Einführung in R

Version 1.0 vom 31.12.2002

Dr. Matthias Fischer

Lehrstuhl für Statistik & Ökonometrie

Universität Erlangen-Nürnberg

Matthias.Fischer@wiso.uni-erlangen.de

Gliederung

- Einführung in **R**
- Zahlen und Vektoren
- Matrizen
- Ein- und Auslesen von Daten
- Erzeugung von Grafiken
- Lineares Modell
- Verteilungen
- Ergänzungen
- Zeitreihenanalyse

Gliederung

- Einführung in **R**
- Zahlen und Vektoren
- Matrizen
- Ein- und Auslesen von Daten
- Erzeugung von Grafiken
- Lineares Modell
- Verteilungen
- Ergänzungen
- Zeitreihenanalyse

Einführung: Was ist R?

- Tool zur umfangreichen Datenanalyse
- Tool zur grafischen Datenanalyse
- Tool zur Matrixalgebra
- einfache, effektive Programmiersprache
- Alternative zu den Programmen MATLAB, OX, GAUSS, SHAZAM, LIMDEP, EVIEWS, TSP, S-PLUS etc.
- Freeware-Version von S-PLUS
- Ergänzung zu Excel oder SPSS

Einführung: **R-Code**

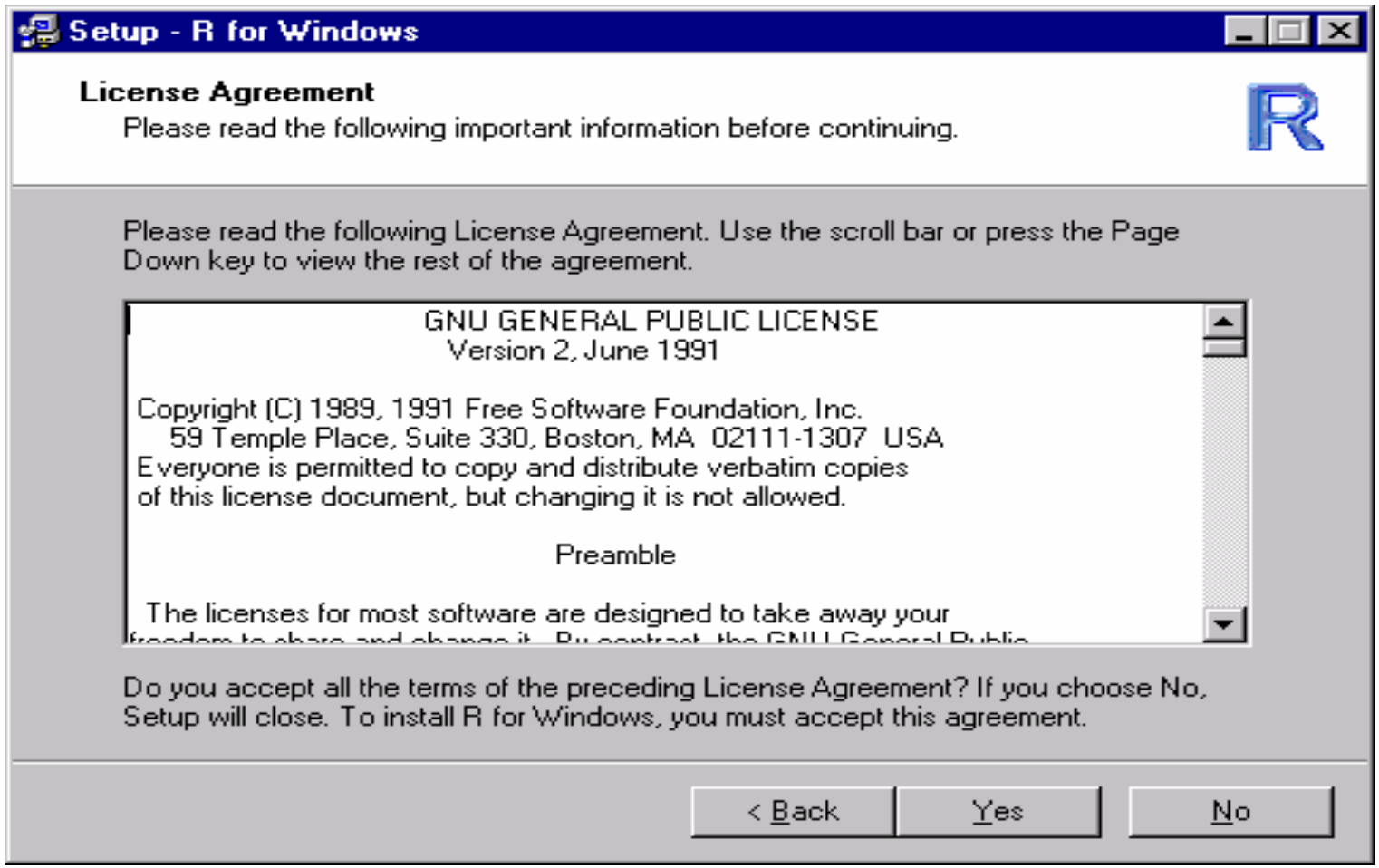
- Download der R-Files unter der Adresse **`http://www.r-project.org/welcome.html`**
- Erwerb einer CD am Lehrstuhl zum (Selbstkosten-) Preis von EUR 3,-

Einführung: Installation von R

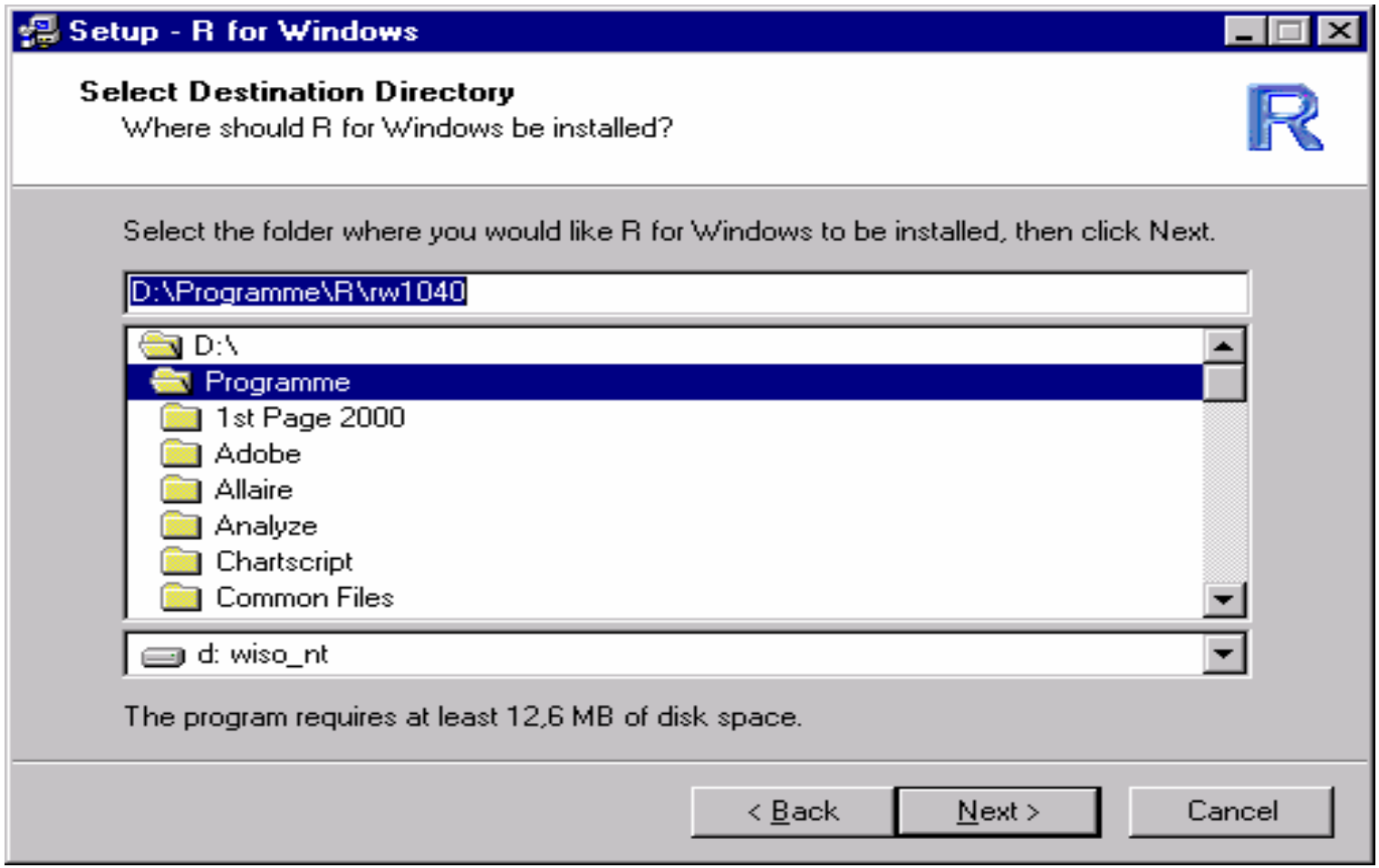
- Installationsanleitung auf CD-ROM
- Starten der Setup-Datei in `E:\R.1.6.1\Base\SetupR.exe`
- Einbinden der Zusatz-Pakete durch entpacken der zip-Dateien aus dem Verzeichnis `E:\R.1.6.1\Base`
- „`E`“ ist Buchstabe des CD-Rom-Laufwerks.

Einführung: Installation von R

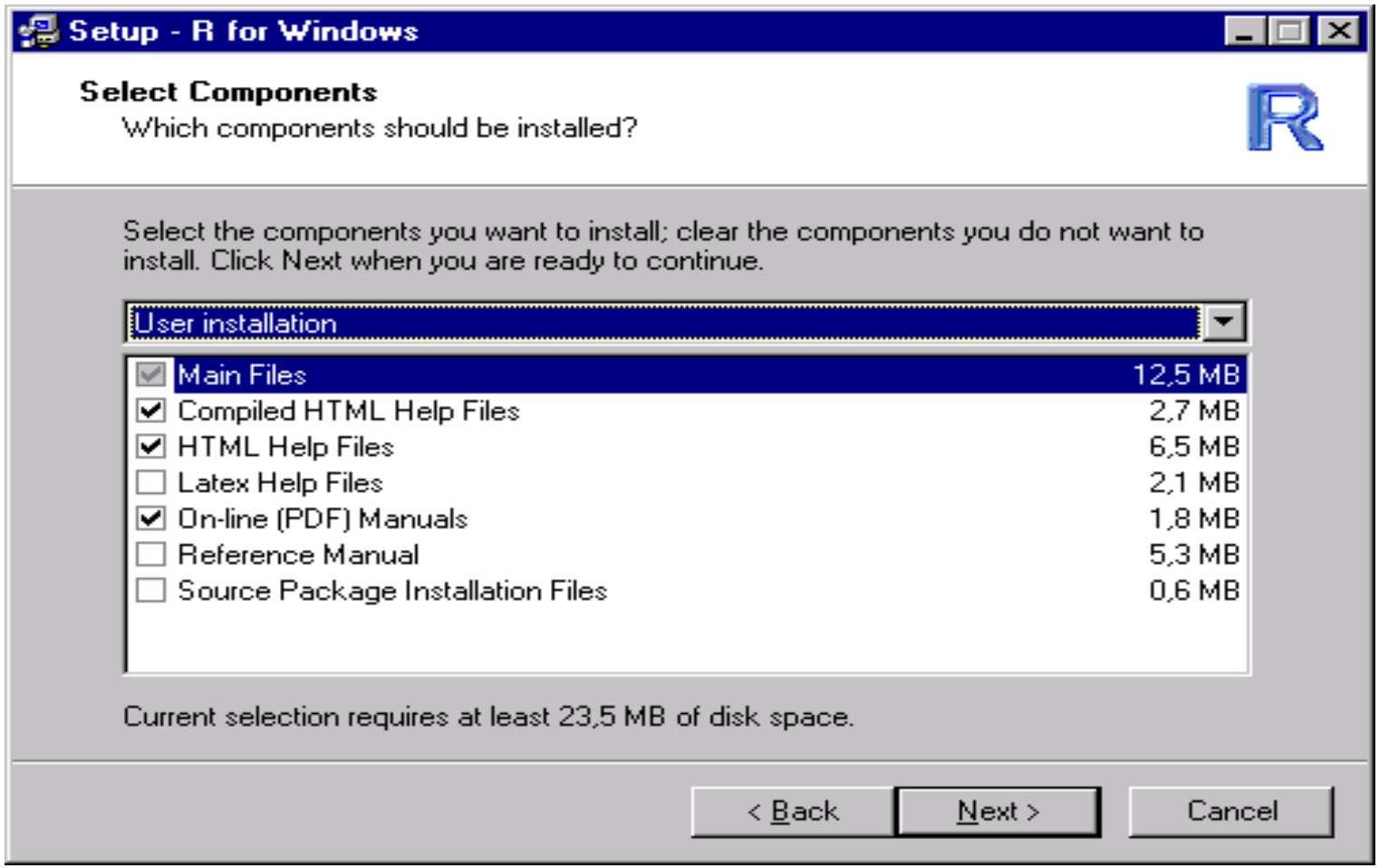
Einführung: Installation von R



Einführung: Installation von R



Einführung: Installation von R



Einbinden von Zusatzpaketen

- R aufrufen.
- Im Menü *Packages* auswählen.
- *Install packages from local zip file*... auswählen.
- *Windows* und *contrib* auswählen.
- Verzeichnis auswählen.
- Benötigtes Package auswählen (z.B acepack.zip) und *Öffnen* anklicken.
- Evtl. Wiederholung der Punkte.

Literatur zu R 1/4

- **Baron, J.** (2002): *R reference card* (auf CD)
- **Cribari-Neto und Zarkos** (1999): *R: Yet another econometric programming environment* (Kopiervorlage am Lst.)
- **Faraway, J.** (2002): Practical Regression and Anova using R, (auf CD)

Literatur zu R

- **Fischer, M.** (2002): *Einführung in R*: Powerpointfolien (auf Homepage)
- **Fischer, M.** (1999): *Einführung in R* (auf CD)
- **Maindonald, J. H.** (2000): Data Analysis and Graphics Using R – An Introduction, (auf CD)

Literatur zu R

- **Paradis, E.** (2002): R for beginners, (auf CD)
- **R Development Core Team** (2000): The R Reference Index, Version 1.4, (auf CD)
- **Sawitzki, G.** (2001): Statistical Computing: Einführung in S, (auf CD)

Literatur zu R 4/4

- **Venables, B.** et al. (1998): *Notes on R: A programming Enverironment for Data Analysis and Graphis*
- **Verzani, J.** (2002): simpleR: Using R for introductory statistics, (auf CD) oder http://www.math.csi.cuny.edu/Statistics/R/simpleR)

Gliederung

- Einführung in **R**
- Zahlen und Vektoren
- Matrizen
- Ein- und Auslesen von Daten
- Erzeugung von Grafiken
- Lineares Modell
- Verteilungen
- Ergänzungen
- Zeitreihenanalyse

Zahlen und Vektoren

- Rechnen mit **R** im Command-Window

```
>1+2        [Addition]
>2*4        [Multiplikation]
>1/2        [Division]
>sqrt(2)    [Wurzel]
```

Zahlen und Vektoren

- Elementare Funktionen

```
>exp(2)          [Exponentialfunktion]
>log(2)          [Logarithmusfunktion]
>abs(-2)         [Absolutbetrag]
>sign(2)         [Vorzeichen]
>floor(2.2)      [Nächstkleinere Zahl]
>ceiling(2.2)    [Nächstgrößere Zahl]
>round(2.34,1)   [Rundung auf 1 NKS]
>sin(2)          [Sinusfunktion]
```

Erzeugung von Vektoren

- Vektor: geordnete Menge von Zahlen
- **R** unterscheidet nicht zwischen Zeilen- und Spaltenvektoren
- Erzeugung von Vektoren:
  - Mit dem combine-Befehl `c`
  - Mit dem sequence-Befehl `seq`
  - Mit dem Operator `:`
  - Mit dem repeat-Befehl `rep`

Erzeugung von Vektoren

- Beispiele:
  - Vektor 1: `c(9,3,2)`
  - Vektor 2: `c(1,1,1,1)`
  - Einfacher: `rep(1,4)`
  - Vektor 3: `c(1,2,3,4)`
  - Einfacher: `1:4`
  - Vektor 3: `c(1,3,5,7)`
  - Einfacher: `seq(1,7,2)`

Namensgebung von Vektoren

- Zuweisungsoperator _ bzw. `<-`
  - Bsp. 1: `>a_seq(-3,3,0.05)`
  - Bsp. 2: `>b<-c(1,3,2,3,4)`
- Anzeigen aller Objekte:

  Befehl `list`: `>ls()`
- Löschen einzelner Objekte:

  Befehl `remove`: `>rm(x,y)`
- Wichtig: Vor Beenden der Sitzung speichern !!!

Rechnen mit Vektoren

- Standardmanipulationen:

```
>x_rep(1,4)
>y_1:4
```

`>x+y`	[Addition]
`>x-y`	[Subtraktion]
`>x/y`	[Elementweise Division]
`>x*y`	[Elementweise Multiplikation]
`>x%*%y`	[Vektor-Multiplikation]

Rechnen mit Vektoren

- Standardfunktionen

`>sum(x)`	[Summe der Elemente]
`>mean(x)`	[Mittelwert]
`>length(x)`	[Länge des Vektors]
`>min(x)`	[Kleinstes Element]
`>max(x)`	[Größtes Element]
`>prod(x)`	[Produkt der Elemente]
`>cumsum(x)`	[Aufsummation]
`>is.vector(x)`	[Vektortest]

Rechnen mit Vektoren

- Selektion von Elemente

```
>z_c(1,2,2,2,5,5,5)
```

`>z[3]`	[3-tes Element]
`>z[c(3,5)]`	[3-tes u. 5-tes Element]
`>z[c(1,2,3)]`	[ersten 3 Elemente]
`>z[1:3]`	[einfacher]
`>z[-7]`	[Vektor ohne 7-tes]
`>z[-c(1,5)]`	[Vektor ohne Nr. 1/5]

Gliederung

- Einführung in **R**
- Zahlen und Vektoren
- Matrizen
- Ein- und Auslesen von Daten
- Erzeugung von Grafiken
- Lineares Modell
- Verteilungen
- Ergänzungen
- Zeitreihenanalyse

Erzeugung von Matrizen

- Matrix: in Zeilen und Spalten geordnete Menge von Zahlen
- Beispiel: 3 x 3-Matrix

$$A = \begin{pmatrix} 2 & 1 & 0 \\ 2 & -2 & 3 \\ 0 & 1 & 2 \end{pmatrix}$$

- Erzeugung von Matrizen:
  - Mit dem Befehl `matrix`
  - Mit dem Befehl `diag`

Erzeugung von Matrizen

- Matrizen mit dem Befehl `matrix`

  Syntax: matrix(x,ncol,nrow,byrow)

x	Daten als Vektor
ncol	Anzahl der Spalten (columns)
nrow	Anzahl der Zeilen (rows)
byrow	T = zeilenweise auffüllen

  Bsp.: `matrix(c(2,2,4,4),2,2,T)`

- Matrizen mit dem Befehl `diag`

  Bsp. 1: `diag(4)`

  Bsp. 2: `diag(c(1,2,3))`

Rechnen mit Matrizen

- Standardmanipulationen:

```
>vect_c(1,2,0,1)
>A_matrix(vect,ncol=2,nrow=2)
>B_diag(2)
```

`>A±B`	[Addition/ Subtraktion]
`>A*B`	[Elementweise Multiplikation]
`>A/B`	[Elementweise Division]
`>A%*%B`	[Vektor-Multiplikation]
`>t(A)`	[Transponierung]
`>solve(A)`	[Invertierung]

Rechnen mit Matrizen

- Selektion von Elemente

```
>A_diag(4)
```

`>A[1,1]`	[Element (1,1)]
`>A[3,]`	[3-te Zeile]
`>A[,2]`	[2-te Spalte]
`>A[c(1,3),]`	[1-te und 3-te Zeile]
`>A[,c(1,3)]`	[1-te und 3-te Spalte]
`>A[-4,]`	[ohne 4-te Zeile]
`>z[,-2]`	[ohne 2-te Spalte]

Rechnen mit Matrizen

- Standardfunktionen

`>sum(A)`	[Summe der Elemente]
`>nrow(A)`	[Anzahl der Zeilen]
`>ncol(A)`	[Anzahl der Spalten]
`>length(A)`	[Anzahl der Elemente]
`>diag(A)`	[Hauptdiagonale]
`>solve(A)`	[inverse Matrix]
`>t(A)`	[transponierte Matrix]
`>eigen(A)`	[Eigenwerte/~vektoren]
`>dim(A)`	[Dimension der Matrix]

Rechnen mit Matrizen

- Erweiterte Funktionen

`>crossprod(A)` [Berechne A‘A]

`>rbind(A,B)` [Verbinde A und B zeilenweise]

`>cbind(A,B)` [Verbinde A und B spaltenweise]

Anwendung der Funktion `sum` auf Zeilen `>apply(A,1,sum)`

Anwendung der Funktion `sum` auf Spalten `>apply(A,2,sum)`

Auch andere Funktionen möglich!

Gliederung

- Einführung in **R**
- Zahlen und Vektoren
- Matrizen
- Ein- und Auslesen von Daten
- Erzeugung von Grafiken
- Lineares Modell
- Verteilungen
- Ergänzungen
- Zeitreihenanalyse

Einlesen von Daten

- mit Hilfe der Tastatur

  `werte_scan()`

  Eingabe von Tastatur

  Bestätigung mit *Return*

  Abschluß mit *2-mal Return*

- aus einer Datei

  Download und Entpacken der Daten consors von Homepage (-> Download -> Hauptstudium)

  ```
  cons_scan(``u:/542700.txt``)
  cons
  ```

Gliederung

- Einführung in **R**
- Zahlen und Vektoren
- Matrizen
- Ein- und Auslesen von Daten
- Erzeugung von Grafiken
- Lineares Modell
- Verteilungen
- Ergänzungen
- Zeitreihenanalyse

Erzeugung von Grafiken I

- Bsp: Darstellung der Consors-Kurse

```
plot(cons)
```

```
plot(cons,type="l")
```

→ Verbinden durch Linie

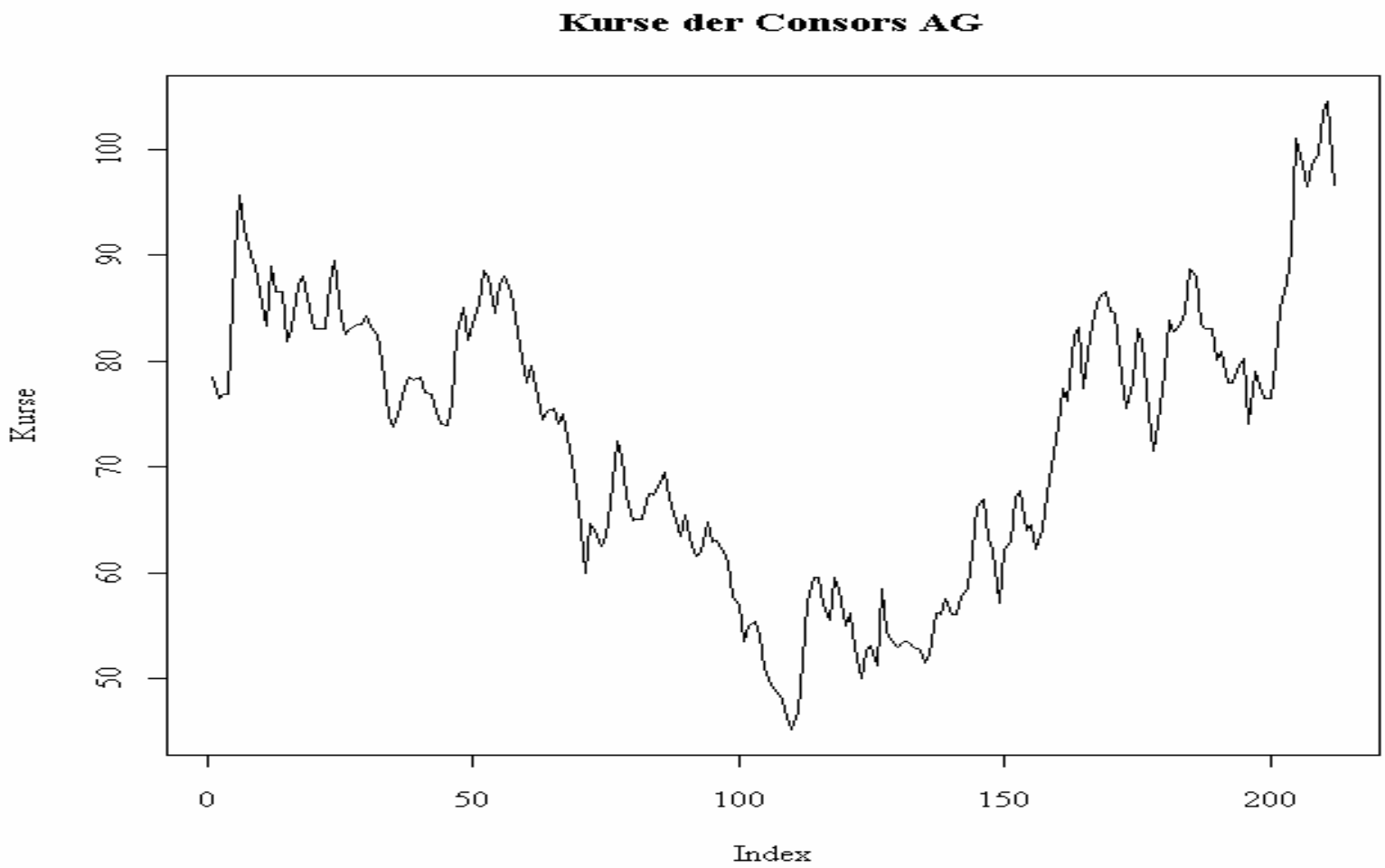
```
plot(cons,type="l",main="Kurse
Consors")
```

→ Einfügen einer Überschrift

```
plot(cons,type="l",main="Kurse
Consors", ylab="Kurse")
```

→ Beschriften der y-Achse

Erzeugung von Grafiken II



Erzeugung von Grafiken III

- Bsp: Darstellung der Consors-Renditen

```
rcons_diff(log(cons),1)

plot(cons,type="l",main="Rendite
n Consors", ylab="Renditen")
```

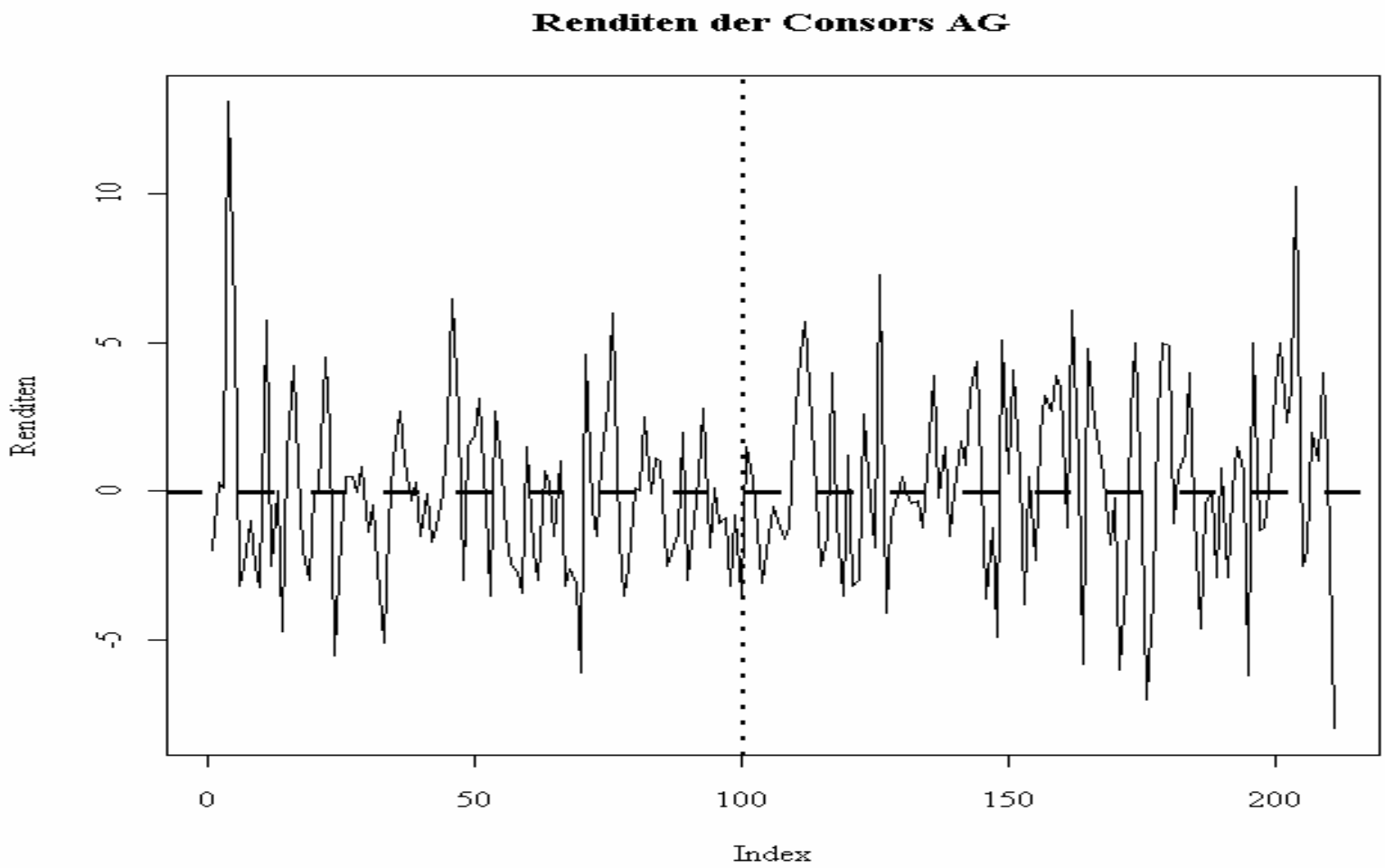
- Einfügen einer horizontalen Linie

```
abline(h=0,lty=2,lwd=3)
```

- Einfügen einer vertikalen Linie

```
abline(v=100,lty=3,lwd=1)
```

Erzeugung von Grafiken IV



Erzeugung von Grafiken V

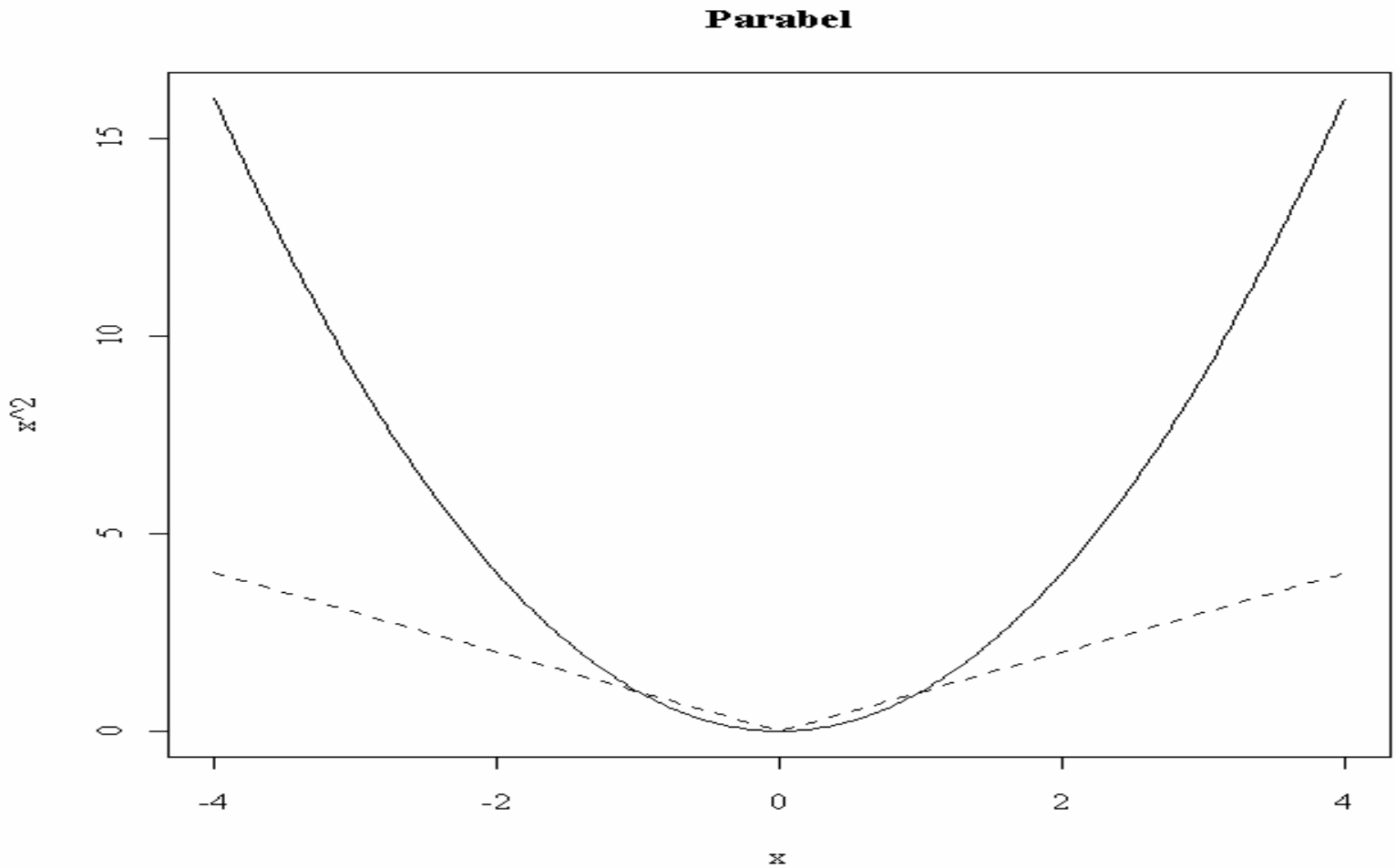
- Bsp: Plot einer Parabel von [-4,4]

```
x_seq(-4,4,,200)
plot(x,x^2,type="l")
```

- Bsp: Ergänzung des Plots durch Kurve von |x|.

```
lines(x,abs(x),lty=2)
```

Erzeugung von Grafiken VI



Gliederung

- Einführung in **R**
- Zahlen und Vektoren
- Matrizen
- Ein- und Auslesen von Daten
- Erzeugung von Grafiken
- Lineares Modell
- Verteilungen
- Ergänzungen
- Zeitreihenanalyse

Lineares Modell I

- Schätzung eines linearen Modells
  $Y_t=\alpha+\beta X_t+U_t$, t=1,...,T
  mit dem Befehl `lm` („linear model“)
- Bsp: Regression der Kurse von Consors auf die Zeit, d.h. X_t=t.

```
y_cons
x_1:212
lm(y~x)      [einfaches lineares Modell ]
lm(y~x-1)    [Modell ohne Achsenabschnitt]
lm(y~x+z)    [Modell mit 2 Exogenen x,z]
```

Lineares Modell II

- Ausgabe erweiterter Regressionsergebnisse

```
summary(lm(y~x))
```

- Weitere Ausgabebefehle

`resid(lm(y~x))`	[Residuen]
`coef(lm(y~x))`	[Koeffizienten]
`predict(lm(y~x))`	[geschätzten Werte]
`anova.lm(lm(y~x))`	[ANOVA]
`plot(lm(y~x))`	[Fit grafisch]
`influence.measure`	[Ausreißeranalyse]

Tests zum linearen Modell

- Testverfahren auf Linearität
  - → Reset-Test von Ramsey (1969)
  - → Harvey-Collier-Test (1977)
  - → Rainbow-Test (1982)
  - → Terasvirta NN-Test (1993)
  - → White NN-Test (1993)
- Testverfahren auf Heteroskedastie
  - → Durbin-Watson Test (1950)
  - → Goldfeld-Quandt Test (1965)
  - → Breusch-Pagan Test (1978)
  - → Harrison-McCabe Test (1979)

Tests zum linearen Modell

- Testverfahren auf Autokorrelation
  - → Durbin-Watson Test (1950)
  - → Breusch-Godfrey-Test (1978)
- Testverfahren auf Normalverteilung
  - → KS-Test (1950)
  - → Chiquadrat-Anpassungstest (1978)
  - → Jarque-Bera-Test (1980)
  - → Shapiro-Wilk-Royston-Test (1982)
  - → Normal-Quantil-Plot
  - → T3-Plot von Gosh (1996)

RESET-Test von Ramsey

- Reset-Test (Bibliothek: lmtest)
- Test auf funktionale Form des Modells
- Nullhypothese: Modell linear
- R-Funktion: `reset`
- `reset(formula, power = 2:3, type = "fitted")`

  `formula` → Symbolische Modellbeschreibung

  `power` → Eingeschlossene Potenzen

  `type` → Default =„fitted“

Goldfeld-Quandt Test

- Goldfeld-Quandt-Test (Bibliothek: lmtest)
- Test auf Heteroskedastie
- Nullhypothese: Homoskedastie
- R-Funktion: `gqtest`
- `gqtest(formula, point=0.5)`

`formula`	→ Symbolische Modellbeschreibung
`point`	→ Bruchpunkt der Varianzen

Breusch-Pagan Test

- Breusch-Pagan (Bibliothek: lmtest)
- Test auf Heteroskedastie
- H_0: keine Heteroskedastie
- R-Funktion: `bptest`
- `bptest(formula, varformula=NULL)`

  `formula` → Symbolische Modellbeschreibung

  `point` → Formel für Varianzbeschreibung

Durbin-Watson Test

- Durbin-Watson-Test (Bibliothek: lmtest)
- Nullhypothese: Keine Autokorrelation
- R-Funktion: `dwtest`
- `dwtest(formula, alternative = c("greater", "two.sided", "less"), iterations = 15)`

`formula` → Symbolische Modellbeschreibung

`alternative` → Spezifizierung von H_A

`iterations` → Anzahl der Iterationen für p-Wert

Breusch-Godfrey Test

- Breusch-Godfrey Test (Bibliothek: lmtest)
- Nullhypothese: Keine Korrelation
- R-Funktion: `bgtest`
- `bgtest(formula, order = 1, type = c("Chisq", "F"))`

  `formula` → Symbolische Modellbeschreibung

  `order` → Ordnung der Korrelation

  `type` → Chiquadrat oder F-Test

Kolmogorov-Smirnov Test

- KS Test (Bibliothek: ctest)
- Nullhypothese: x und y aus gleichen Verteilung
- R-Funktion: `ks.test`
- `ks.test(x, y, alternative= c("two.sided", "less", „greater"))`

  `x` → Vektor mit den Daten

  `y` → Daten oder Verteilung

  `alternative` → Festlegung der Alternative

Jarque-Bera-Test

- JB-Test (Bibliothek: tseries)
- Nullhypothese: Normalverteilung
- R-Funktion: `jarque.bera.test`
- `jarque.bera.test(x)`

Shapiro-Wilk-Royston-Test

- SWR (Bibliothek: ctest)
- Nullhypothese: Normalverteilung
- R-Funktion: `shapiro.test`
- `shapiro.test(x)`

T3-Plot

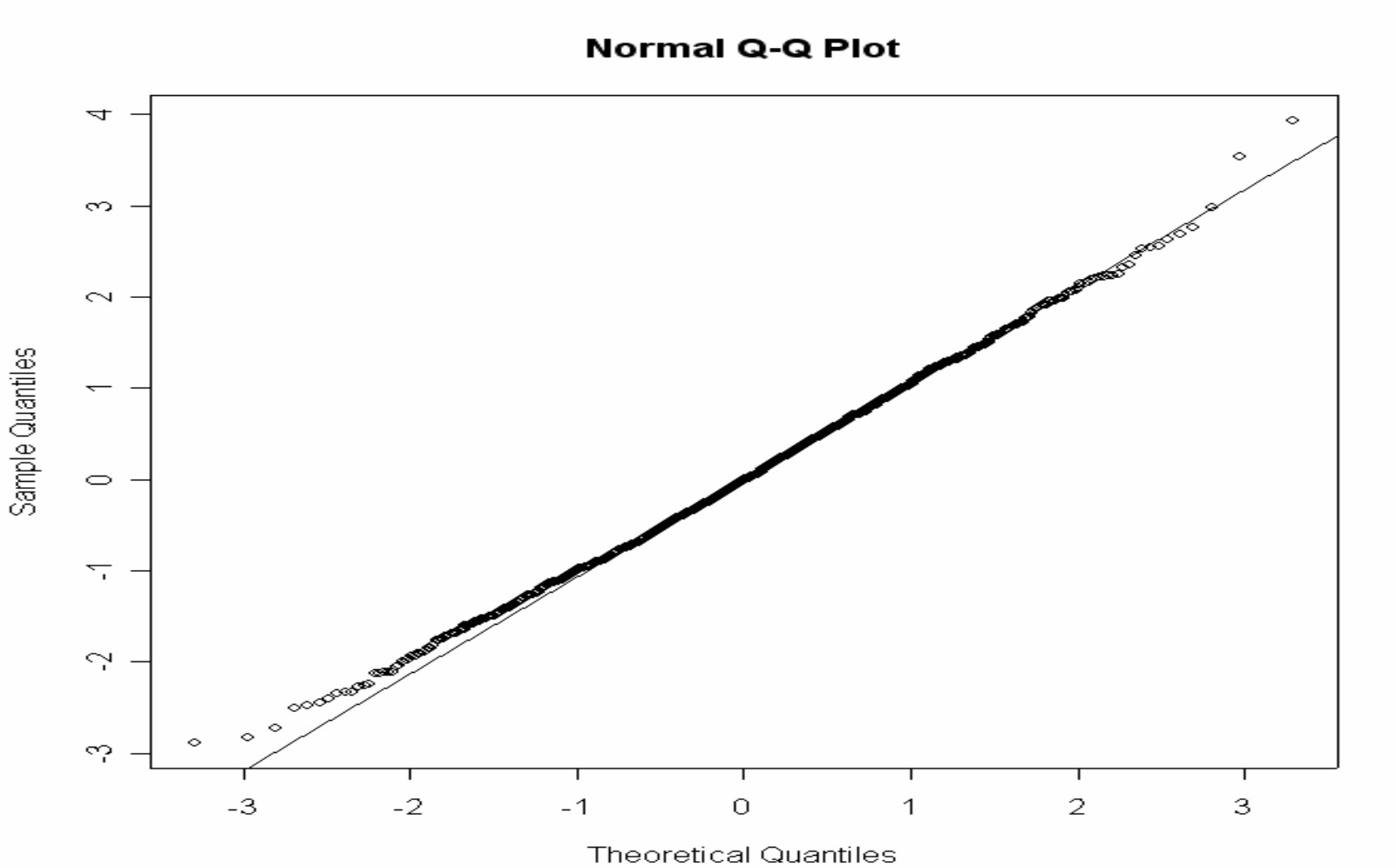
- T3-Plot nach Gosh:

  Funktion `t3plot` als Download auf Homepage

Normal-Quantile-Plot

- NQ-Plot: Normal-Quantile-Plot
  - Funktion: `qqnorm(x)`
  - „Ideallinie“: `qqline(x)`
- Beispiel: Normal-Quantile-Plot für 1000 normalverteilte Zufallsvariablen
  - `x_rnorm(1000) # Zufallszahlen`
  - `qqnorm(x)  # NQ-Plot`
  - `qqline(x)  # NQ-Linie`

NQ-Plot: Beispiel



Gliederung

- Einführung in **R**
- Zahlen und Vektoren
- Matrizen
- Ein- und Auslesen von Daten
- Erzeugung von Grafiken
- Lineares Modell
- Verteilungen
- Ergänzungen
- Zeitreihenanalyse

Verteilungen

- **R** berechnet
  - Werte der Dichtefunktion `d`
  - Werte der Verteilungsfunktion `p`
  - Werte der Quantilsfunktion `q`
  - Zufallszahlen einer Verteilung `r`
- Zwei Bestandteile
  - Funktionskürzel, z.B. `d` für Dichtefunktion
  - Verteilungskürzel, z.B. `norm` für Normalverteilung

Verteilungen

- Bsp 1: Dichtefunktion einer Normalverteilung mit Mittelwert 0 und Varianz 1 an der Stelle 2.

```
dnorm(2,0,1)
```

- Bsp 2: 95%-Quantil einer χ^2-Verteilung mit 3 Freiheitsgraden.

```
qchisq(0.95,3)
```

- Bsp 3: 10 t-verteilte Zufallsvariablen mit 4 Freiheitsgraden.

```
rt(10,4)
```

Verteilungen

- Bsp. 4: Zeichnen Sie die Dichtefunktion einer Standardnormalverteilung im Bereich von -4 bis 4.
- Lösung:

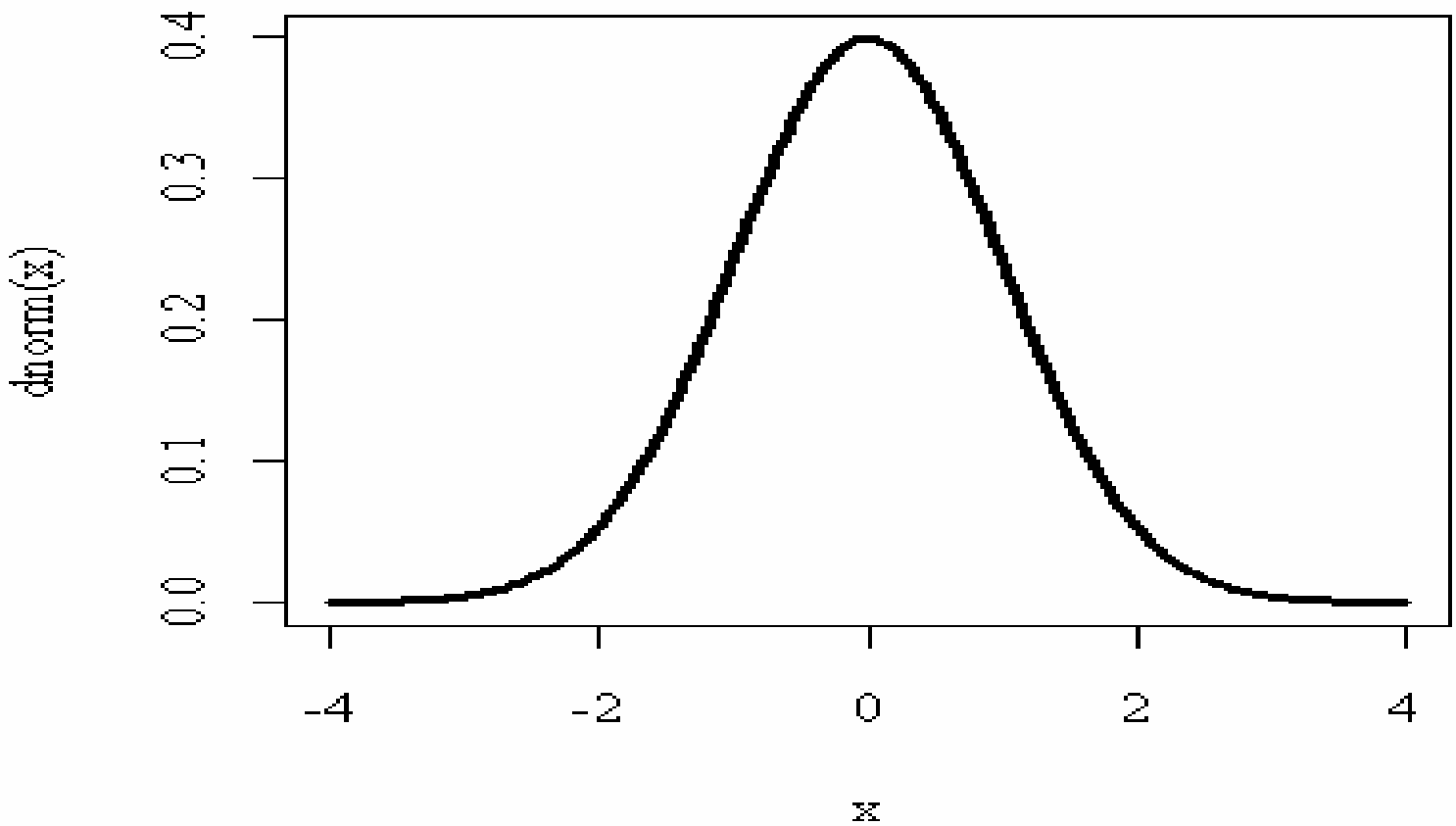
  Erzeugung von 200 Stützstellen durch

  ```
  x_seq(-4,4,,200)
  ```

  Plot der Dichtefunktion durch

  ```
  plot(x,dnorm(x),type='l')
  ```

Verteilungen



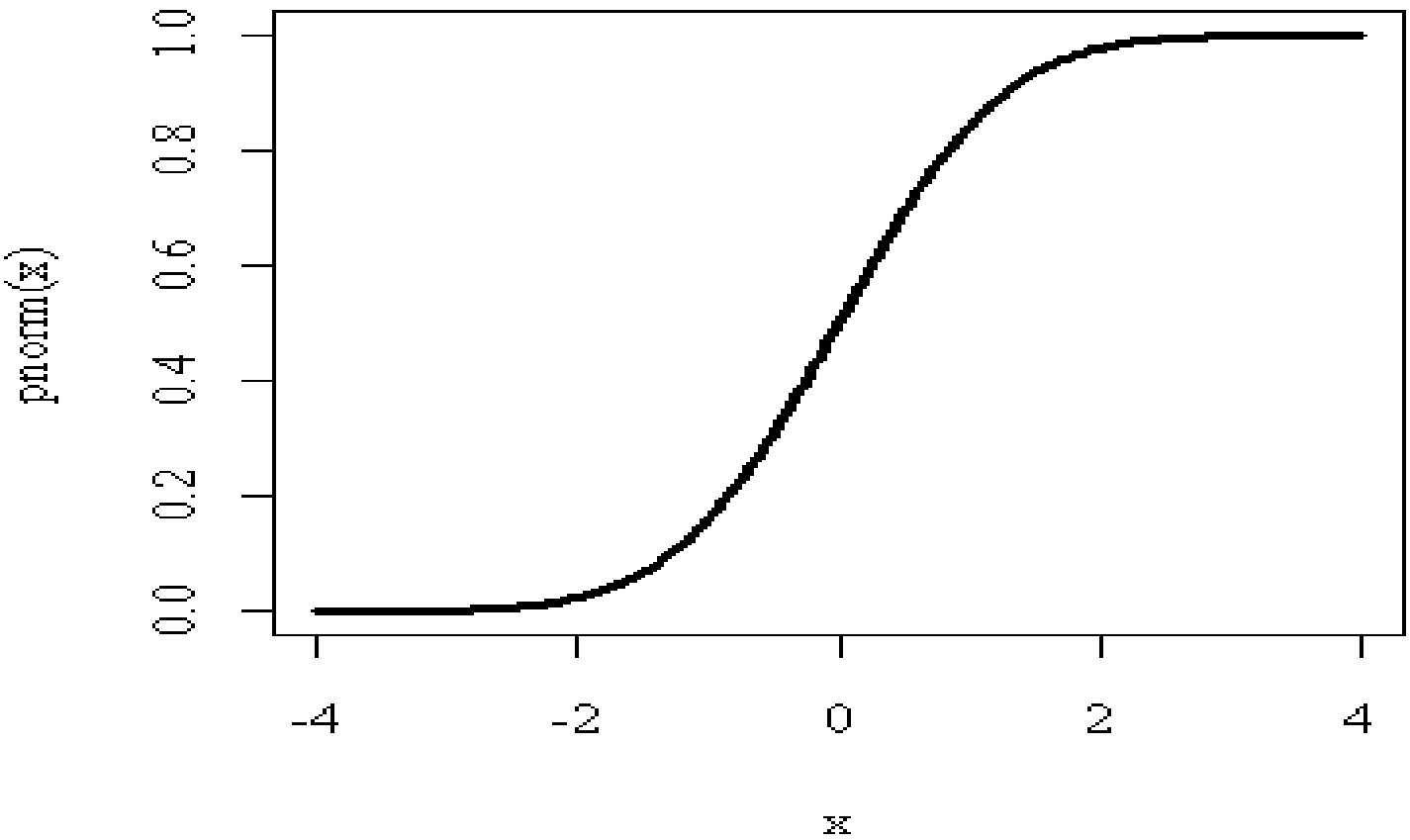
Verteilungen

- Bsp. 5: Zeichnen Sie die Verteilungs-funktion einer Standardnormalverteilung im Bereich von -4 bis 4.
- Lösung:

  Plot der Verteilungsfunktion durch

  ```
  plot(x,pnorm(x),type='l')
  ```

Verteilungen

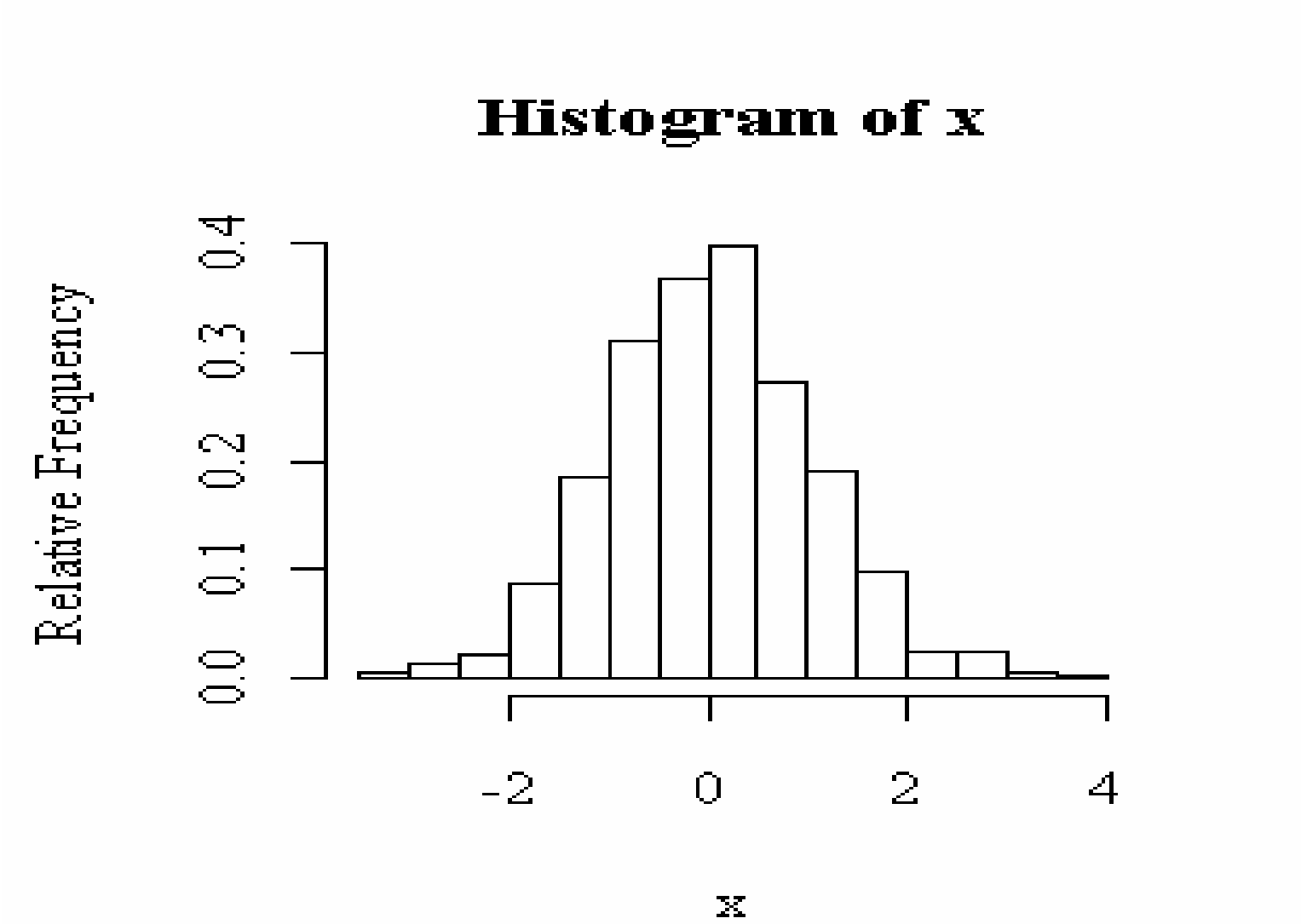


Verteilungen

- Plot einer empirischen Dichtefunktion
  - Histogramm
  - Nichtparametrische Dichteschätzung
- Bsp. 6: Histogramm für 1000 normalverteilte Zufallsvariablen

```
zz_rnorm(1000)
hist(zz,nclass=10,prob=T)
hist(zz,nclass=50,prob=T)
```

Verteilungen

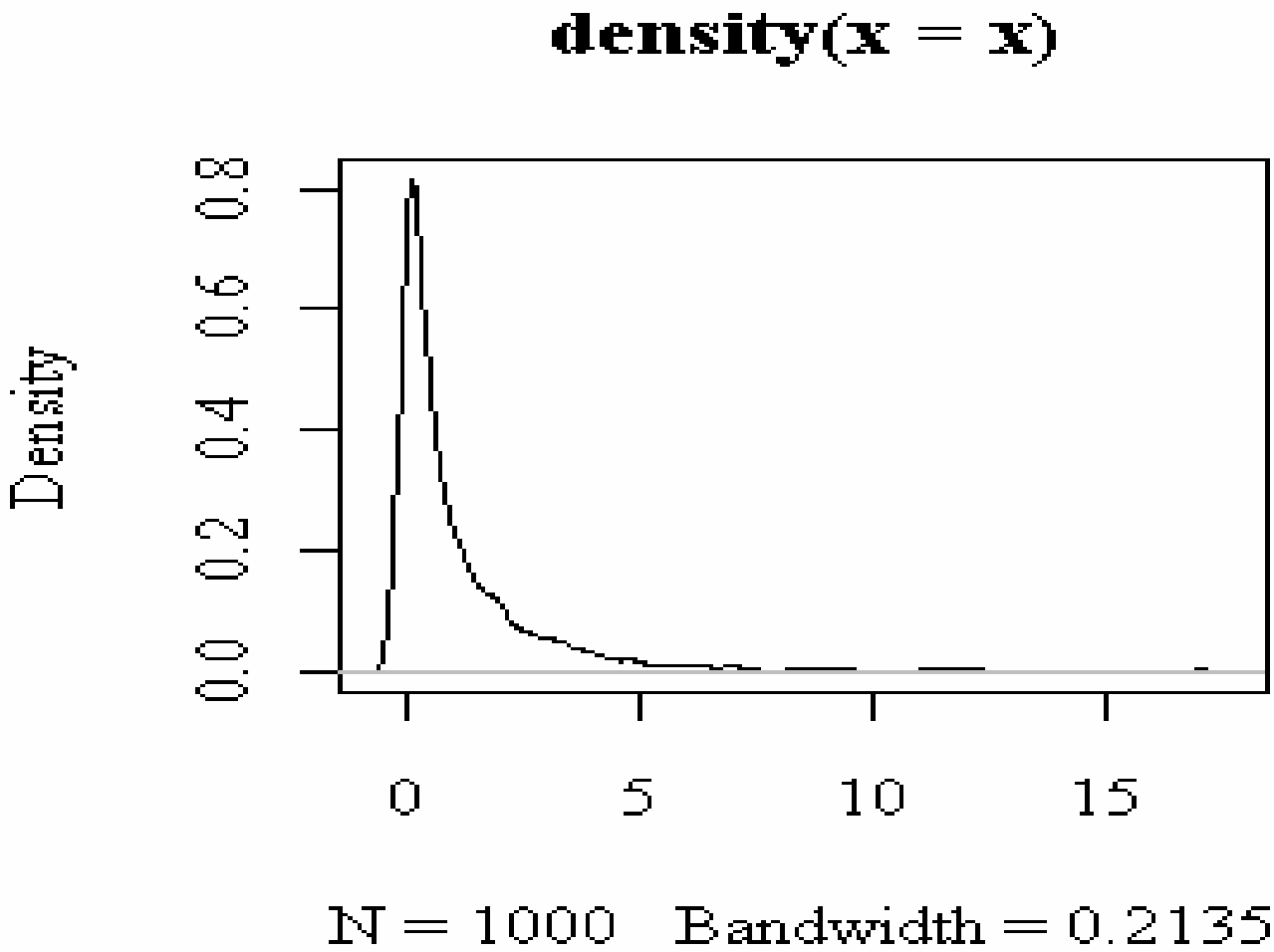


Verteilungen

- Bsp. 7: Nichtparametrische Dichteschätzung für 1000 F-verteilte Zufallsvariablen mit 1 Zähler und 10 Nennerfreiheitsgraden

```
zz_rf(1000,1,10)
plot(density(zz))
```

Verteilungen

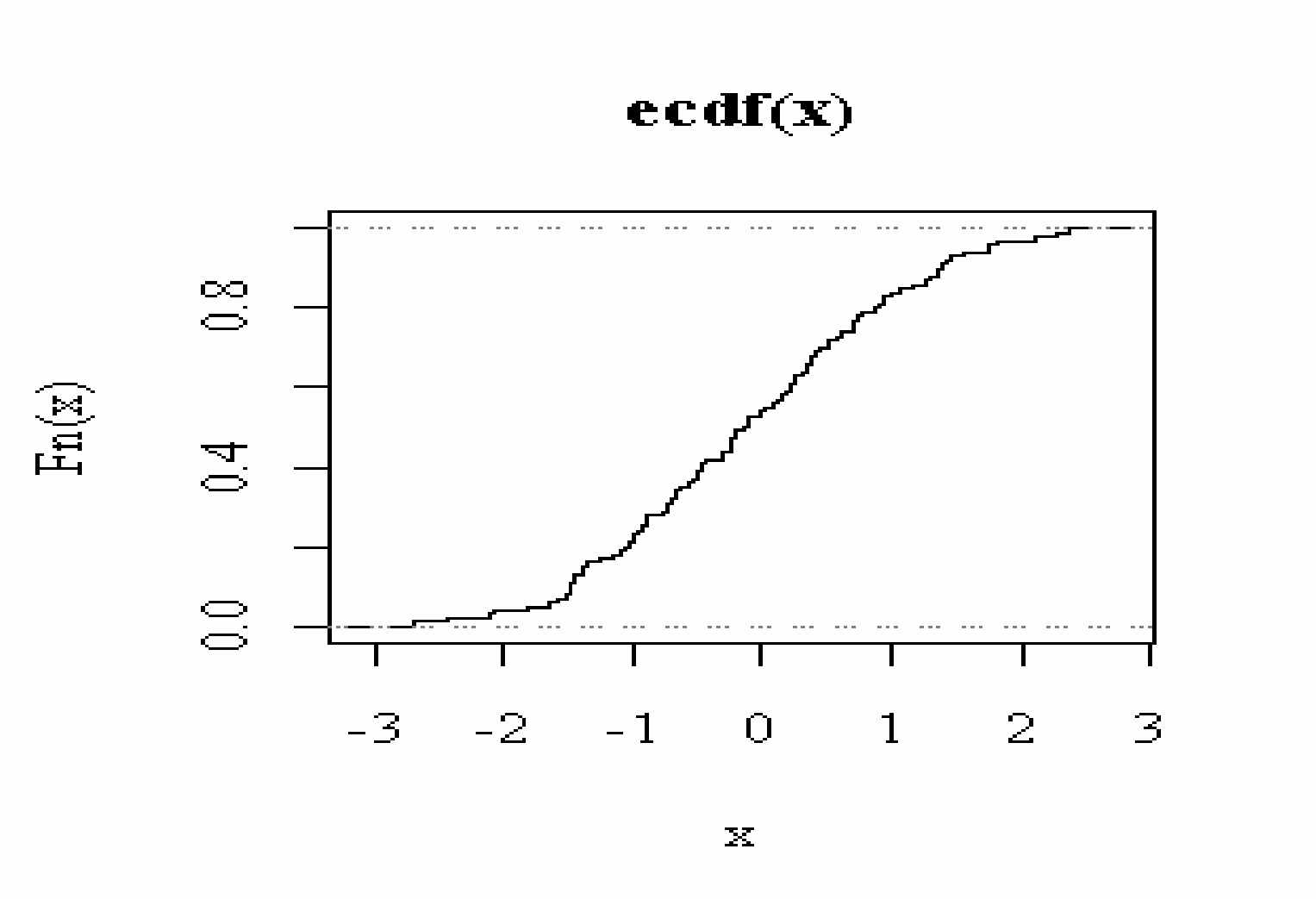


Verteilungen

- Empirischen Verteilungsfunktion (EDF)
  - Zusatzpaket *stepfun*
  - Einbinden durch `library(stepfun)`
- Bsp. 8: Zeichnung der EDF von 100 N(0,1)-verteilten Zufallsvariablen

```
z_rnorm(100,0,1)
plot(ecdf(z),do.points=F,verticals=T)
```

Verteilungen

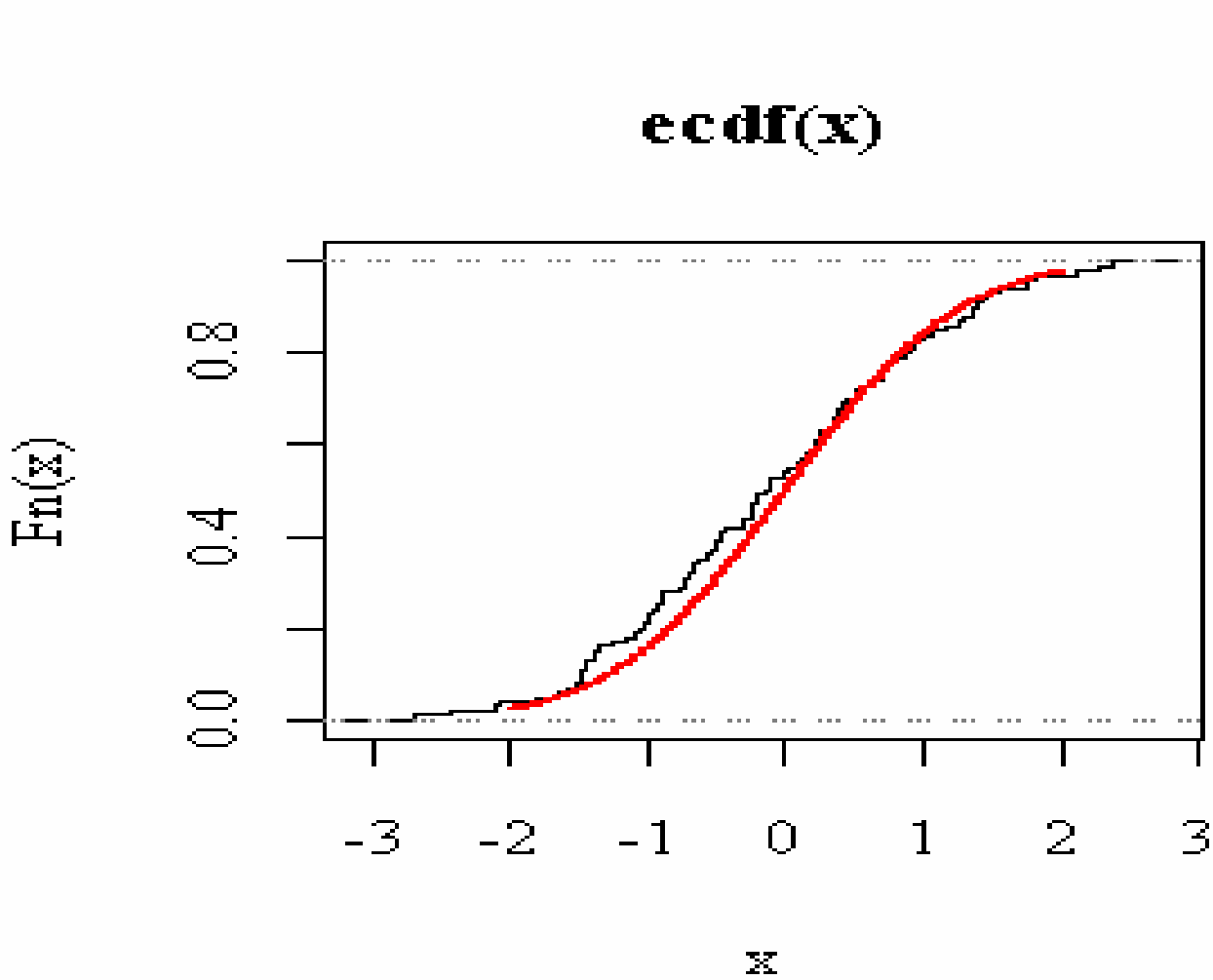


Verteilungen

- Bsp. 8: Ergänzen Sie den ECDF-Plot um die tatsächliche Verteilungsfunktion einer N(0,1)-verteilten Zufallsvariable.

```
x_seq(-2,2,,200)
lines(x,dnorm(x),lty=1,col=2)
```

Verteilungen



Gliederung

- Einführung in **R**
- Zahlen und Vektoren
- Matrizen
- Ein- und Auslesen von Daten
- Erzeugung von Grafiken
- Lineares Modell
- Verteilungen
- Ergänzung: Zusatzpakete & Funktionen
- Zeitreihenanalyse

Einbinden von Zusatzpakten

- Aufruf durch Befehl `library`

  ```
  library(tseries)
  ```

- Relevante Pakete

tseries	[Zeitreihenpaket]
stepfun	[Empirische Verteilung]
ctest	[Standardtestverfahren]
lmtest	[]

Schreiben einer eigenen Funktion

- Erstmalige Bearbeitung durch `fix`
- Erneute Bearbeitung durch `edit`
- Beispiel

  `summe_fix(summe)`

  in Editor: `function(a,b){a+b}`

  Datei speichern

  `summe(2,3)`

  `summe_edit(summe)`

  in Editor: `function(a,b){(a+b)^2}`

  Datei speichern `summe(2,3)`

Gliederung

- Einführung in **R**
- Zahlen und Vektoren
- Matrizen
- Ein- und Auslesen von Daten
- Erzeugung von Grafiken
- Lineares Modell
- Verteilungen
- Ergänzungen
- Zeitreihenanalyse

Einzelwerttests I

- Funktion aus Bibiothek ts

```
acf(x,lag.max,type,plot)
```

`x`
→ Daten

`lag.max`
→ Länge der Verzögerung

`type=„correlation“ bzw. „covariance“`
→ Typ

`plot`
→ Grafische Ausgabe

Einzelwerttests II

- Funktion aus Bibiothek ts

```
pacf(x,lag.max,type,plot)
```

`x`
→ Daten

`lag.max`
→ Länge der Verzögerung

`type=„correlation“ bzw. „covariance“`
→ Typ

`plot`
→ Grafische Ausgabe

Mehrwerttests

- Funktion aus Bibiothek ts

```
Box.test(x,lag=1,type)
```

`x`

→ Daten

`lag=1`

→ Länge der Verzögerung

`type=„Box-Pierce“ bzw. „Ljung-Box“`

→ Wahl der Tests

Schätzung von AR-Prozessen

- Funktion aus Bibiothek ts

  `ar(x,aic=T,ord.max,method)`

  `x` → Daten

  `aic=T` → Automatische Modellauswahl

  `order.max=p` → Feste Modellordnung

  `method=„yule-walker“,“ols“,“mle“`
  → Wahl der Schätzmethode

  `demean=T` → Schätzung eines Mittelwerts

Schätzung: ARMA-Modelle 1

- Regressionsansatz von Durbin
- Idee für ARMA(1,1):

$$X_t=\phi X_{t-1}-\theta U_{t-1}+U_t$$

1. Schritt: Anpassen eines AR-Prozesses hoher Ordnung, z.B. k=8

2. Schritt: Schätzung der Residuen W_t für AR-Prozeß

3. Schritt: Ersetze unbekannte Residuen der ARMA-Gleichung U_t durch geschätzte Residuen der AR-Gleichung W_t

4. Schritt: KQ-Schätzung der modifizierten Gleichung $X_t=\phi X_{t-1}-\theta W_{t-1}+W_t$

Schätzung: ARMA-Modelle 2

- Regressionsansatz von Durbin
- selbstgeschriebene Funktion

  `arma.durbin(x,p,q,k)`

  → Download von Homepage

  - `x` → Daten
  - `p` → Ordnung des AR-Teils
  - `q` → Ordnung des MA-Teils
  - `k` → Ordnung des AR-Teils für Startschätzung

Schätzung: ARMA-Modelle 3

- Conditional Least Square-Methode
- Funktion aus Bibiothek tseries

```
arma(x,order,lag,include.intercept)
```

`x` → Daten

`order=c(1,1)` → Ordnung des Modells

`lag=list(ar=c(1,3,7),ma=c(1,12))`
→ Feste Modellordnung

`include.intercept=TRUE`
→ Berücksichtigung eines Achsenabschnitts

Schätzung: ARMA-Modelle 4

- Arbeiten mit „ARMA-Objekten“

`ergeb_arma(r.consors)` → Objekt erzeugen

`summary(ergeb)` → Schätzergebnisse kompakt

`residuals(ergeb)` → Residuen der Anpassung

`fitted(ergeb)` → Geschätzte Zeitreihenwerte

`print(ergeb)` → Grafische Diagnostics

`coef(ergeb)` → Koeffizienten

Schätzung: ARMA-Modelle 5

- Maximum-Likelihood-Methode
- Funktion aus Bibiothek tseries

```
arima0(x,order,include.mean)
```

`x` → Daten

`order=c(1,0,1)` → Ordnung des Modells

`include.mean=TRUE`
→ Berücksichtigung eines Achsenabschnitts

GARCH-Modelle

- `garch:` Funktion aus Bibiothek tseries
- Maximum-Likelihood-Schätzung der Parameter eines GARCH(p,q)-Modells
- `garch(x, order = c(1, 1), coef = NULL, itmax = 200,...)`

  `x` → Daten

  `order=c(1,1)` → Ordnung des Modells

  `coeff` → Startvektor für GARCH-Parameter

  `itmax` → Maximale Iterationen bei Optimierung

GARCH-Modelle: Listenelemente

- order: Ordnung des GARCH-Modells
- coef: Geschätzte GARCH Koeffizienten des fitted model.
- n.likeli: Negativer log-likelihood Funktionswert evaluated at the coefficient estimates (apart from some constant).
- n.used: Anzahl der Beobachtungen
- residuals: Reihe der Residuen
- fitted.values: Bivariate Reihe der bedingten Standardabweichung

GARCH-Modelle

- Simulation eines GARCH-Prozesses
- Schreibe Funktion `garch.sim`

```
function(n=1000){
 a <- c(0.1, 0.5, 0.2)   # ARCH(2) coefficients
 e <- rnorm(n)           # Erzeugung der Störgrößen
 x <- double(n)
 x[1:2] <- rnorm(2, sd = a[1]/(1.0-a[2]-a[3]))
 for(i in 3:n){          # Generiere ARCH(2) Prozeß
 x[i]<-e[i]*sqrt(a[1]+a[2]*x[i-1]^2+a[3]*x[i-2] ^2)
 }
 return(x[101:(n+100)]) # Weglassen Einschwung
}
```

GARCH-Modelle

- Anpassung eines GARCH-Prozesses (1)

```
x <- garch.sim(1000)
ergeb <- garch(x, order = c(0,2))
summary(ergeb)      # Diagnostic Tests
plot(x.arch)        # Grafische Diagnostiken
```

- Anpassung eines GARCH-Prozesses (2)

```
data(EuStockMarkets)
dax <- diff(log(EuStockMarkets))[,"DAX"]
dax.garch <- garch(dax)   # Fit a GARCH(1,1) to returns
summary(dax.garch)        # ARCH effects are filtered
plot(dax.garch)           # conditional normality seems
```

Test auf Stationarität / Kointegration

- Testverfahren auf Integration/Station.
  - Augmented Dickey-Fuller-Test (1975)
  - Phillips-Perron-Test (1988)
  - KPSS-Test (1992)
- Testverfahren auf Kointegration
  - CRDW-Test (1983) [*nicht implementiert*]
  - Phillips-Ouliaris-Test (1990)

Augmented Dickey-Fuller-Test

- ADF-Test (Bibliothek: tseries)
- Einheitswurzeltest
- H_0: Prozeß hat Einheitswurzel
- R-Funktion: `adf.test`
- `adf.test(x,alternative = c("stationary","explosive"), k=trunc((length(x)-1)^(1/3)))`

`x`	→ Daten
`alternative`	→ Alternative (Default=„stationär“)
`k`	→ Lag-Ordnung des AR-Prozesses

Philips-Perron-Test

- PP-Test (Bibliothek: tseries)
- Einheitswurzeltest
- H_0: Prozeß hat Einheitswurzel
- R-Funktion: `pp.test`
- `pp.test(x, alternative = c("stationary", "explosive"))`

  `x` → Daten

  `alternative` → Alternative (Default=„stationär“)

KPSS-Test

- KPSS-Test (Bibliothek: tseries)
- Test auf Stationarität
- H_0: Niveau bzw. Trendstationarität
- R-Funktion: `kpss.test`
- `kpss.test(x, null = c("Level„ ,"Trend"))`

  `x` → Daten

  `null` → Nullhypothese (Default=„Level“)

Phillips-Ouliaris-Test

- PO-Test (Bibliothek: tseries)
- Test auf Kointegration
- H_0: Reihe ist nicht kointegriert.
- R-Funktion: `kpss.test`
- `po.test(x, demean = TRUE)`

  `x` → Zeitreihe

  `demean` → Achsenabschnitt in Regression