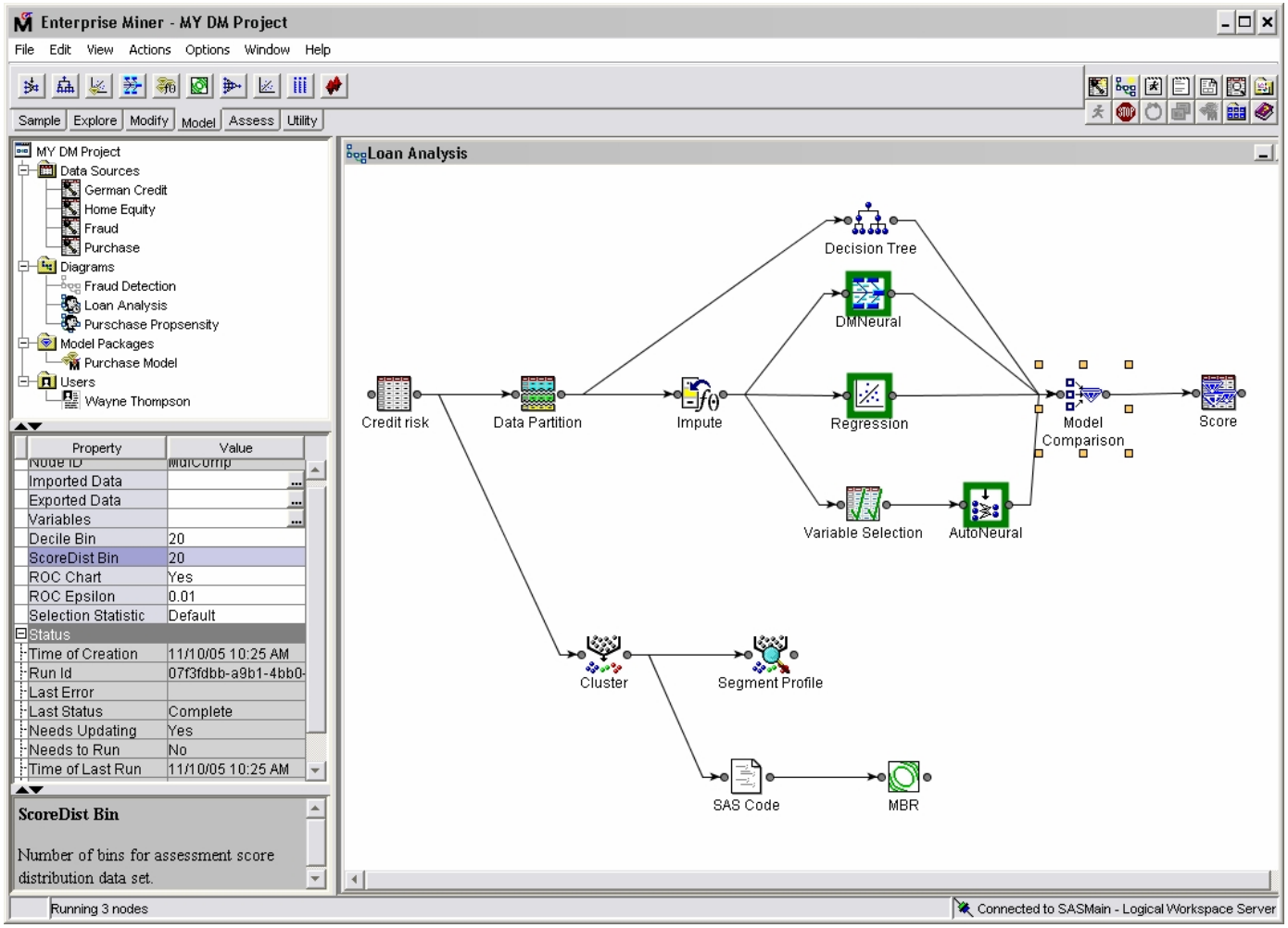
Statistics and computing: SPSS



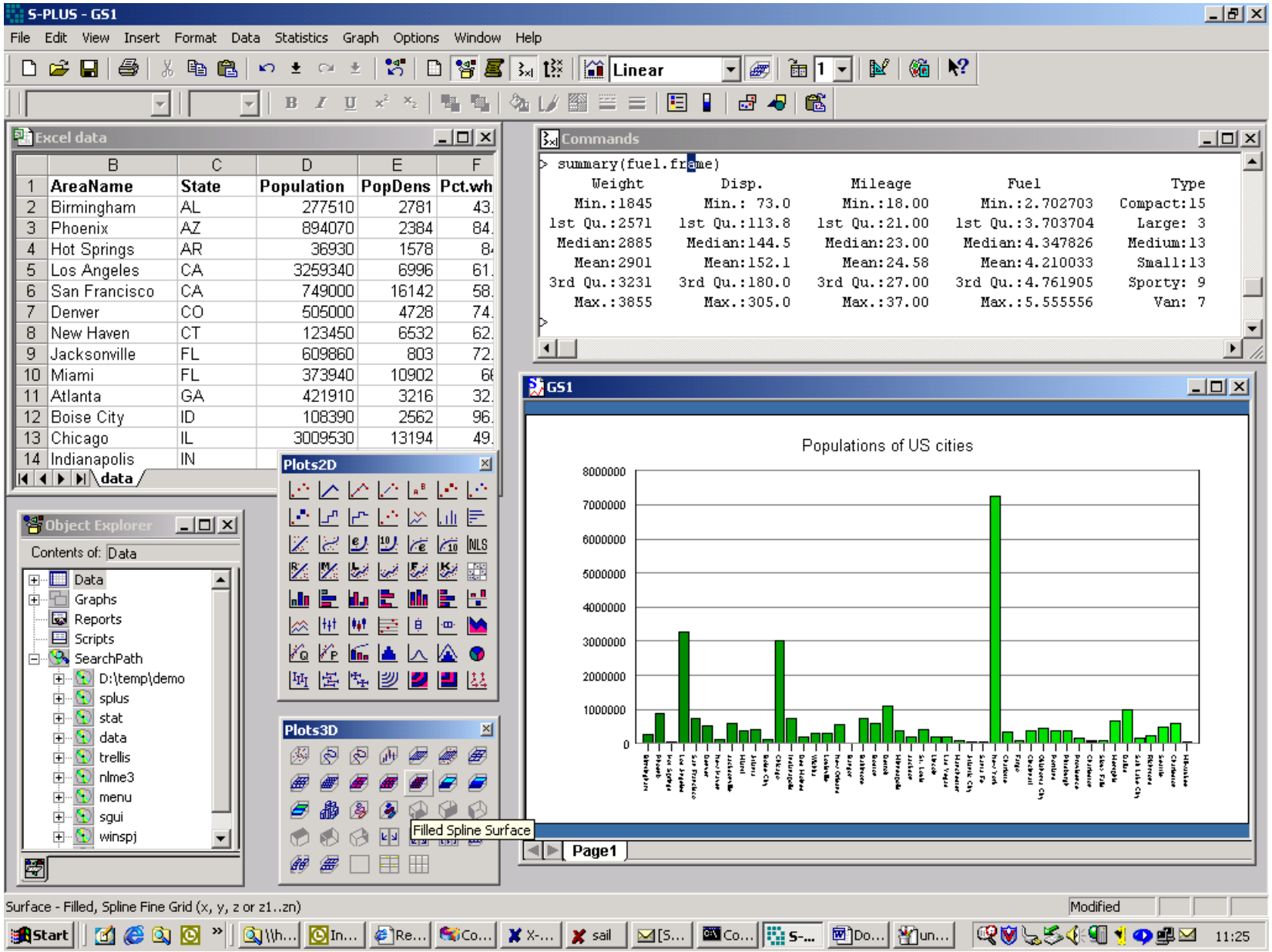
Statistics and computing: SAS



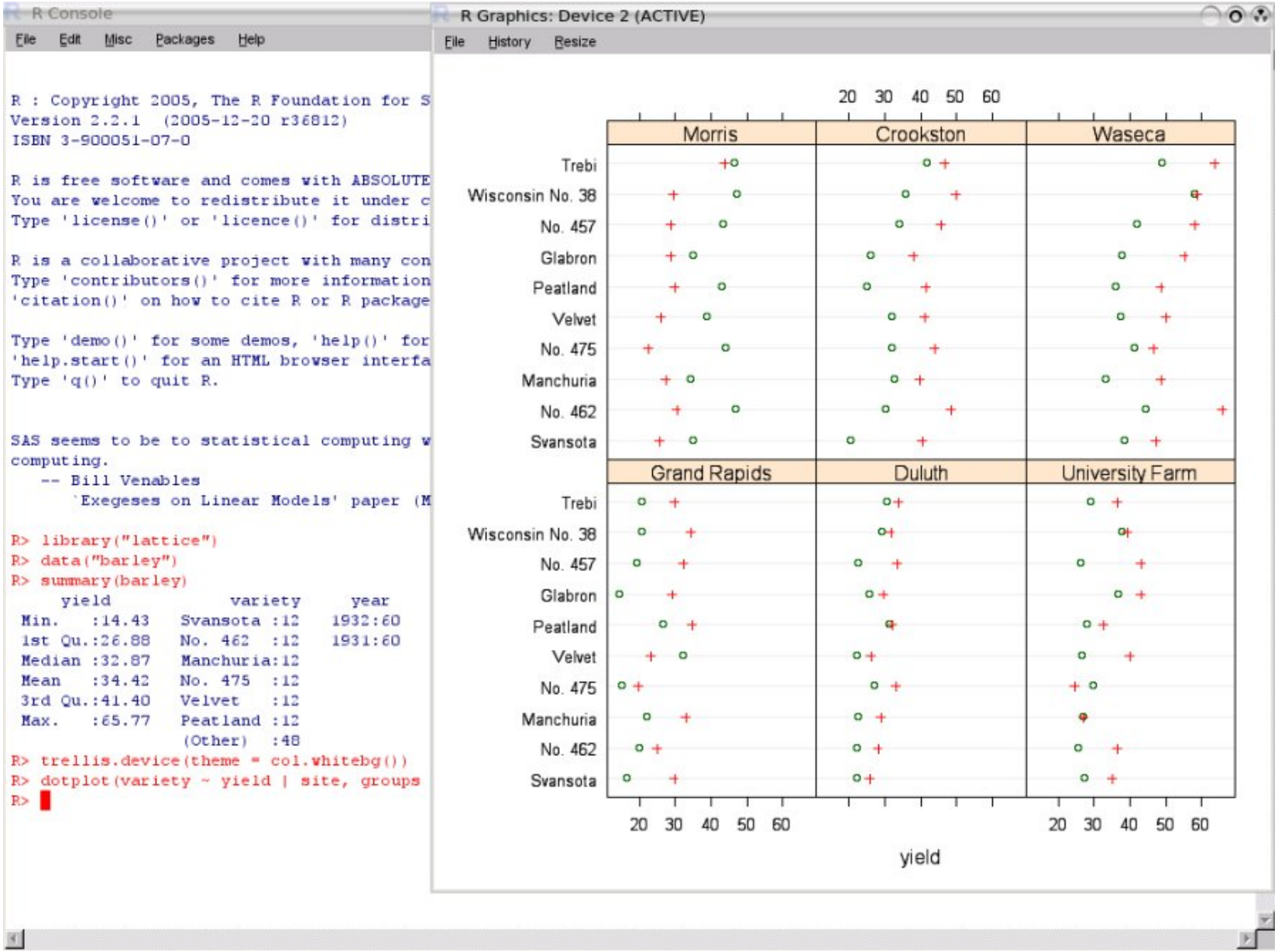
Statistics and computing: SAS



Statistics and computing: S-PLUS



Statistics and computing: R



Statistics and computing: Software

	GUI	CLI
learn	easy	harder
flexible	not really	very
repeat tasks	tiring	easy
reproduce results	hard	easy

GUIs are very popular with practitioners, but trained statisticians usually will have to do at least some programming at a certain point.

Reproducibility is crucial in scientific work, automatization in many corporate applications. Hence, all large packages offer some “command language” . . . S/R offers the most complete and modern programming language.

The R project

History of S:

1976: John Chambers and co-workers at Bell Labs begin work on a project that will become S (S1).
1981: Licenses for a portable Unix version of S outside Bell Labs (S2).
1988: Statistical software package S-PLUS based on S.
1992: Object orientation and statistical modeling toolbox included (S3).
1993: Exclusively licensed to MathSoft (now Insightful).
1998: New object orientation model introduced (S4).
2004: Sold to Insightful.

The R project

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.
2004: First useR! conference in Vienna.
2007: R-forge server launched.

The R project

Home of the R project is

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

where manuals, FAQs, links, and many other informations are available.

The R software (current version 2.4.0) can be obtained from

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

in source and binary form along with many extension packages.

The R project

Free software: Open-source software like R does not cost money (*free as in beer*). But it takes time to learn, and as time is money, this talk is mostly about free software in the sense of *free as in speach*.

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.

The R project

The base R system is maintained by 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, Duncan Murdoch, Paul Murrell, Martyn Plummer, Brian Ripley, Duncan Temple Lang, Luke Tierney, and Simon Urbanek.

But R would not be what it is without the support of a very large and active user community around the world, both in academia and the industry, who contributed by donating code, bug fixes, documentation, packages, discussion on the mailing lists

The R project

The R user community communicates via means of mailing lists (R-help, R-devel, R-bugs, SIGs) where questions can be asked, problems discussed, bugs reported, solutions suggested.

Using the R language interactively allows for a smooth transition from *using* R to *developing* in R. Bundles of new functions, manual pages, data sets, examples, demos, documentation can be effectively shared in the R community via means of CRAN packages.

Basic functionality

- an oversized pocket calculator.
- matrix-based language.
- full-featured programming language: (statistical) data structures, flow control, object orientation, interfaces to other languages, operating system interaction.
- statistical toolbox: exploratory data analysis, inference, (generalized) linear models, multivariate analsyis, time-series analysis, ...
- production-quality graphics.

Basic functionality

```
R> 1 + 1

[1] 2

R> 2^3

[1] 8

R> x <- c(2, 7)
R> x

[1] 2 7

R> 1/x

[1] 0.5000000 0.1428571
```

Basic functionality

```
R> y <- matrix(c(1, 2, 3, 4), ncol = 2)
R> y

     [,1] [,2]
[1,]    1    3
[2,]    2    4

R> y %*% x

     [,1]
[1,]   23
[2,]   32

R> solve(y)

     [,1] [,2]
[1,]   -2  1.5
[2,]    1 -0.5
```

Basic functionality

Fundamental language design principle:

Everything in R is an object.

Every object has a class (e.g., numeric, factor, function, . . .) and *methods* for *generic functions* are provided (or can be defined).

Typically, methods for `print()`, `summary()`, or `plot()` are offered.

Basic functionality

R is a functional language. Functions can in principle take arbitrary objects as their arguments and return arbitrary objects.

Not only vectors and matrices can be supplied and returned/printed!

Objects can be complex and capture all necessary information: e.g., time series, fitted linear models, ...

Even functions can be passed as an argument to (or returned by) another function.

Diabetes in native populations

SPIEGEL ONLINE WISSENSCHAFT

NACHRICHTEN | VIDEOS | ENGLISH | FORUM | SPIEGEL DIGITAL | ABOS + SHOP

Home | Politik | Wirtschaft | Panorama | Sport | Kultur | Netzwelt | **Wissenschaft** | UniSPIEGEL | SchulSPIEGEL | Reise | Auto

Nachrichten > Wissenschaft > Mensch & Technik

13. November 2006 Druckversion | Versenden | Leserbrief

MODERNE LEBENSART

Folgen der Fettsucht bedrohen Urvölker

Ureinwohner in Amerika, Asien und Australien passen sich immer stärker dem modernen Lebensstil an. Die Folge: Fettleibigkeit und Diabetes verbreiten sich rasant. Experten warnen inzwischen davor, dass ganze Gruppen von Ureinwohnern ausgelöscht werden könnten.

Source: http://www.spiegel.de/ accessed 2006-11-13.

Diabetes in native populations

Data set from the UCI repository of machine learning databases: A population of women who were at least 21 years old, of Pima Indian heritage and living near Phoenix, Arizona, was tested for diabetes according to WHO criteria. The preprocessed data comprise 724 observations of the following variables:

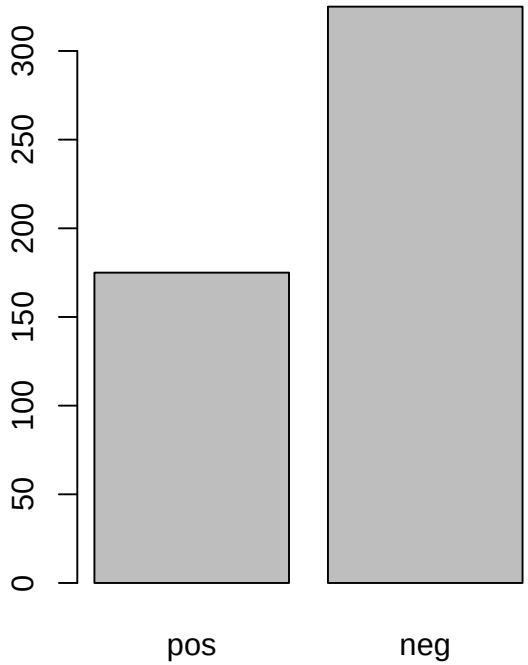
Variable	Description
`pregnant`	number of pregnancies
`glucose`	glucose concentration in an oral glucose tolerance test
`pressure`	blood pressure (mm Hg)
`mass`	body mass index (kg/m^2)
`pedigree`	diabetes pedigree function
`age`	age in years
`diabetes`	test for diabetes

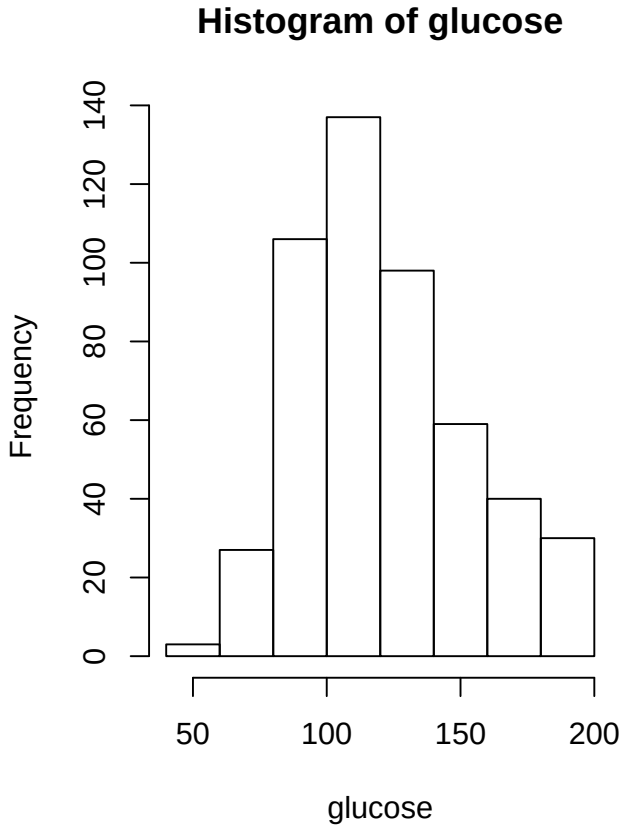
Diabetes: Exploratory analysis

```
R> class(glucose)

[1] "numeric"

R> summary(glucose)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   44.0    99.0   116.0   121.6   142.2   197.0

R> class(diabetes)

[1] "factor"

R> summary(diabetes)

pos neg
175 325
```

Diabetes: Exploratory analysis

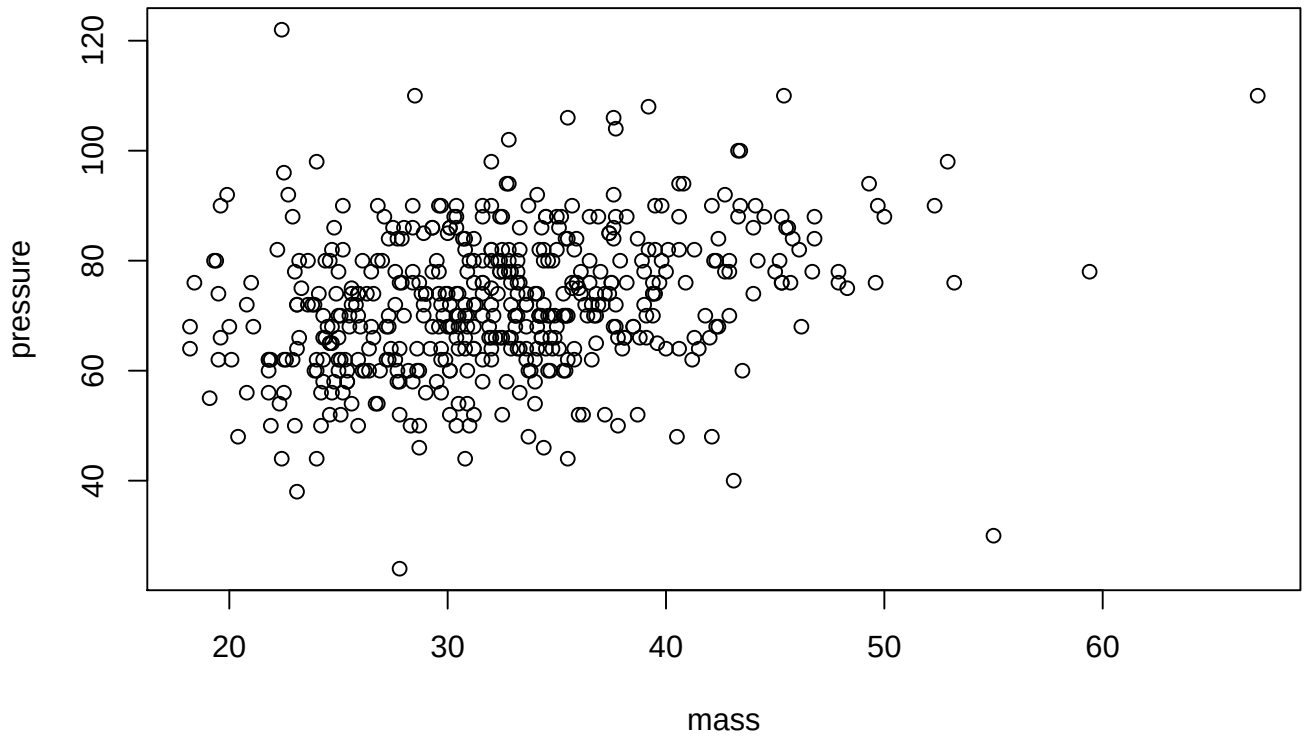
```
R> plot(diabetes)
R> hist(glucose)
```





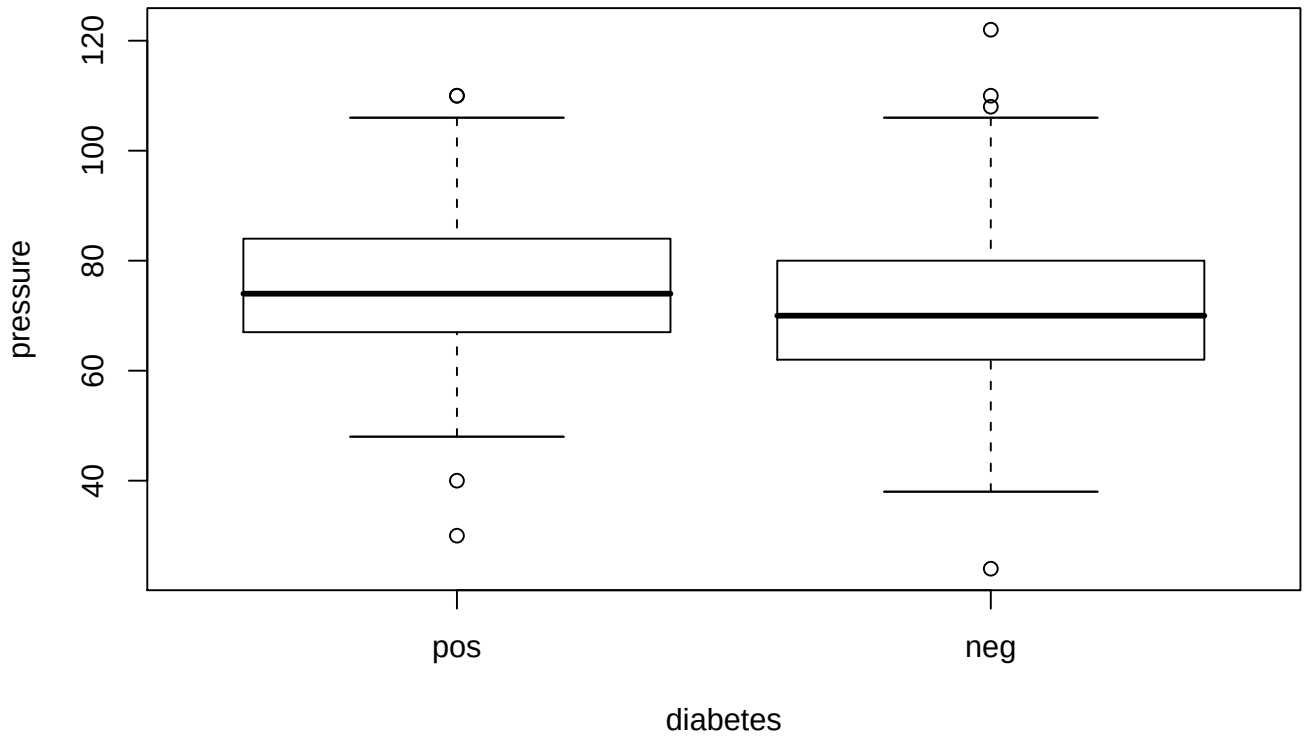
Diabetes: Exploratory analysis

```
R> plot(pressure ~ mass, data = pid)
```



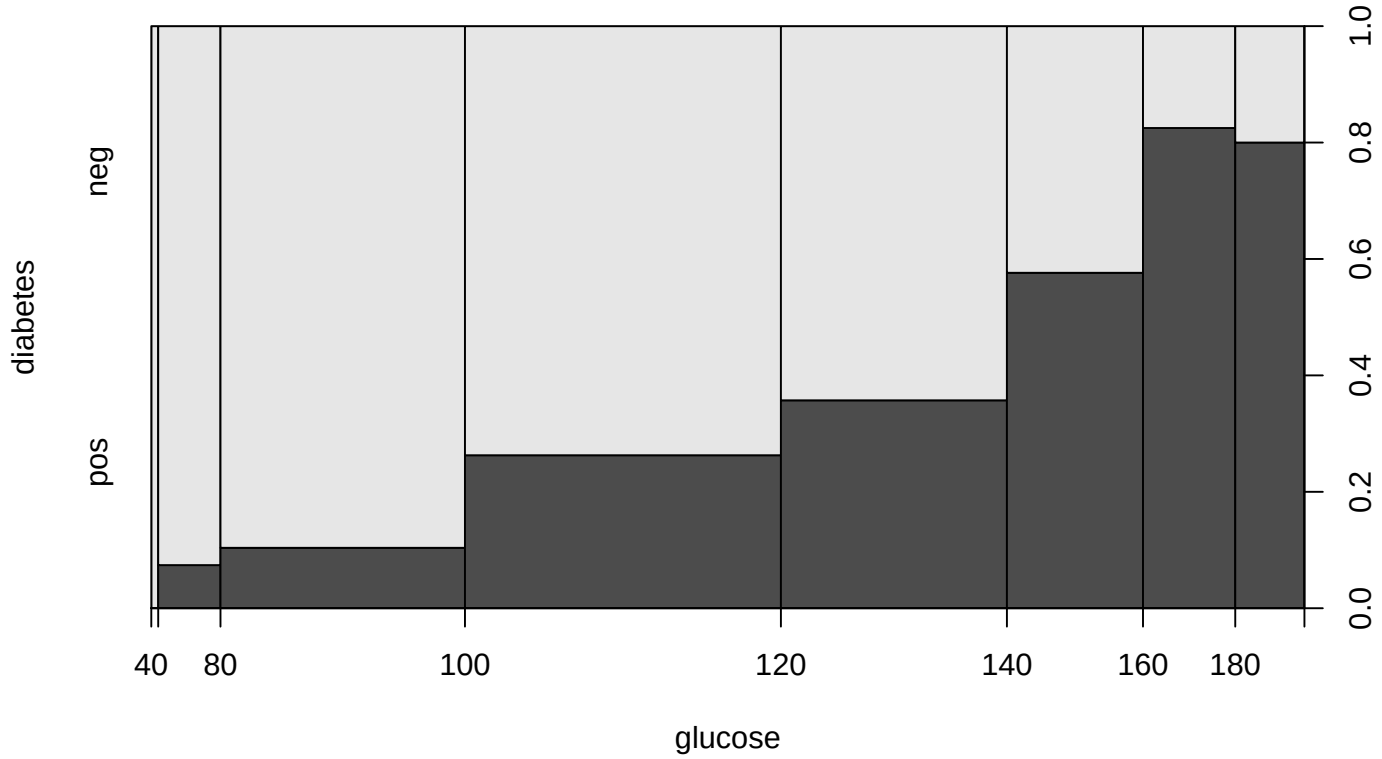
Diabetes: Exploratory analysis

```
R> plot(pressure ~ diabetes, data = pid)
```



Diabetes: Exploratory analysis

```
R> plot(diabetes ~ glucose, data = pid)
```



Diabetes: Linear regression

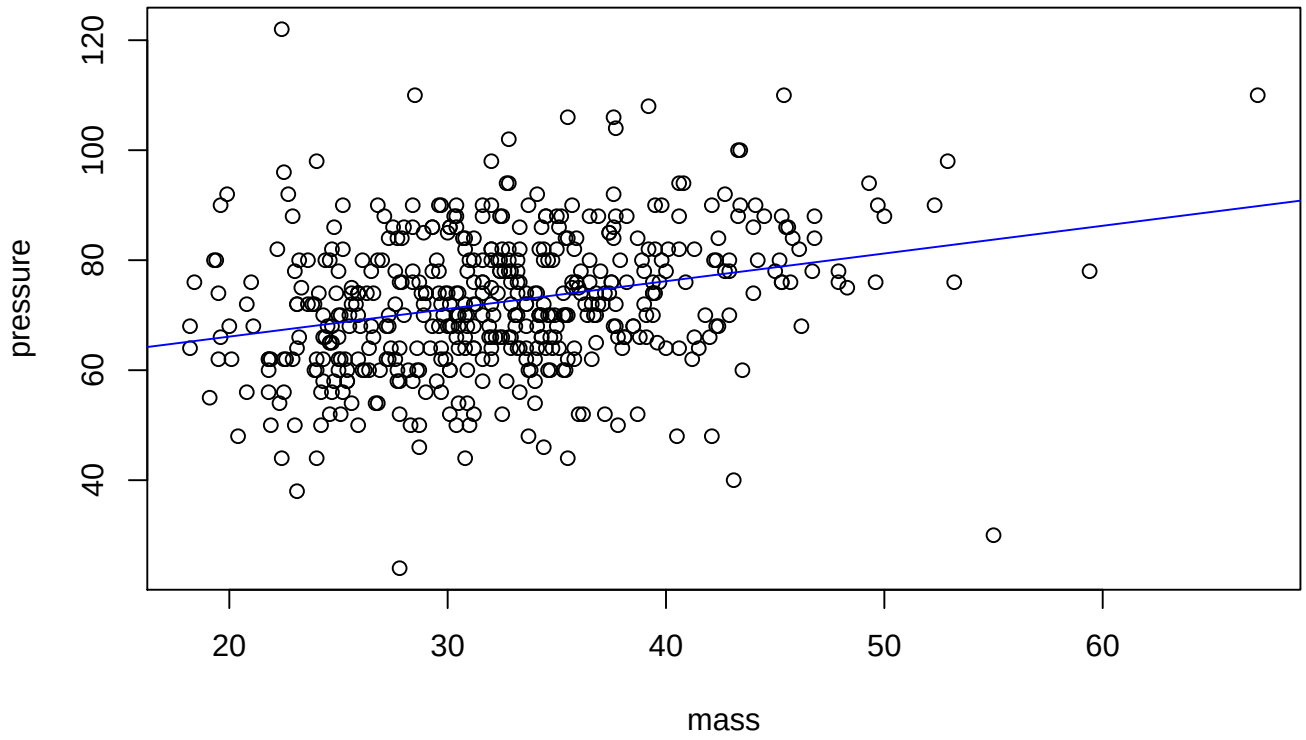
```
R> pid_lm <- lm(pressure ~ mass, data = pid)
R> pid_lm

Call:
lm(formula = pressure ~ mass, data = pid)

Coefficients:
(Intercept)         mass
    56.0272       0.5038


R> class(pid_lm)

[1] "lm"
```

Diabetes: Linear regression

```
R> summary(pid_lm)

Call:
lm(formula = pressure ~ mass, data = pid)

Residuals:
     Min       1Q   Median       3Q      Max
-53.7381  -7.6641  -0.5782   7.7252  54.6869

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 56.02724    2.56295  21.860  < 2e-16 ***
mass         0.50383    0.07723   6.523 1.69e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 12.25 on 498 degrees of freedom
Multiple R-Squared: 0.07873,        Adjusted R-squared: 0.07688
F-statistic: 42.56 on 1 and 498 DF,  p-value: 1.691e-10
```

Diabetes: Linear regression

```
R> plot(pressure ~ mass, data = pid)
R> abline(pid_lm, col = "blue")
```



Diabetes: Tree models

Tree models employ a simple recursive partitioning algorithm:

- Select the explanatory variable x most associated with the dependent variable y, or stop.
- Split the data into sub-groups which provide the best separation of y.
- Repeat the procedure recursively in each of the new sub-groups.

Tree-growing algorithms might differ in choice of association measure, stopping criterion, split criterion or sub-group selection. All have in common that they result in a tree whose leafs (terminal nodes) can be used for predictions.

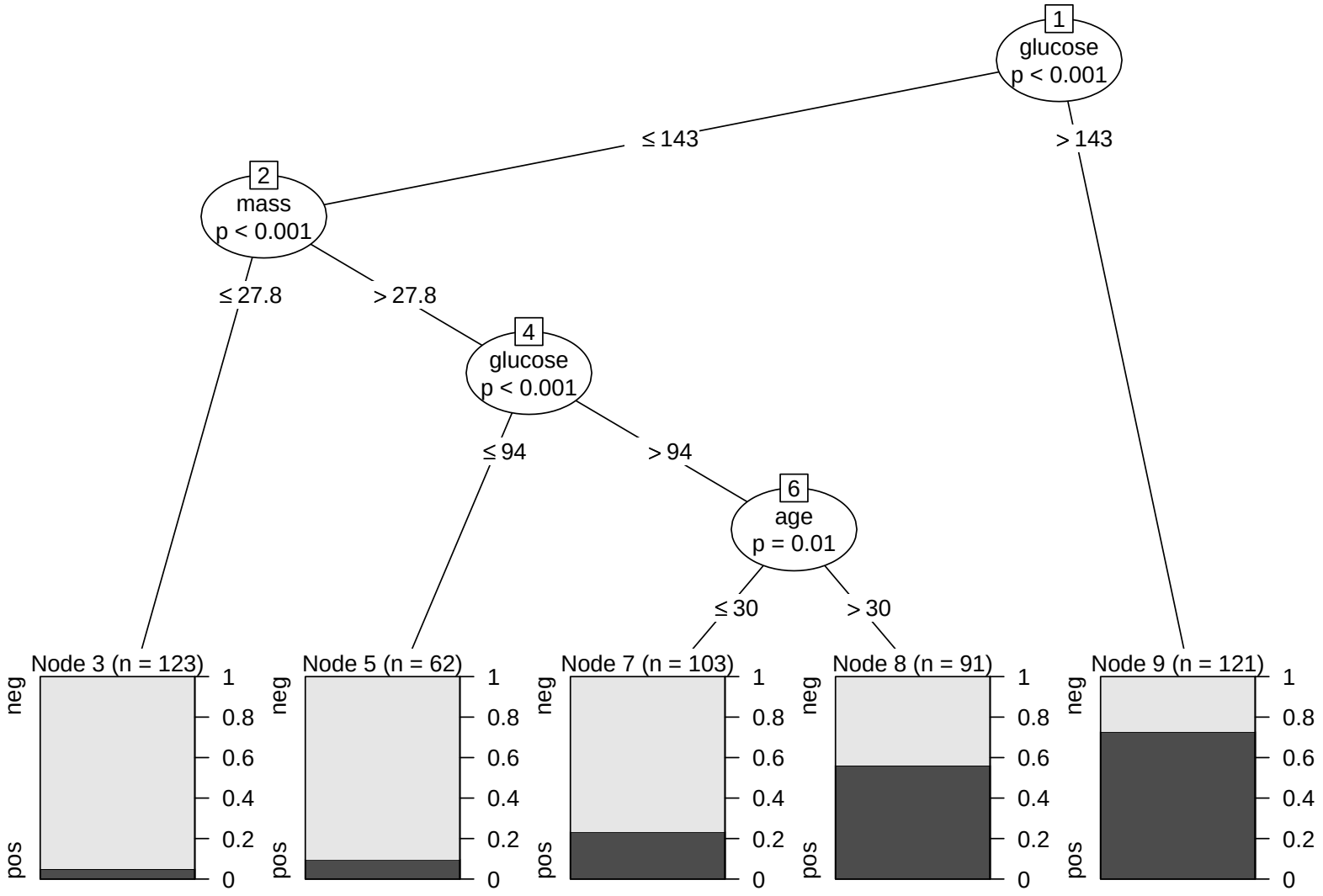
Diabetes: Tree models

One of these algorithms is CTree (conditional inference trees) provided by the function `ctree()` from package **party**.

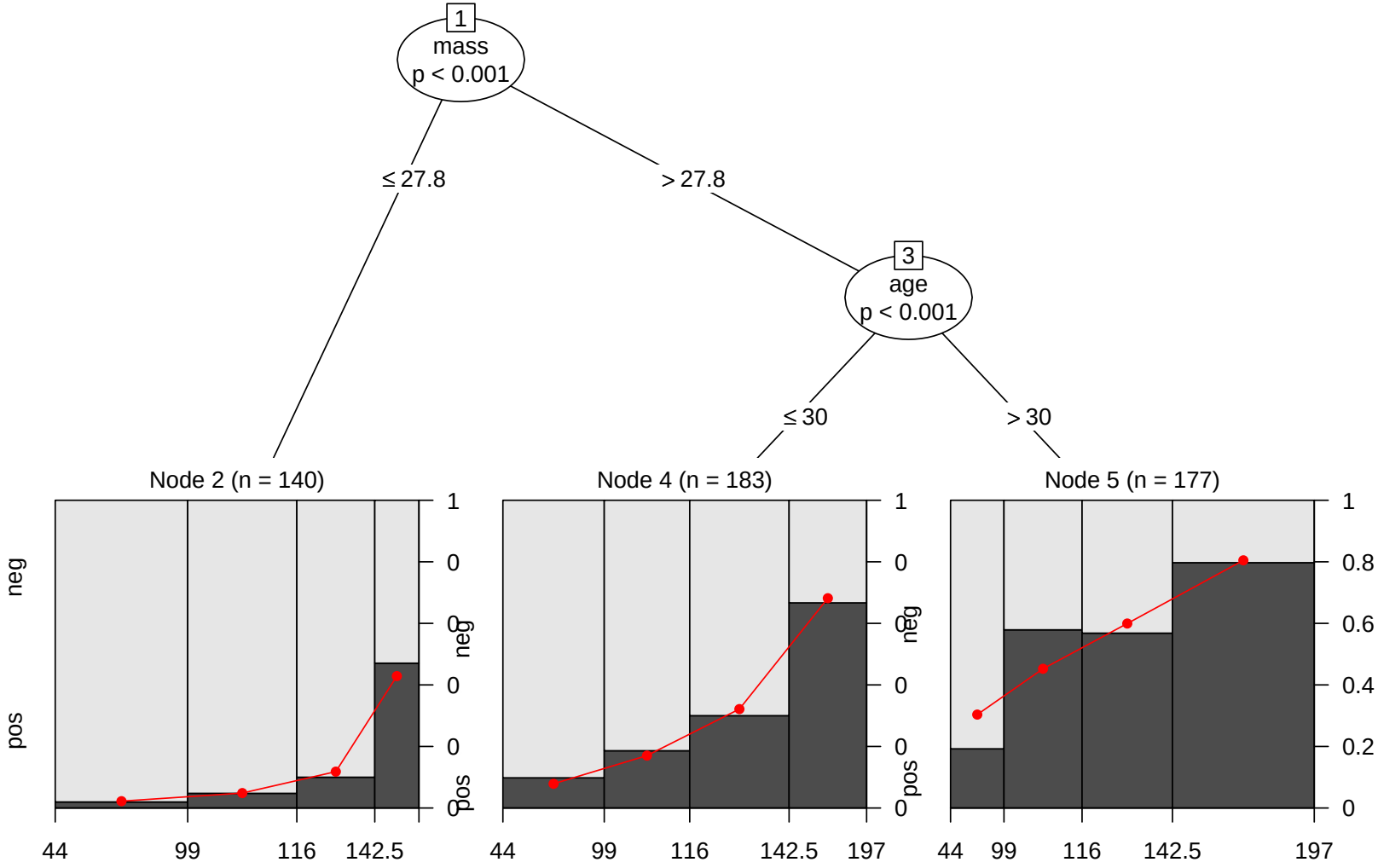
```
R> library("party")
R> pid_ctree <- ctree(diabetes ~ glucose + pregnant + pressure +
+     mass + pedigree + age, data = pid)
R> plot(pid_ctree)
```

Another alternative are model-based trees (provided by `mob()`) that can incorporate parametric models in the nodes, e.g., generalized linear models.

Diabetes: Tree models



Diabetes: Tree models



Diabetes: Tree models

Fitted models can be used for prediction on new data, using the generic function `predict()`

```
R> pred_ctree <- predict(pid_ctree, newdata = pid2)
```

and we can compare predictions with true observations

```
R> table(true = pid2$diabetes, pred = pred_ctree)
```

```
      pred
true   pos neg
  pos   54  20
  neg   37 113
```

leading to misclassification rates of 25.4% and 22.3%, respectively.

Packages

One of the core strengths of R is its extensibility. Users of R can become developers very easily and write their own packages.

A package can contain not only R code but also source code in other languages (e.g., C, C++, FORTRAN, Java), documentation, data, ...

CRAN currently hosts more than 800 packages (and counting). They can be automatically installed an updated over the internet.

Packages

Installation can be done easily from within R, e.g., by

```
install.packages("party")
```

or better make that

```
install.packages("party", dependencies = TRUE)
```

because **party** relies on several other packages. After installation the package can be simply loaded via

```
library("party")
```

making its functions available.

Summary

- R is a general purpose environment for data analysis and graphics with support for a wide variety of statistical techniques.
- 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.
- Open source (GPL), ideal for teaching.
- Works on all common operating system platforms.
- Reporting and replication easily via combination with LaTeX.

Summary

More information on

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