Introduction to R

Round I

Henning Rust and Olivier Mestre

E2C2 Summer School, Comorova, September 2007

Outline

Introduction

What is R?

The R-Project says:

- a language and environment for statistical computing and graphics
- similar to/a dialect of S
  (John Chambers, since 1976 at Bell Labs)
- provides a wide variety of statistical and graphical techniques
- produces well-designed publication-quality plots, including mathematical symbols and formulae

Why R?

- *To turn ideas into software, quickly and faithfully* (John Chambers on S)
- freely available (a GNU project, GPL) `www.r-project.org`
- community driven process with packages supplied by researchers
- over 1100 packages available via CRAN
- high level statistical functions
- extensive plotting facilities
- available for many plattforms: Linux, Unix, Windows, MacOS, etc.

Using R

```
R version 2.5.0 (2007-04-23)
Copyright (C) 2007 The R Foundation for Statistical Computing
ISBN 3-900051-07-0

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

Using R

R can be used

- interactively
  - using the R console or
  - an editor environment (e.g. Emacs, Eclipse, Tinn-R, etc.) or
  - a GUI (as e.g. for Windows and Mac)
- with script files started
  - from the R console or
  - from a shell

Tip: try using R together with Emacs and the ESS package

Getting Help

- HTML: `> help.start()`
  efficient introduction: "An Intoduction to R"
- Books
  `http://www.R-project.org/doc/bib/R-publications.html`
- FAQ
  `http://cran.r-project.org/faqs.html`
- Mailing lists
  `http://www.R-project.org/mail.html`
  on nicely posed questions, answers will be given quickly, frequently from R core team members
- Wiki
  `http://wiki.r-project.org/rwiki/`

Course Structure

Block A

- We present some concepts
- you listen

Block P

- You solve some problems
- We take a break
- Slides and data files are in "Introduction to R" on "Desktop"

Course Structure

Block A

- We present some concepts
- you listen

Block P

- You solve some problems
- We take a break
- Slides and data files are in “Introduction to R” on “Desktop”

Course Structure

```
while(!done){
```

Block A

- We present some concepts
- you listen

Block P

- You solve some problems
- We take a break
- Slides and data files are in “Introduction to R” on “Desktop”

}

Basic Concepts

Commands

Expression

```
> 1 - pi + exp(1.7)
[1] 3.332355
```

Assignments

- `a <- 6` or `a = 6`
- `b <- a <- 6`

Functions

- `writePaper <- function(idea,authors,journal){ ... }`

Commands

Expression

```
> 1 - pi + exp(1.7)
[1] 3.332355
```

Assignments

- `a <- 6` or `a = 6`
- `b <- a <- 6`

Functions

- `writePaper <- function(idea,authors,journal){ ... }`

Commands

Expression

```
> 1 - pi + exp(1.7)
[1] 3.332355
```

Assignments

- `a <- 6` or `a = 6`
- `b <- a <- 6`

Functions

- `writePaper <- function(idea,authors,journal){ ... }`

Vectors

Scala

- vector of length 1
- `a <- 29`

Vector

- numeric `a <- c(1,5,2,8)`
- logical `b <- c(TRUE,TRUE,FALSE)` or `b <- c(T,T,F)`
- character `d <- c("red" ,"green","blue")`

Vectors

Scala

- vector of length 1
- `a <- 29`

Vector

- numeric `a <- c(1,5,2,8)`
- logical `b <- c(TRUE,TRUE,FALSE)` or `b <- c(T,T,F)`
- character `d <- c("red" ,"green","blue")`

Simple Vector Operations

Length

```
> a <- c(1, 5, 2, 8)
> length(a)
```

```
[1] 4
```

Access Element

```
> a[3]
```

```
[1] 2
```

```
> a[c(1, 4)]
```

```
[1] 1 8
```

```
> a[a < 3]
```

```
[1] 1 2
```

Simple Vector Operations

Length

```
> a <- c(1, 5, 2, 8)
> length(a)
```

```
[1] 4
```

Access Element

```
> a[3]
```

```
[1] 2
```

```
> a[c(1, 4)]
```

```
[1] 1 8
```

```
> a[a < 3]
```

```
[1] 1 2
```

Simple Vector Operations

Generate Sequences

```
> a <- 1:10
> a
```

```
 [1]  1  2  3  4  5  6  7  8  9 10
```

```
> b <- seq(from = -pi, to = pi, length = 5)
> b
```

```
[1] -3.141593 -1.570796  0.000000  1.570796
[5]  3.141593
```

```
> c <- rep(0, 5)
> c
```

```
[1] 0 0 0 0 0
```

Function Definition

Functions

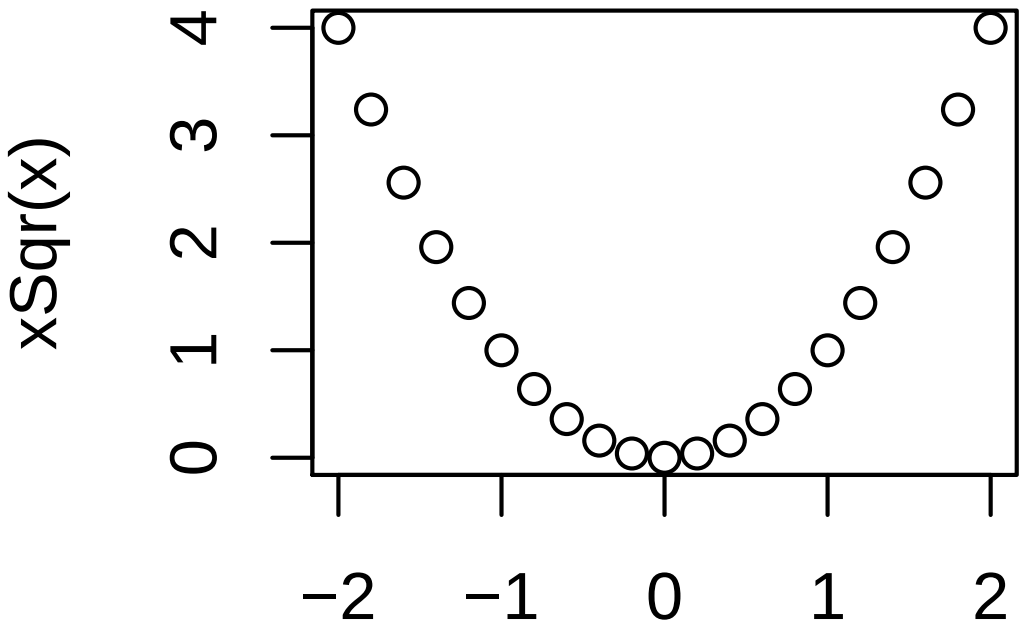
```
> xSqr <- function(x) {
+     y <- x^2
+     return(y)
+ }
> xSqr(2)
```

```
[1] 4
```

Plotting

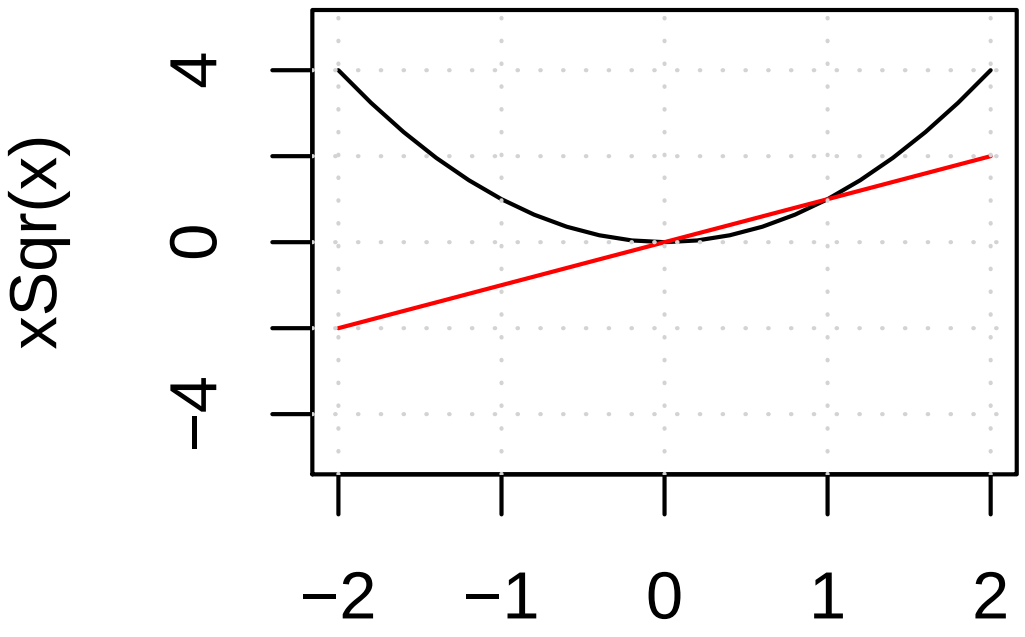
Plotting a Function

```
> x <- seq(from = -2, to = 2, length = 21)
> plot(x, xSqr(x))
```



Plotting a Function II

```
> x <- seq(from = -2, to = 2, length = 21)
> plot(x, xSqr(x), type = "l", ylim = c(-5,
+     5))
> lines(x, x, col = "red")
> grid()
```



Plot Sine and Cosine

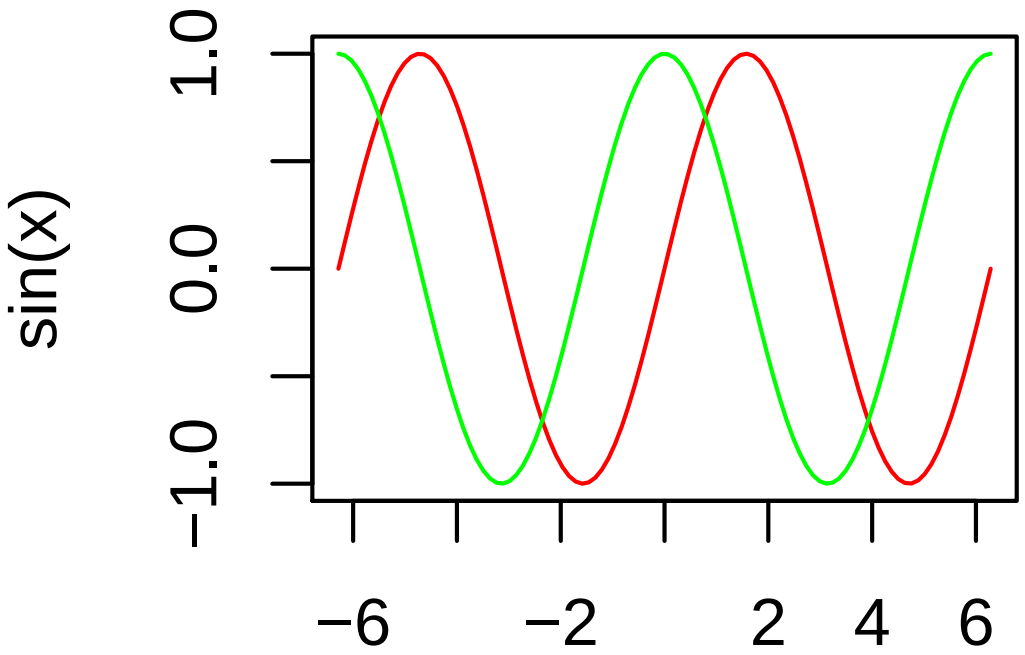
Task I

- generate a vector with values between -2π and 2π
- plot a sine (`sin()`) in red
- and a cosine (`cos()`) in green

Plot Sine and Cosine

Solution

```
> x <- seq(-2 * pi, 2 * pi, length = 100)
> plot(x, sin(x), "l", col = "red")
> lines(x, cos(x), col = "green")
```



Matrices

Define Matrix

```
> mat <- matrix(1:9, 3, 3)
> mat
```

```
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

Access Matrix Element

```
> mat[2, 3]
```

```
[1] 8
```

Matrices

Define Matrix

```
> mat <- matrix(1:9, 3, 3)
> mat
```

```
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

Access Matrix Element

```
> mat[2, 3]
```

```
[1] 8
```

Matrices

Access Rows

```
> mat[2, ]
```

```
[1] 2 5 8
```

Access Columns

```
> mat[, 3]
```

```
[1] 7 8 9
```

```
> mat
```

```
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

Matrices

Access Rows

```
> mat[2, ]
[1] 2 5 8
```

Access Columns

```
> mat[, 3]
[1] 7 8 9
```

```
> mat
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

Matrices

Add Rows

```
> mat2 <- rbind(mat, c(1, 1, 1))
> mat2
```

```
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
[4,]    1    1    1
```

Add Columns

```
> mat3 <- cbind(mat, c(2, 2, 2))
> mat3
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    4    7    2
[2,]    2    5    8    2
[3,]    3    6    9    2
```

Matrices

Add Rows

```
> mat2 <- rbind(mat, c(1, 1, 1))
> mat2
```

```
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
[4,]    1    1    1
```

Add Columns

```
> mat3 <- cbind(mat, c(2, 2, 2))
> mat3
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    4    7    2
[2,]    2    5    8    2
[3,]    3    6    9    2
```

Matrices

Add Rows

```
> mat2 <- rbind(mat, c(1, 1, 1))
> mat2
```

```
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
[4,]    1    1    1
```

Add Columns

```
> mat3 <- cbind(mat, c(2, 2, 2))
> mat3
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    4    7    2
[2,]    2    5    8    2
[3,]    3    6    9    2
```

Simple Matrix Operations

Dimension and Transpose

```
> mat <- matrix(1:12, 3, 4)
> dim(mat)
```

```
[1] 3 4
```

```
> t(mat)
```

```
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
[4,]   10   11   12
```

Simple Matrix Operations

Multiplication

```
> mat1 <- matrix(1:4, 2, 2)
> mat2 <- diag(1, 2)
> mat1 * mat2
```

```
     [,1] [,2]
[1,]    1    0
[2,]    0    4
```

```
> mat1 %*% mat2
```

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

Data Input/Output

Write Matrices

```
> a <- matrix(c(1:5, rnorm(5)), 5, 2)
> write(t(a), ncol = 2, file = "test.dat")
```

test.dat:

```
1 1.856992
2 -0.7361518
3 -1.874884
4 -0.0971819
5 -0.4247956
```

Alternatives

- `write.table()`
- `write.csv()`

Read/Write R-Data Files

Save Data

```
> a <- rnorm(100)
> b <- matrix(c(1:10, runif(10)), 10,
+     2)
> save(a, b, file = "Data.Rdat")
```

Load Data

```
> rm(list = ls())
> ls()

character(0)

> load("Data.Rdat")
> ls()

[1] "a" "b"
```

Read/Write R-Data Files

Save Data

```
> a <- rnorm(100)
> b <- matrix(c(1:10, runif(10)), 10,
+     2)
> save(a, b, file = "Data.Rdat")
```

Load Data

```
> rm(list = ls())
> ls()
```

```
character(0)
```

```
> load("Data.Rdat")
> ls()
```

```
[1] "a" "b"
```

Plot Sine plus Noise

Task II

- generate a time vector `t` with values from 0 to 100
- generate a sine wave for this time vector with frequency 1/20
- add Gaussian white noise with variance 0.1 to it (`rnorm()`)
- plot original and noisy curve
- write `t` plus the two time series into one ASCII file

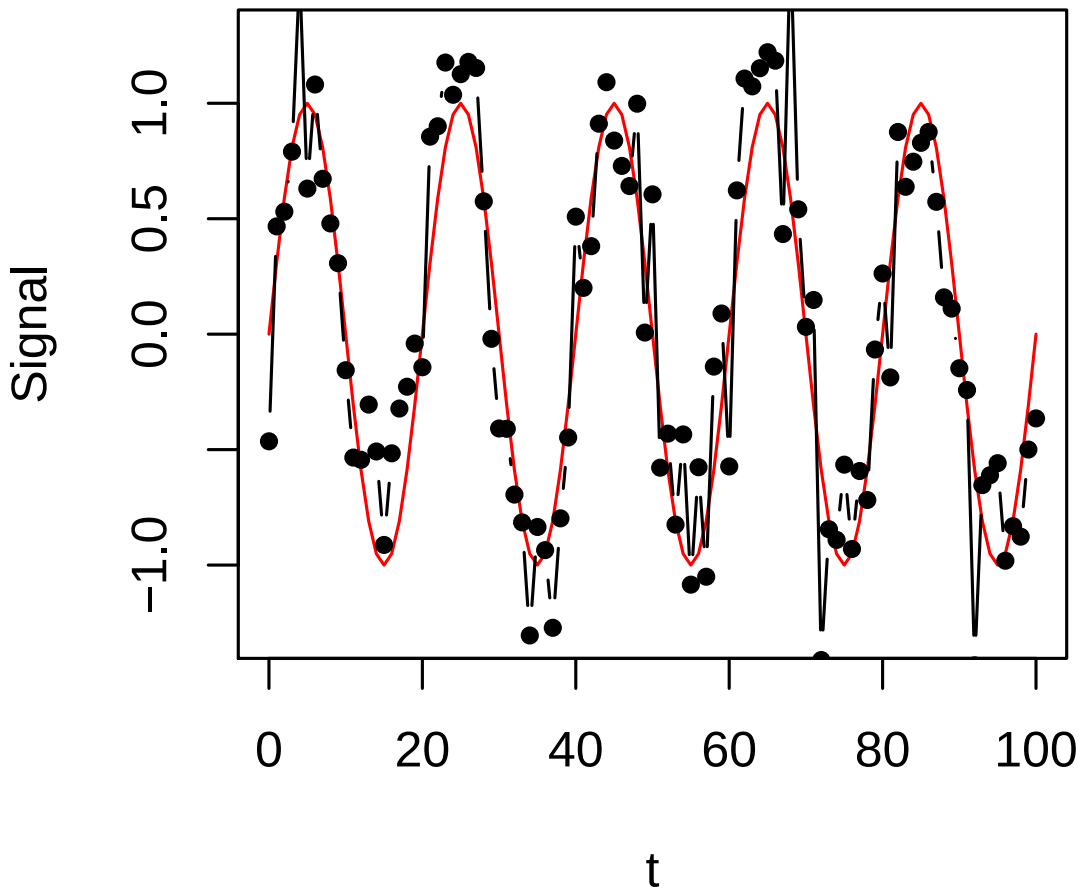
Plot Sine plus Noise

Solution

```
> t <- 0:100
> original <- sin(2 * pi * t/20)
> noisy <- original + rnorm(original,
+     sd = sqrt(0.1))
> plot(t, original, "l", col = "red",
+     ylab = "Signal", ylim = c(-1.3,
+         1.3))
> points(t, noisy, "b", pch = 20)
> write(t(matrix(c(t, original, noisy),
+     ncol = 3)), file = "sine.dat",
+     ncol = 3)
```

Plot Sine plus Noise

Solution



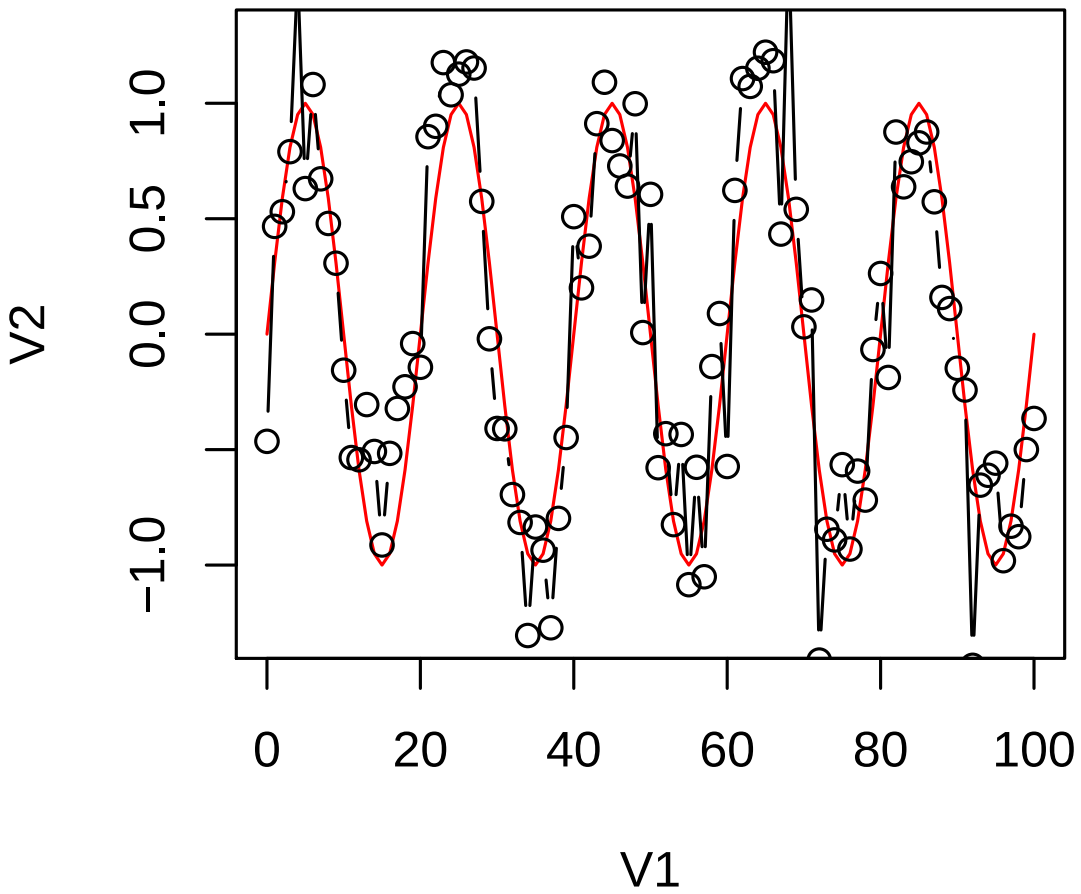
Read ASCII Data

`read.table()`

```
> dat <- read.table("sine.dat")
> plot(dat[, 1:2], type = "l", col = "red",
+      ylim = c(-1.3, 1.3))
> lines(dat[, c(1, 3)], type = "b")
```

Read ASCII Data

`read.table()`



Read ASCII Data

Alternatives

- `read.csv()` for CSV data, e.g. from ooCalc or Excel
- `read.fortran()` for Fortran-style format
- `read.delim()`
- `scan()` underlying generic function
- cf. also `http://cran.r-project.org/doc/manuals/R-data.html`

Simple Data Transformation

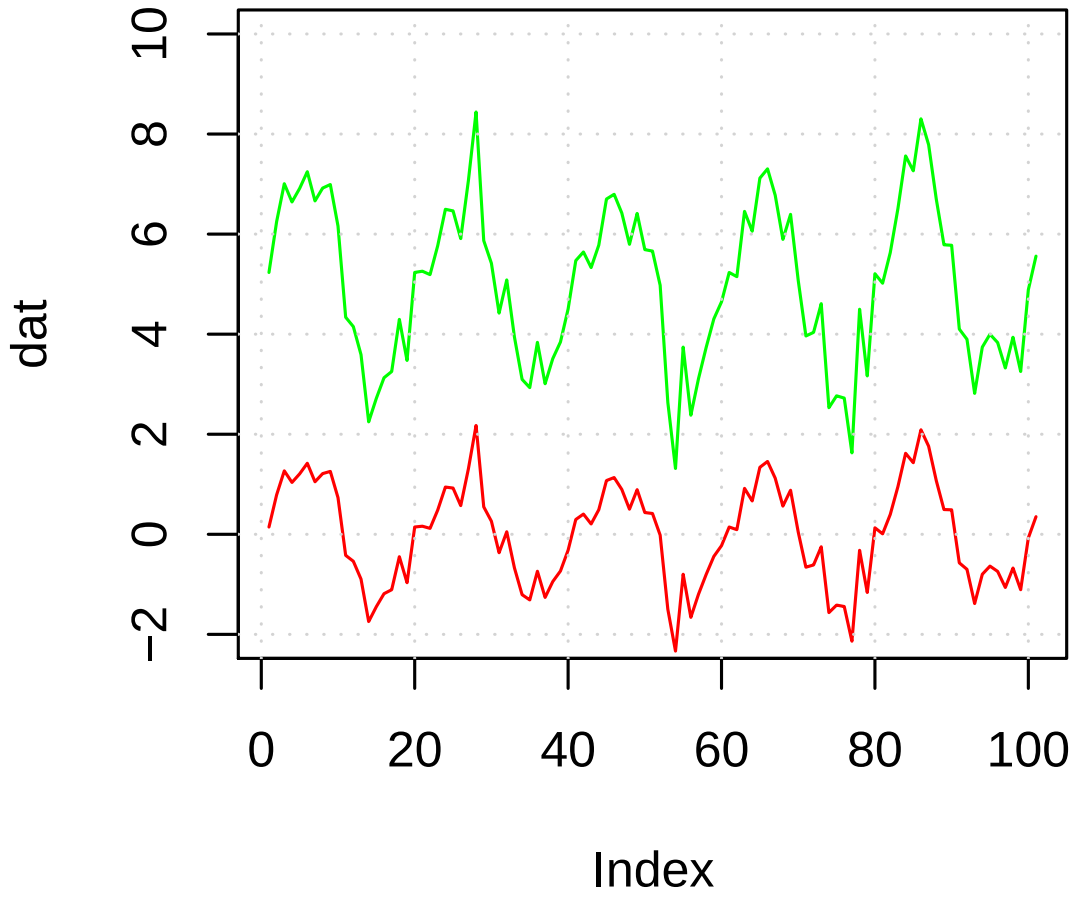
Data Transformations

Center and Rescale

```
> dat <- read.table("sine2.dat")[, 3]
> dat.center <- (dat - mean(dat))/sd(dat)
> plot(dat, type = "l", col = "green",
+     ylim = c(-2, 10))
> lines(dat.center, col = "red")
> grid()
```

Data Transformations

Center and Rescale



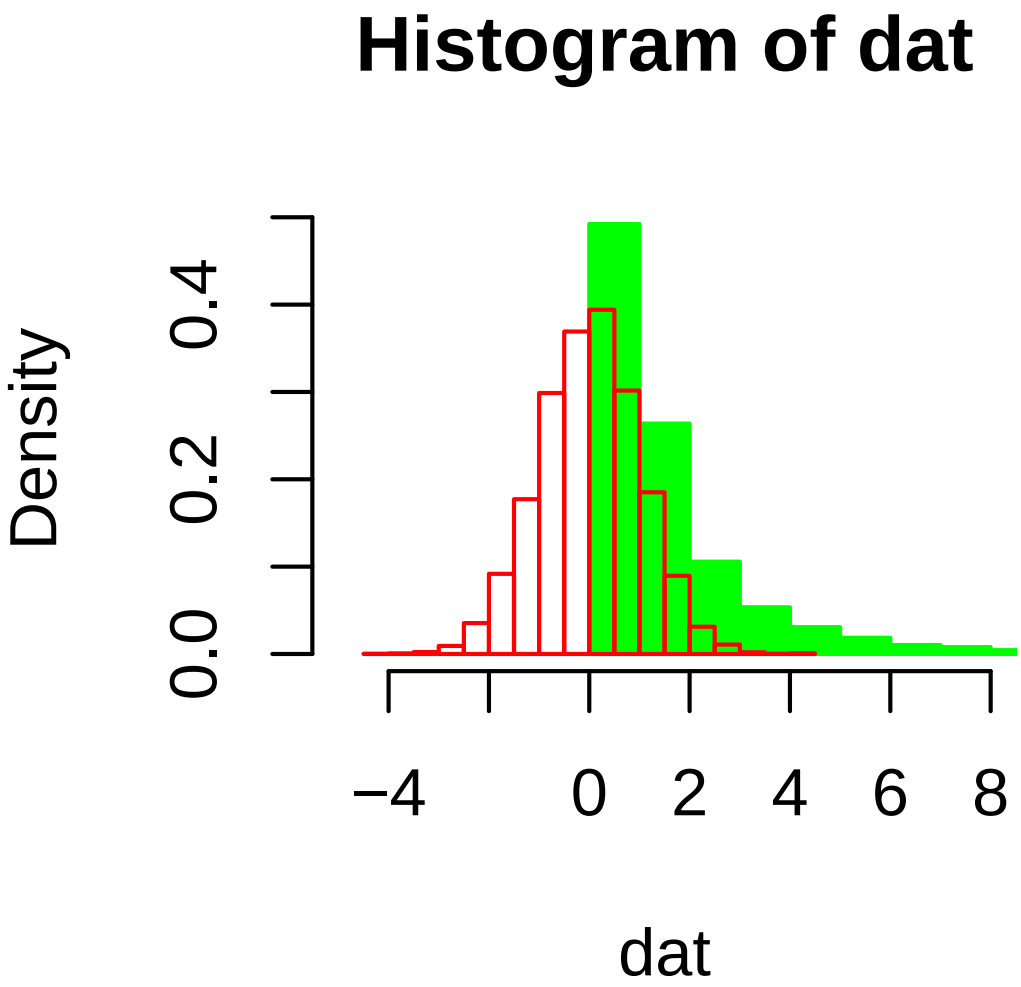
Data Transformations

Logarithmic Transformation

```
> dat <- rlnorm(10000)
> dat.log <- log(dat)
> hist(dat, xlim = c(-5, 8), border = "green",
+     prob = TRUE, breaks = 100, col = "green")
> hist(dat.log, add = TRUE, border = "red",
+     prob = TRUE)
```

Data Transformations

Logarithmic Transformation



Log-Transform Discharge

Task

- read ASCII data file `discharge.dat` into matrix `dat` with `read.table()`
- plot data
- plot histogram
- log transform
- compare histograms
- center and rescale
- use `dat.mat <- cbind(dat,dat.bc,dat.centered)` to combine dats into one matrix
- try summary(dat.mat)
- write `dat.mat` to a file using `write.table`

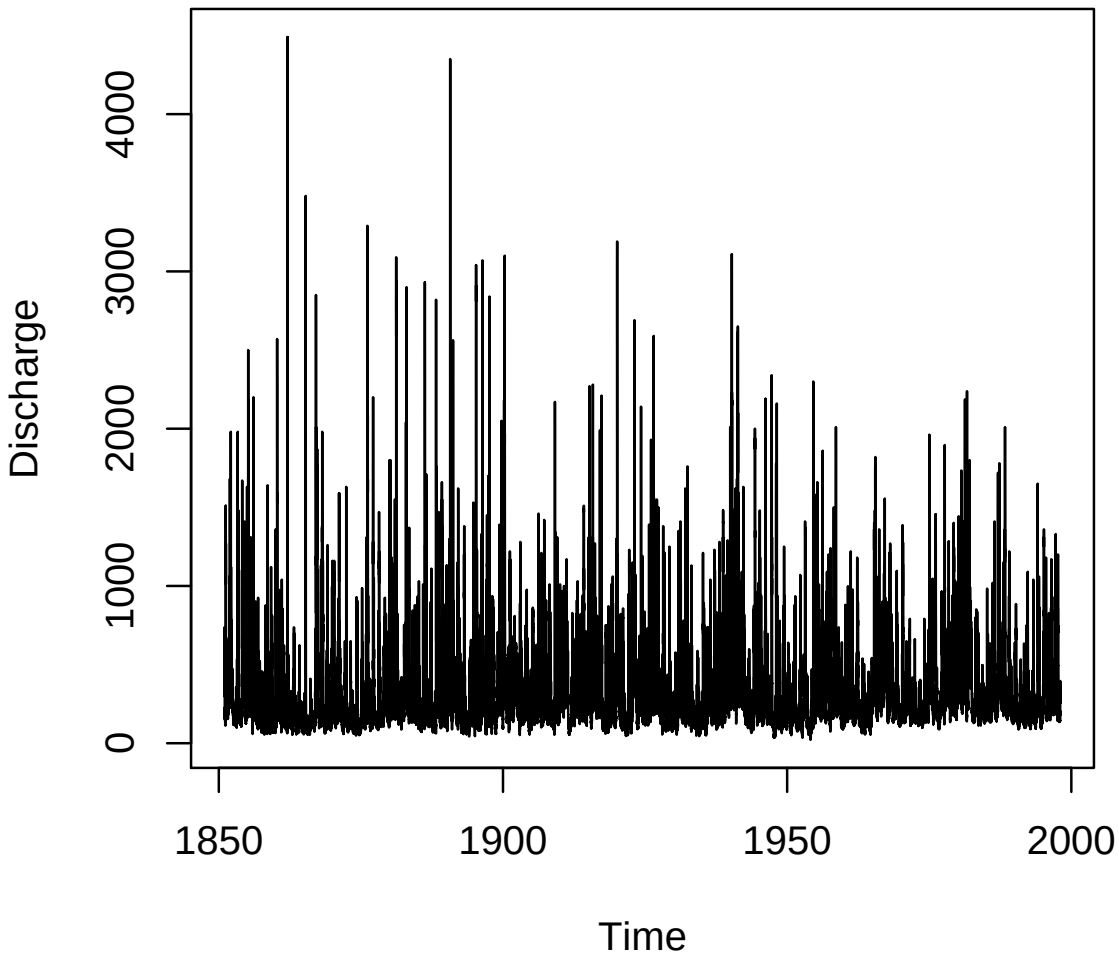
Log-Transform Discharge

Solution, Part I

```
> dat <- read.table("discharge.dat",
+     col.names = c("day", "month",
+         "year", "Q"))
> plot(dat$Q, type = "l", ylab = "Discharge")
> plot(ts(dat$Q, start = c(1851, 1),
+     frequency = 365), ylab = "Discharge")
> hist(dat$Q, xlab = "Discharge", prob = TRUE,
+     col = "green", border = "green",
+     xlim = c(-1000, 2000))
```

Log-Transform Discharge

Solution, Part I



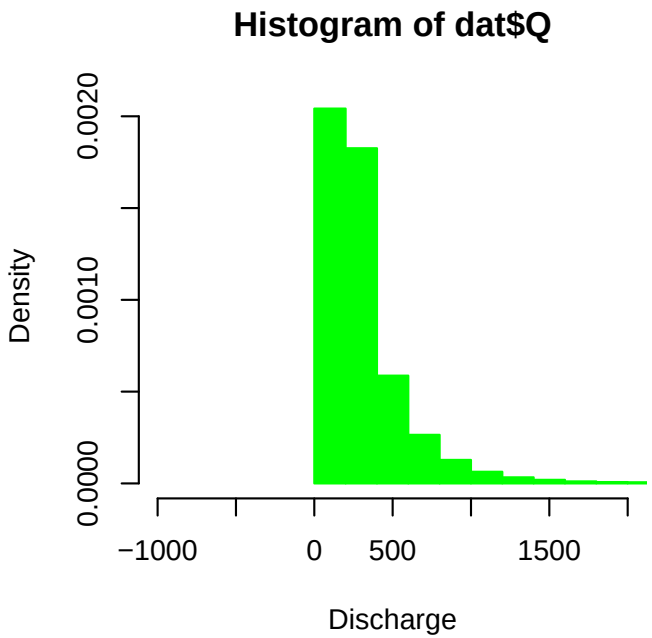
Log-Transform Discharge

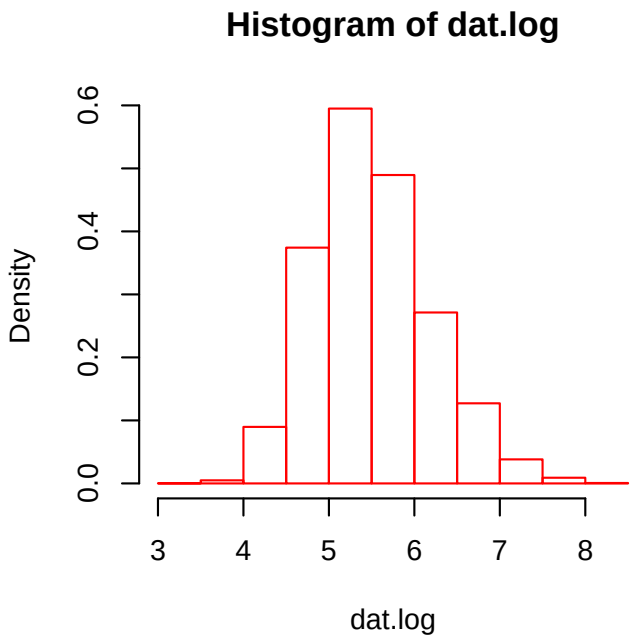
Solution, Part II

```
> dat.log <- log(dat$Q)
> hist(dat.log, prob = TRUE, border = "red")
> dat.centered <- (dat.log - mean(dat.log))/sd(dat.log)
> plot(ts(dat.centered, start = c(1851,
+     1), frequency = 365), ylab = "Discharge")
> dat.table <- cbind(dat, dat.log, dat.centered)
> write.table(dat.table, file = "discharge_log_centered.dat",
+     row.names = FALSE)
```

Log-Transform Discharge

Solution, Part II





Postscript, PDF, etc.

List of Graphics Devices

- `postscript()` PostScript
- `pdf()` Portable Document Format
- `pictex()` LaTeX/ PicTeX graphics commands
- `xfig()` XFIG format
- `X11()` X11 window system
- `windows()` windows window system
- `png()` portable network graphics (bitmap)
- `jpeg()` JPEG bitmat (bitmap)

syntax is similar, options are different, cf.

```
> ?postscript
```

PostScript

eps for Inclusion in LaTeX

```
> postscript("Plot.eps", paper = "special",
+     width = 6, height = 4, family = "Palatino",
+     horizontal = FALSE)
> plot(rnorm(100), rnorm(100), col = 2,
+     xlab = "x", ylab = "y", main = "Postscript")
> dev.off()
```

PDF, e.g. for Inclusion in pdfLaTeX

```
> pdf("Plot.pdf", width = 4, height = 3,
+     family = "Palatino")
> plot(rnorm(100), rnorm(100), col = 2,
+     xlab = "x", ylab = "y", main = "PDF")
> dev.off()
```

PostScript

eps for Inclusion in LaTeX

```
> postscript("Plot.eps", paper = "special",
+     width = 6, height = 4, family = "Palatino",
+     horizontal = FALSE)
> plot(rnorm(100), rnorm(100), col = 2,
+     xlab = "x", ylab = "y", main = "Postscript")
> dev.off()
```

PDF, e.g. for Inclusion in pdfLaTeX

```
> pdf("Plot.pdf", width = 4, height = 3,
+     family = "Palatino")
> plot(rnorm(100), rnorm(100), col = 2,
+     xlab = "x", ylab = "y", main = "PDF")
> dev.off()
```

Quit

```
> q()
```

Introduction to R

Round II

- Data Frames and Lists
- Logical Operators
- Flow-Control (If-then-else)
- Loops
- Classical Functions
- Some Statistics
  - Moments
  - Some Distributions
  - Tests

Appendix

In Relation to Others

Spreadsheet

- Gnumerics
- OpenOffice.Org Calc
- Excel
- Quattro Pro
- ...

Symbolic algebra

- Mathematica
- Maple
- OpenOffice.Org Math
- GAUSS
- ...

Numeric

- S (S-Plus / R)
- MatLab / Octave
- SASS / SPSS
- PV-Wave
- ...

In Relation to Others

Spreadsheet

- Gnumerics
- OpenOffice.Org Calc
- Excel
- Quattro Pro
- ...

Symbolic algebra

- Mathematica
- Maple
- OpenOffice.Org Math
- GAUSS
- ...

Numeric

- S (S-Plus / R)
- MatLab / Octave
- SASS / SPSS
- PV-Wave
- ...

In Relation to Others

Spreadsheet

- Gnumerics
- OpenOffice.Org Calc
- Excel
- Quattro Pro
- ...

Symbolic algebra

- Mathematica
- Maple
- OpenOffice.Org Math
- GAUSS
- ...

Numeric

- S (S-Plus / R)
- MatLab / Octave
- SASS / SPSS
- PV-Wave
- ...

What makes it different

from other specialised numerical software?

- fully planned and coherent system
- NOT an incremental accretion of very specific and inflexible tools
- designed around a computer language (interpreter)
- add additional functionality by defining new functions
- much of the system is written in R (transparent)
- C, C++ and Fortran code can be linked

R and Matlab

- **Need:** transfer data and code from R to Matlab and (maybe) vice versa.
- **Problem:** the languages differ so much that just translating code is not sufficient nor wanted

Some attemps to combine R and Octave/Matlab:

- `R.Matlab` lets R communicate with MatlabServer (needs Matlab) and reads and writes `.mat` files (without Matlab)
- `ROctave` Octave (a free Matlab-clone) functions in R and vice versa
- **Matlab to R code conversion** a shell script to help with that, no general approach so far. Ask me for `matlab2R`

Simple Transformations

Box-Cox Transformation

$$y = \begin{cases} (x^p - 1)/p, & p \neq 0 \\ \log(x), & p = 0 \end{cases}$$

```
> library(car)
> dat <- rexp(10000)
> box.cox.powers(dat)

Box-Cox Transformation to Normality

 Est.Power Std.Err. Wald(Power=0)
    0.2714    0.007       38.9773
 Wald(Power=1)
     -104.6444


L.R. test, power = 0:  1791.9184   df = 1   p = 0
L.R. test, power = 1:  8112.1541   df = 1   p = 0

> dat.bc <- box.cox(dat, 0.2665)
```

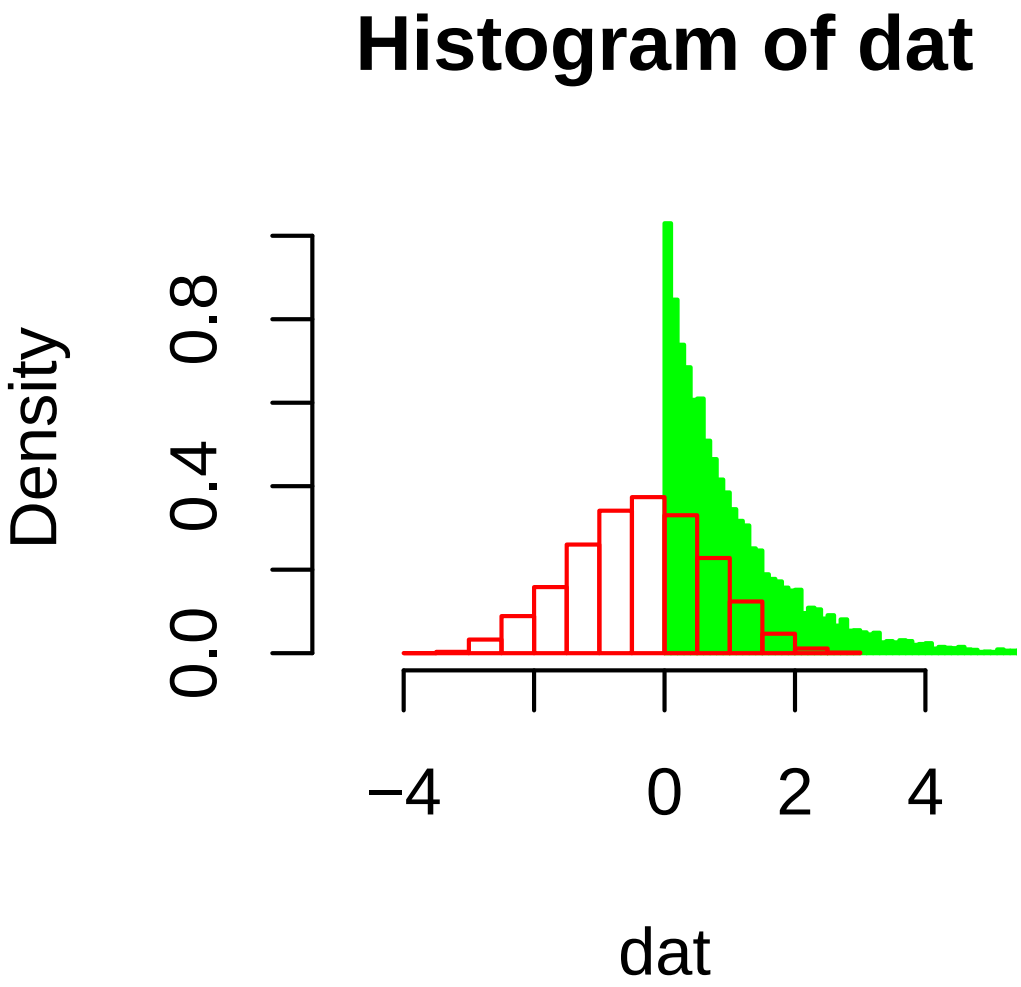
Simple Transformations

Box-Cox Transformation

```
> hist(dat, xlim = c(-5, 5), border = "green",
+     prob = TRUE, breaks = 100, col = "green")
> hist(dat.bc, add = TRUE, border = "red",
+     prob = TRUE)
```

Simple Transformations

Box-Cox Transformation



Read ASCII Data

scan()

```
> dat <- scan("sine.dat", list(t = 0,
+     original = 0, noisy = 0))
> plot(dat$t, dat$original, "l", col = "red",
+     ylim = c(-1.3, 1.3))
> lines(dat$t, dat$noisy, "b")
```

Read ASCII Data

scan()