

# WERKZEUGE DER MUSTERERKENNUNG UND DES MASCHINELLEN LERNENS

Vorlesung im Sommersemester 2017

Prof. E.G. Schukat-Talamazzini

Stand: 6. März 2017

## Teil I

### Einführung in die Sprache

#### Softwarewerkzeuge und ihre Benutzerschnittstelle

##### Graphikorientiert (GUI)

Menüauswahl  
Formulareintrag  
Drag & Drop

##### Befehlsorientiert (CLI)

Fixiertes Kommandoensemble  
Aufrufparameter und -syntax  
Skriptunterstützung

##### Sprachorientiert (PLI)

Formale Anwendungssprache  
Interpreterprogramm  
BS/PS-Interface

##### Programmorientiert (API)

Unterprogrammbibliothek  
Klassenbibliothek  
(wie PLI) erweiterbar

#### Inhaltlicher Vorlesungsaufbau

##### Die Programmiersprache 'R'

- **Sprache 'R' im Überblick**  
Hilfe, Installation, Sitzung
- **Rechnen und Datentypen**  
Felder, Listen, Funktionen, Operatoren
- **Text und Daten**  
Zeichenketten, Eingabe/Ausgabe, Graphikausgabe
- **Statistische Modelle**  
Regression, Optimierung, Verteilungen

##### Anwendungsgebiete von 'R'

- **Zeitreihen**  
Trend/Saison, Faltung, ACF, Spektrum, AR/MA, GARCH
- **Transformationen**  
Hauptachsen, MDS, Faktoren, ICA, NMF, LDA
- **Klassifikatoren**  
k-NN, MLP, SVM, SCT, OLS/GLS, LDA/QDA
- **Clusteranalyse**  
divisiv/agglomerativ, K-Means, Fuzzy, EM-Entmischung, spektral

## Was kann 'R'?

Sprachumfang

Grafikausgabe

Installation des 'R'-Pakets

Ablauf einer 'R'-Sitzung

Elementare Grafikbefehle

- ein leistungsfähiger Taschenrechner
- ein vektor/tensorbasierter Mathematikprozessor
- eine funktional-objektorientierte Programmiersprache
- ein Werkzeug zur statistischen Datenanalyse
- ein mächtiges Kommandosystem für wiss. Grafiken
- ein Interpreter für „die Sprache C  $\cup$  Lisp“
- ein offenes und kostenloses Softwareprodukt (S-Plus nicht!)
- in den Bereichen Statistik, Datamining, Bioinf weit verbreitet
- ein Algorithmen-Wiki für die internat. Datamining-Szene



## Wissenswertes über

- Dokumente des 'R'-Pakets: <file:///usr/lib/R/doc/html>

Tutorial, Language Definition, Installation, Erweiterungspakete, Datenim-/export, FAQ, ...

- Webseiten des 'R'-Projekts:

international: <http://www.r-project.org/> · national: <http://www.r-project.de/>

- Ausgewählte Handbücher und Kursmaterialien

Contributed Documentation ( $\geq 50$  Tutorials): <http://cran.r-project.org/other-docs.html>

- Interaktive Hilfe zur Programmlaufzeit

(*Funktionsaufruf*)

1. Funktionsbeschreibung

2. Beispieldemonstration

3. Funktionsdeklaration

4. Suchfunktion

`funcname (a,b,...)`

`help (funcname)`

bzw. `?funcname`

`example (funcname)`

`funcname`

`help.search (topic)`

bzw. `??topic`

## Was kann 'R'?

Sprachumfang

Grafikausgabe

Installation des 'R'-Pakets

Ablauf einer 'R'-Sitzung

Elementare Grafikbefehle

## 'R' als Taschenrechner

- 'R' ist interaktiv: der Programmtext wird interpretiert
- 'R'-Programm  $\hat{=}$  Folge von Ausdrücken
- der Ausdruck wird ausgewertet; der Wert wird angezeigt:  
`(17+4)*(35-25)+4501 ... [1] 4711`
- nach Zuweisungsausdrücken wird die Wertausgabe unterdrückt:  
`xvar <- (17+4)*(35-25)+4501 ... (nix)`
- der Wert wurde aber selbstverständlich zugewiesen:  
`xvar + 1111 ... [1] 5822`
- Variable werden beim Erstauftauch automatisch deklariert  
`objects() ... [1] 'swap' 'writePNM' 'xvar'`

## Listen, Ausdrücke & Parsebäume

- Listen enthalten Komponenten von unterschiedlichem Typ:  
`lis <- list (zahl=12, TRUE, 'zwei Wörter')`
- Selektion per Index oder per Name:  
`lis[[2]] ... [1] TRUE oder lis$zahl ... [1] 12`
- Programmtexte können geparsert werden:  
`ex <- parse (text='a-b; (17+4)*45; x+y')`
- Es wird eine Liste von Parsebäumen erzeugt:  
`ex2 <- ex[[2]] ; ex2 ... (17 + 4) * 45`
- Jeder Ausdruck liegt als Kantorovic-Ableitungsbaum vor:  
`ex2[1] ex2[2] ex2[3] ex2[4]  
'*'() (17 + 4)() 45() NULL()`
- Kantorovic-Bäume können (u.U.) ausgewertet werden:  
`eval(ex2) ... [1] 945`  
`eval(ex) ... Error in eval(expr, envir, enclos) : Object 'a' not found`

## Vektoren, Matrizen, Tensoren

- Die elementaren Datentypen von 'R' sind Vektoren:  
`x <- c(1,4,9,16) ; x ... [1] 1 4 9 16`
- Es gibt zahlreiche Konstruktoren für Vektorobjekte:  
`y <- 1:4 ; y ... [1] 1 2 3 4`
- Arithmet./logische Verknüpfungen operieren komponentenweise:  
`x+y ... [1] 2 6 12 20`
- Matrizen bestehen aus Datenvektor & Strukturinformation:  
`matrix (data=1:12, nrow=3, ncol=4, byrow=FALSE)`

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

- Es gibt zahlreiche Werteselektionsmöglichkeiten:  
`mat[3] mat[2,3] mat[2,1:3] mat[2:3,2]  
[1] 3 [1] 8 [1] 2 5 8 [1] 5 6`

## Datensätze laden — auch über das Internet

- Als Dateinamen werden auch URLs akzeptiert:  
`myurl <- 'http://stat.ethz.ch/Teaching/Datasets/NDK/sport.dat'  
d.sport <- read.table (myurl)`
- Datensätze mit benannten Mustern & Merkmalen:

	weit	kugel	hoch	disc	stab	speer	punkte
O'BRIEN	7.57	15.66	207	48.78	500	66.90	8824
BUSEMANN	8.07	13.60	204	45.04	480	66.86	8706
DVORAK	7.60	15.82	198	46.28	470	70.16	8664
FRITZ	7.77	15.31	204	49.84	510	65.70	8644
HAMALAINEN	7.48	16.32	198	49.62	500	57.66	8613
NOOL	7.88	14.01	201	42.98	540	65.48	8543
ZMELIK	7.64	13.53	195	43.44	540	67.20	8422
GANIYEV	7.61	14.71	213	44.86	520	53.70	8318
PENALVER	7.27	16.91	207	48.92	470	57.08	8307
HUFFINS	7.49	15.57	204	48.72	470	60.62	8300
PLAZIAT	7.82	14.85	204	45.34	490	52.18	8282
MAGNUSSON	7.28	15.52	195	43.78	480	61.10	8274
SMITH	7.47	16.97	195	49.54	500	64.34	8271
MUELLER	7.25	14.69	195	45.90	510	66.10	8253
CHMARA	7.75	14.51	210	42.60	490	54.84	8249

- Auswahl von Zeilen und Spalten, z.B. `d.sport[, 'kugel']`  
`[1] 15.66 13.60 15.82 15.31 16.32 14.01 13.53 14.71 16.91 15.57 14.85 15.52`  
`[13] 16.97 14.69 14.51`

## Aufrufen & Deklarieren von Funktionen

- Funktionsdeklaration mit formalen Parametern:

```
funcname <- function (arglist) {body}
```

- Parameterbezeichner *vname*
- Parameter mit Voreinstellung *vname=default*
- Restparameterliste ...

- Funktionsaufruf mit aktuellen Parametern:

```
funcname(arglist)
```

- Positionelle Übergabe *expr*
- Namentliche Übergabe *vname=expr*
- Restparameterübergabe ...

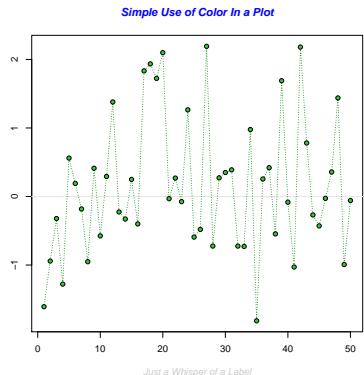
- Beispiel:

```
zeichne <- function (x, y=NULL, title='Grafik', ...) {
  if (is.null (y))
    { y <- x; x <- 1:length(x) }
  plot (y ~ x, main=title, ...)
}

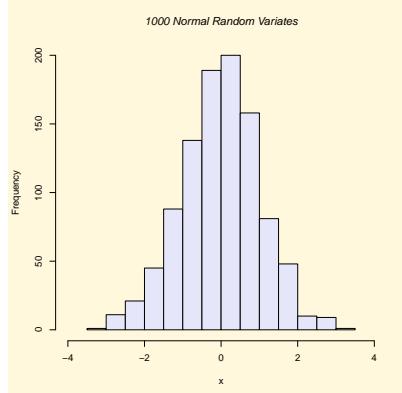
zeichne (sin(1:20), cos(1:20), title='Kreis', col='red')
```

## Wertesequenzen & Wertemengen

Skalare Datensammlung mit/ohne Reihenfolgeinformation



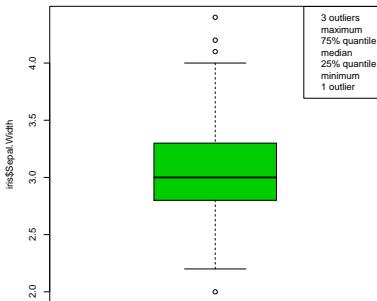
Stützwertediagramm



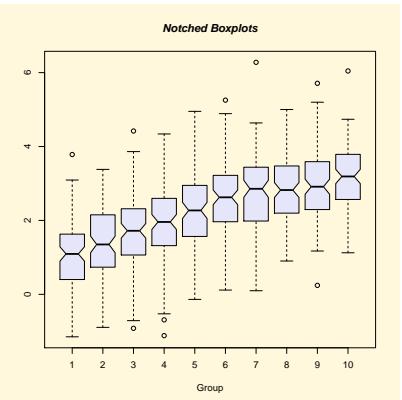
Balkendiagramm

## Wertemengen & Verteilungsform

Wahrscheinlichkeitsverteilungen auf stetigen Skalen



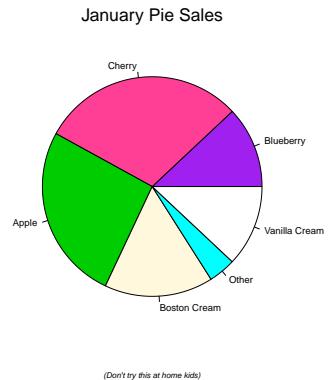
Box-Whisker-Plot



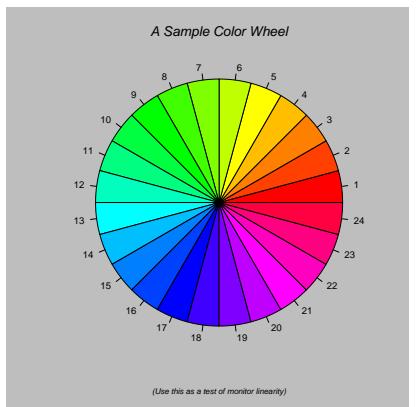
Mittel & Abweichung(en)

## Häufigkeiten & Proportionen

Wahrscheinlichkeitsverteilungen auf diskreten Skalen



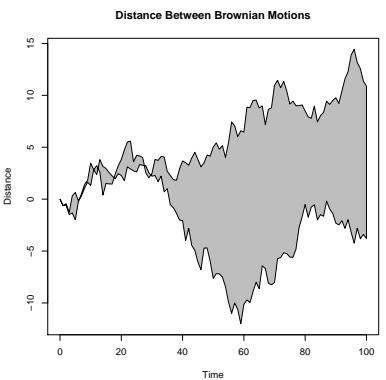
Tortengrafik



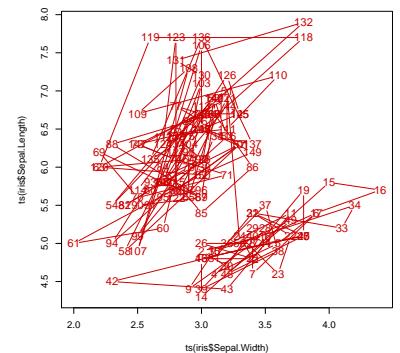
Wagenradgrafik

## Wertesequenzpaare

Konkurrierende Sequenzen · Komplementäre Sequenzen



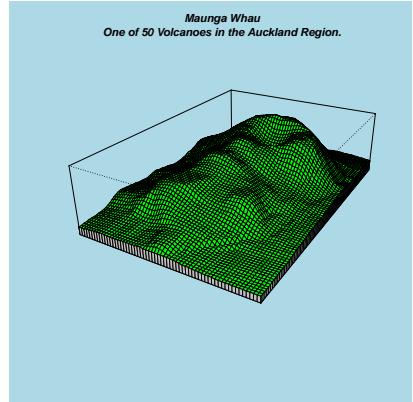
Minimum/Maximum



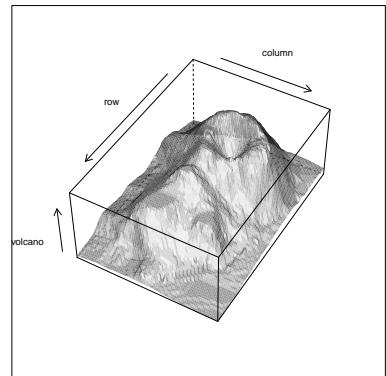
Zweikanal-Zeitreihen (2D-Trajektorie)

## Datenmatrizen, Einkanalbilder I

Gebirgsmetapher durch Gitterprojektion oder 3D-Rendering



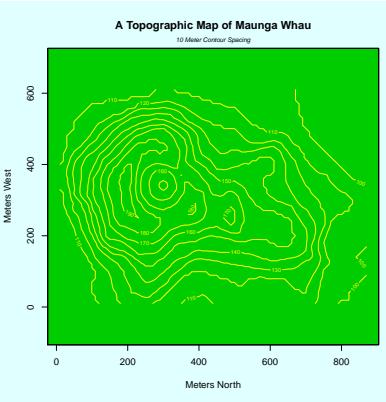
Drahtgitterlandschaft



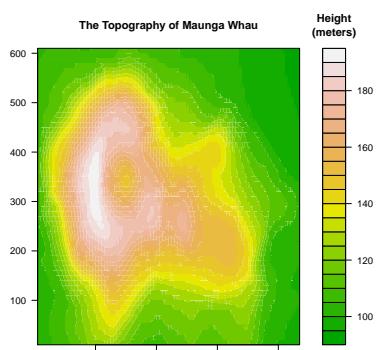
Schattierung

## Datenmatrizen, Einkanalbilder II

Werte durch Grenzen · Werte durch Farben



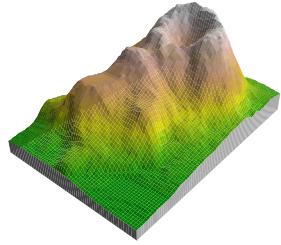
Konturen/Isolinien



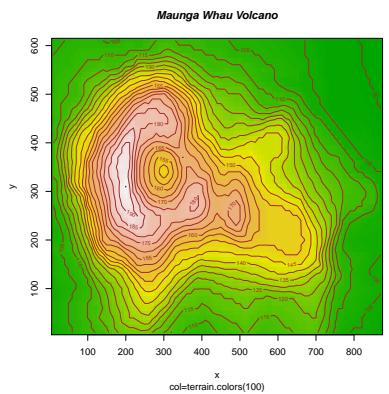
Falschfarbendarstellung

## Kombinierte Skalendarstellung

Zwei Modalitäten in einem Koordinatensystem



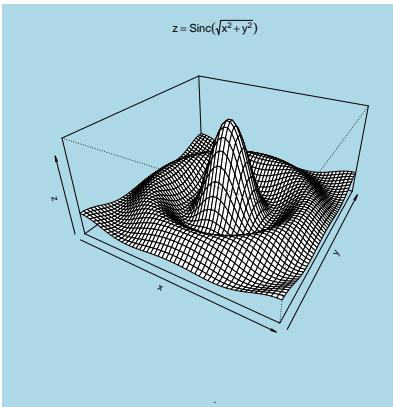
Schattierung & Falschfarben



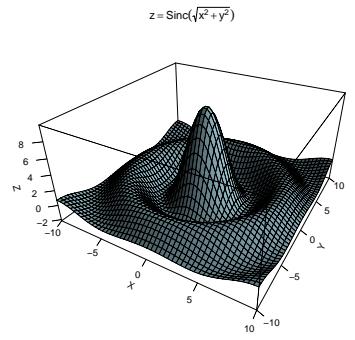
Isolinien & Falschfarben

## Funktionsverläufe $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ zweier Variablen

Zuordnungsvorschrift statt Datenmenge



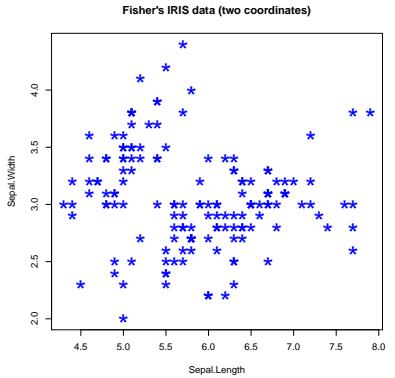
Drahtgitterlandschaft



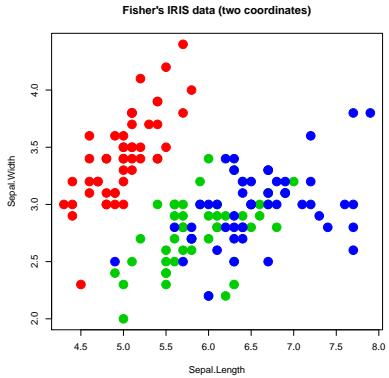
Gitter & Schattierung

## Zweikanalige Daten

Scatterplot ohne/mit Klassenkennzeichnung



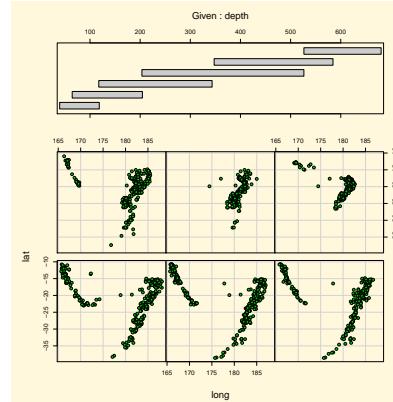
alle Datenpunkte  $(x_i, x_j)$  in der Ebene



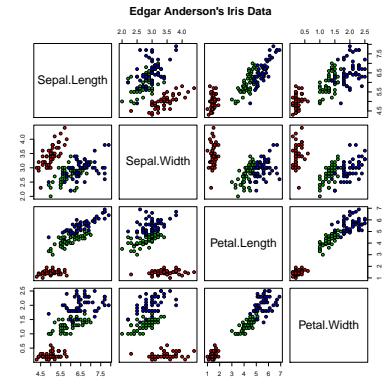
dto., mit Klassenmarkierung

## Drei- und mehrkanalige Daten

Schubladengrafik · Kombinatorischer Scatterplot



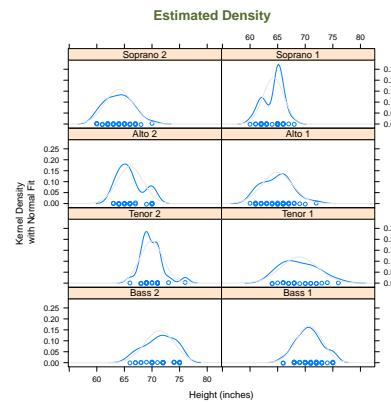
$x_3$ -partitionierte  $(x_1, x_2)$ -Diagramme



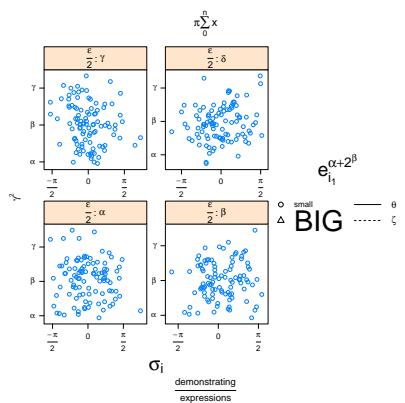
alle möglichen  $(x_i, x_j)$ -Diagramme

## Darstellung von Verteilungsmodellen

Empirische/geschätzte Dichte · Text- und Formelsatz



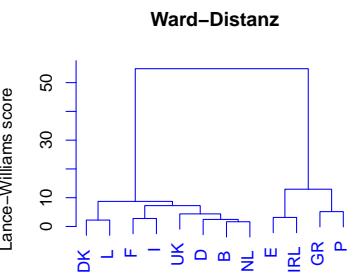
Gruppenbezogene Dichtevisualisierung



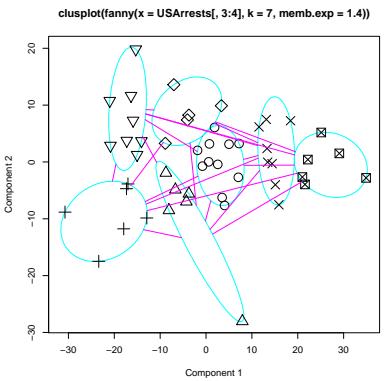
Formelsatz in LATEX-Syntax

## Clusteranalyse (Gruppierung)

Hierarchische Gruppierung · Unscharfe Gruppierung



Dendrogramm (AGNES)



Partition (fuzzy K-means)

Was kann 'R'?

Sprachumfang

Grafikausgabe

Installation des 'R'-Pakets

Ablauf einer 'R'-Sitzung

Elementare Grafikbefehle



am Institut für Informatik

'R' ist verfügbar an allen Rechnern im Linux-PC-Pool des FRZ

ppc212&gt; R

R version 2.11.1

```
R version 2.11.1 (2010-05-31)
Copyright (C) 2010 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
```

R ist freie Software und kommt OHNE JEGLICHE GARANTIE.  
 Sie sind eingeladen, es unter bestimmten Bedingungen weiter zu verbreiten.  
 Tippen Sie 'license()' or 'licence()' für Details dazu.

R ist ein Gemeinschaftsprojekt mit vielen Beitragenden.  
 Tippen Sie 'contributors()' für mehr Information und 'citation()',  
 um zu erfahren, wie R oder R packages in Publikationen zitiert werden können.

Tippen Sie 'demo()' für einige Demos, 'help()' für on-line Hilfe, oder  
 'help.start()' für eine HTML Browserschnittstelle zur Hilfe.  
 Tippen Sie 'q()', um R zu verlassen.



## zu Hause unter Linux installieren

1. Rufen Sie die Webseite <http://cran.r-project.org/>  
CRAN = Comprehensive 'R' Archive Network
2. Nutzen Sie die ONE-CLICK-Installation  
`R binaries ~> linux ~> suse ~> SuSE 12.1 R-patched-devel-2.14.1`
3. ... oder laden Sie sich eine Distribution herunter und installieren Sie das Paket als 'root':  
`rpm -i /home/schukat/hereis/R-base-2.4.0-1.i586.rpm`

```
error: Failed dependencies:
xorg-x11-fonts-100dpi is needed by R-base-2.4.0-1
blas is needed by R-base-2.4.0-1
libgfortran.so.0 is needed by R-base-2.4.0-1
```

4. Installieren Sie die fehlenden Pakete nach ([YAST2](#)), z.B.  
`blas · xorg-x11-fonts-100dpi · fortran o.ä.`



## zu Hause unter Windows installieren

Warum sollte das eigentlich **nicht** möglich sein? · Wer hat da eben gelacht?

- (Algorithmus)
- 1 Laden Sie sich eine Binärversion für Windows XP herunter.
  - 2 Überzeugen Sie den Installationsassistenten Ihres Vertrauens davon, daß es sich beim 'R'-Paket um ein Egoschutterprogramm handelt.
  - 3 Führen Sie einen Systemneustart durch.
  - 4 Entfernen Sie die verkohlten Kunststoffreste von der Auslegeware.

(zumDingA)

Was kann 'R'?

Sprachumfang

Grafikausgabe

Installation des 'R'-Pakets

Ablauf einer 'R'-Sitzung

Elementare Grafikbefehle

## Beginn und Ende einer 'R'-Sitzung

Konfiguration einer individuellen Umgebung

- Starten einer Sitzung:  
`.../working-directory> R`
- Beenden einer Sitzung:  
`quit()` oder `q()` oder `q('yes')` oder `q('no')`
- Der individuelle Zuschnitt Ihrer 'R'-Umgebung:  
`.../working-directory/.Rprofile`
- Prolog zu Sitzungsbeginn:  
... die Funktion `.First()` wird ausgeführt
- Epilog bei Sitzungsende:  
... die Funktion `.Last()` wird ausgeführt
- BEISPIEL:  
Die Startdatei `.Rprofile` enthält die Funktionsdefinitionen:  
`.First <- function() { source ('./lib/mydefs.R') ; cat (paste (date(), 'Hola')) }`  
`.Last <- function() { graphics.off() ; cat (paste (date(), 'Adios')) }`

## Kommandos zur Objektverwaltung

Listen · Löschen · Ein/Ausgabe von Daten und Kommandos

- Auflisten aller definierten Objekte:

`objects()` oder `ls()`

- Löschen definierter Objekte:

`remove(x,y,fun)` oder `rm(list=ls())` (?!?)

- 'R'-Kommandos von Datei lesen

`source(file)` (Eingabeumlenkung)

- 'R'-Ausgaben/Meldungen in Datei schreiben

`sink(file=NULL, append=FALSE)` (Ausgabeumlenkung)

- 'R'-Objekte in eine Datei speichern

`save(..., list=character(0), file=)`

- 'R'-Objekte aus einer Datei laden

`load(file)` oder `loadURL(url)` (char vector)

- Alle Sitzungsobjekte werden nach `.RData` geschrieben

## Stapelverarbeitung für -Programme

Aufruf von und Parameterübergabe an 'R'-Skriptdateien

- Abarbeitung einer Datei mit 'R'-Programmcode:

`Rscript progfile.R`

(entspricht Ausführung von `source("progfile.R")` im 'R'-Interpreter)

- Aufruf eines 'R'-Skripts mit Parameterübergabe:

`Rscript progfile.R string1 string2 string3 ... ... ...`

Nach der Zuweisung

`arg <- commandArgs(trailingOnly=TRUE)`

finden sich in den Komponenten `arg[1]`, `arg[2]` usw. des character-Vektors `arg` die Aufrufparameter als Zeichenketten.

- Etwaige Grafikausgaben finden wir im aktuellen Arbeitskatalog unter dem Namen `Rplots.pdf`.

(sofern Pdf voreingestelltes Ausgabeformat ist)

## Zeichnen von Funktionsverläufen

Was kann 'R'?

Sprachumfang

Grafikausgabe

Installation des 'R'-Pakets

Ablauf einer 'R'-Sitzung

Elementare Grafikbefehle

- Generische Funktion delegiert an zuständige Methode:

`plot(func,...) ~> plot.function(func,...)`

- Methode zum Kurvenzeichnen:

`curve(expr, from, to, n=101, add=FALSE, type='l', ylab=NULL, log=NULL, xlim=NULL, ...)`

- Der Funktionsname ist gegeben, z.B.:

`plot(sin)`

- Eine Funktionsdefinition ist gegeben, z.B.:

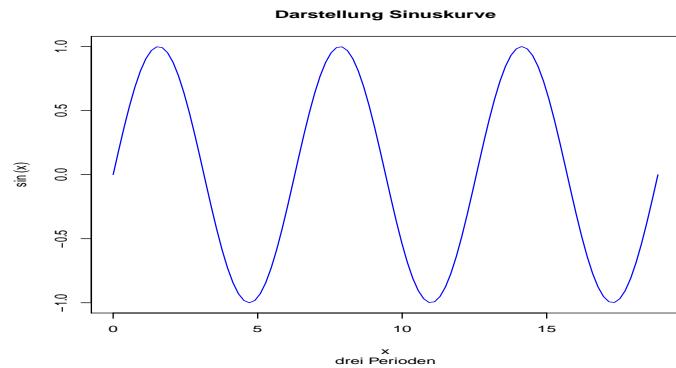
`plot(function(z) {2*z+5})`

- Das Intervall für die Argumentwerte ist spezifiziert, z.B.:

`plot(sin, -2, +4)`

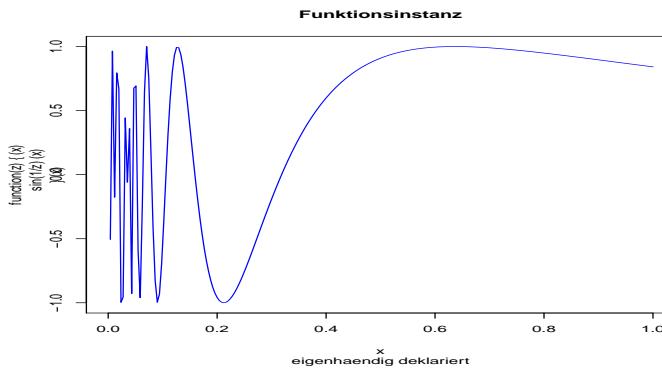
## Beispiel: eine Sinuskurve

```
plot ( sin, from=0, to=6*pi, col="blue",
       main="Darstellung Sinuskurve",
       sub="drei Perioden" )
```



## Beispiel: eine Funktionsdeklaration

```
plot ( function(z) {sin(1/z)}, col='blue', n=256,
       main='Funktionsinstanz',
       sub='eigenhaendig deklariert' )
```



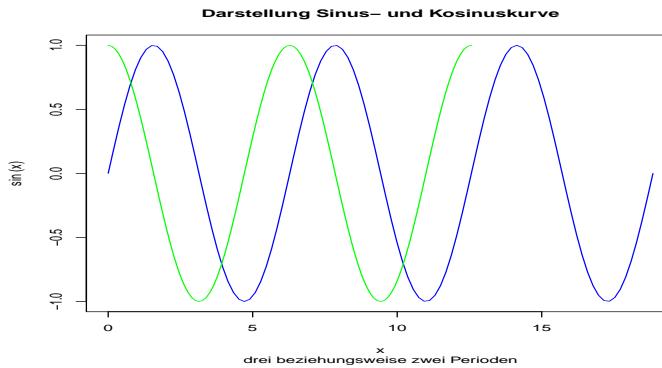
## Bildübergreifende Grafikbefehle

- Überlagern mehrerer Verläufe in einem Achsenkreuz  
`plot (x,y, add=TRUE)`
- Umleiten der Grafikausgabe von Schirm auf Datei  
`pdf (file='Rplot.pdf', ???)`  
 und entsprechend für `postscript pictex png jpeg GTK GNOME xfig bitmap`  
 zurück mit `X11 (display, ???)`
- Keine neue Zeichnung ohne Bestätigung  
`par (ask=TRUE)`
- Aufteilung der Leinwand in mehrere Grafikfelder  
`par (mfrow=c(n,m))` oder  
`par (mfcol=c(n,m))`

(Stapelbetrieb)

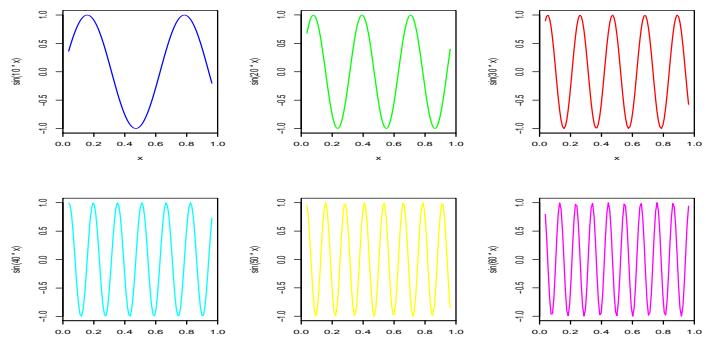
## Beispiel: zwei Kurven in einem Bild

```
plot ( sin, from=0, to=6*pi, col="blue",
       main="Darstellung Sinus- und Kosinuskurve",
       sub="drei beziehungsweise zwei Perioden")
plot ( cos, from=0, to=4*pi, add=TRUE, col="green")
```



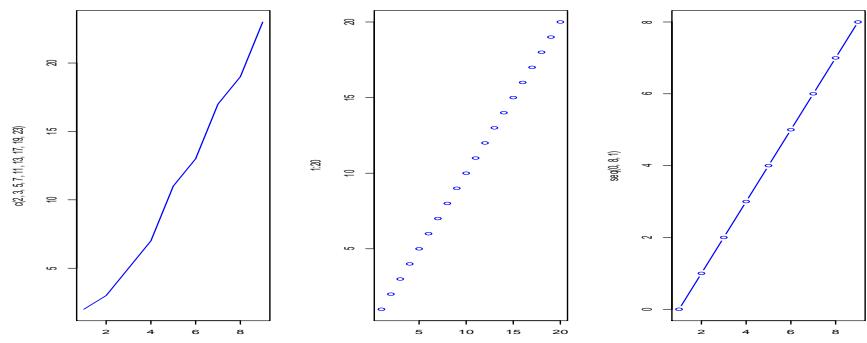
## Beispiel: sechs Bilder auf einem Blatt

```
par (mfrow = c(2,3))
curve (sin(10*x), col="blue"); curve (sin(20*x), col="green")
curve (sin(30*x), col="red"); curve (sin(40*x), col="cyan")
curve (sin(50*x), col="yellow"); curve (sin(60*x), col="magenta")
```



## Beispiel: eindimensionale Wertefolgen

```
par (mfrow = c(1,3))
plot (c(2,3,5,7,11,13,17,19,23), type='l', col='blue')
plot (1:20, type='p', col='blue')
plot (seq (0,8,1), type='b', col='blue')
```

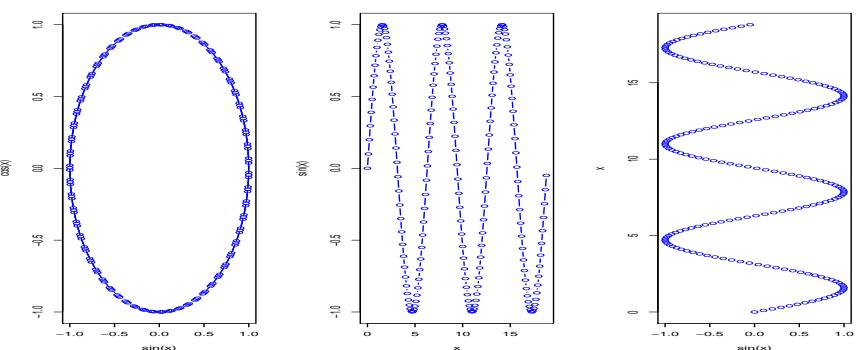


## Wertesequenzen und Punktesequenzen

- Wertesequenzen zeichnen: `plot(x)`, z.B.:  
`plot (c (2,3,5,7,11,13,17))`  
`plot (1:20)`  
`plot (seq (0,8,0.1))`  
`plot (sin (seq (0,8,0.1)))`
- Punktesequenzen zeichnen: `plot(x,y)`  
z.B. `x <- seq (0,2*pi,0.1)` nebst  
`plot (x, sin(x))` oder  
`plot (sin(x), x)` oder  
`plot (cos(x), sin(x))`
- Kurvendarstellung durch `type='?'` gesteuert:  
`points` `lines` `both`  
`overplotted` `contour-only`  
`high-density` `steps` `Steps` ...

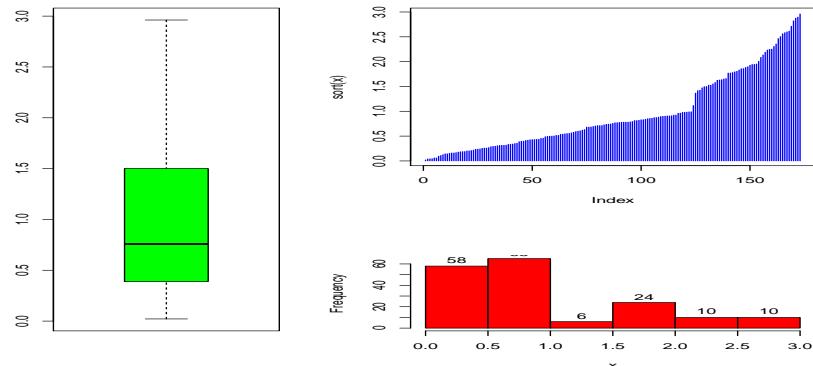
## Beispiel: zweidimensionale Wertefolgen

```
par (mfrow = c(1,3))
x <- seq (0, 6*pi, 0.1)
plot (sin(x), cos(x), type='b', col='blue')
plot (x, sin(x), type='b', col='blue')
plot (sin(x), x, type='b', col='blue')
```



## Leinwandaufteilung mit variablen Feldmaßen

```
layout (matrix (c(1,1,2,3), ncol=2), width=c(2,3), height=c(3,2))
x <- c (runif (123), rnorm (50, mean=2, sd=0.5))
boxplot (x, col="green")
plot (sort(x), col="blue", type="h")
hist (x, col="red", lab=TRUE, main="")
layout (1) # nicht vergessen!!
```



## Zusammenfassung (1)

1. 'R' ist eine **funktionale** Programmiersprache mit rudimentärer Objektorientierung (**polymorphe** Aufrufe).
2. 'R' arbeitet **interaktiv** (Interpreter statt Compiler), unterstützt aber Stapelverarbeitung (**Skripten**).
3. 'R' gilt als spektakulär für wissenschaftliches Rechnen, **Statistik**, **Datenmodellierung** und -**visualisierung**; für Bild- und Textverarbeitung gibt es performantere Wettbewerber.
4. Der elementare Datentyp von 'R' ist der **Vektor**.
5. In 'R' sind auch **Funktionen** und **Ausdrücke** Sprachobjekte, können also zur Laufzeit manipuliert werden.
6. 'R' unterstützt die E/A beliebiger 'R'-Objekte durch **automatische Serialisierung**.
7. **Grafiken** werden kumulativ erzeugt; zuerst das **Koordinatensystem**, dann sukzessive weitere **Zeichnungskomponenten**.
8. Komplexere Grafiken werden durch **Leinwandaufteilung** und sequentielle **Komposition** einfacher Grafiken erzeugt.