

---

# Eine Einführung in



---

*Universität Duisburg-Essen  
Lehrstuhl für Ökonometrie  
Martin Schmelzer*

UNIVERSITÄT  
DUISBURG  
ESSEN

*Offen im Denken*



# Inhaltsverzeichnis

<b>1 Grundlagen</b>	<b>4</b>
1.1 Was ist R, warum nutzen wir R? . . . . .	4
1.2 Installation von R und RStudio . . . . .	4
1.3 Die Benutzeroberfläche . . . . .	5
1.4 Die Dokumentation: wie war das noch gleich? . . . . .	6
1.5 Installation und Verwendung zusätzlicher Pakete . . . . .	6
<b>2 Stock and Watson - Chapter 2</b>	<b>7</b>
<b>3 Stock and Watson - Chapter 3</b>	<b>7</b>
<b>4 Stock and Watson - Chapter 4</b>	<b>7</b>

# 1 Grundlagen

## 1.1 Was ist R, warum nutzen wir R?

R ist eine Interpretersprache für statistische Anwendungen. Ihre Nutzergemeinde steigt stetig an. Jeder kann zum R-Projekt beitragen, indem er eigene Pakete entwickelt und den anderen Nutzern zur Verfügung stellt. Durch diesen Open Source-Charakter bietet **R** eine Vielzahl von analytischen Möglichkeiten für seine Anwender. Beispielsweise gibt es Pakete, um das Rechnen mit Zeitreihen zu vereinfachen oder aber um mit den API's von Facebook oder Twitter zu kommunizieren und so Social-Media-Daten abzufragen. Aber auch die großen "Datenkraken" ihrerseits haben **R** als nützliches Werkzeug entdeckt. Gerade die Analyse von "Big Data" (rechen- und speicherintensiv) ist eine Stärke von **R**.

## 1.2 Installation von R und RStudio

**R** selbst verfügt nicht über eine sonderlich ansprechende Benutzeroberfläche. Um **R** effektiv nutzen zu können, empfiehlt sich das Arbeiten mit einem Editor. Wir werden **RStudio** als Arbeitsoberfläche nutzen. Laden Sie zunächst beide Programme herunter:

### 1. Die eigentliche Programmierungsumgebung R:

<http://cran.r-project.org/>

Auf dieser Seite finden Sie die aktuelle Version von **R** für alle gängigen Betriebssysteme.

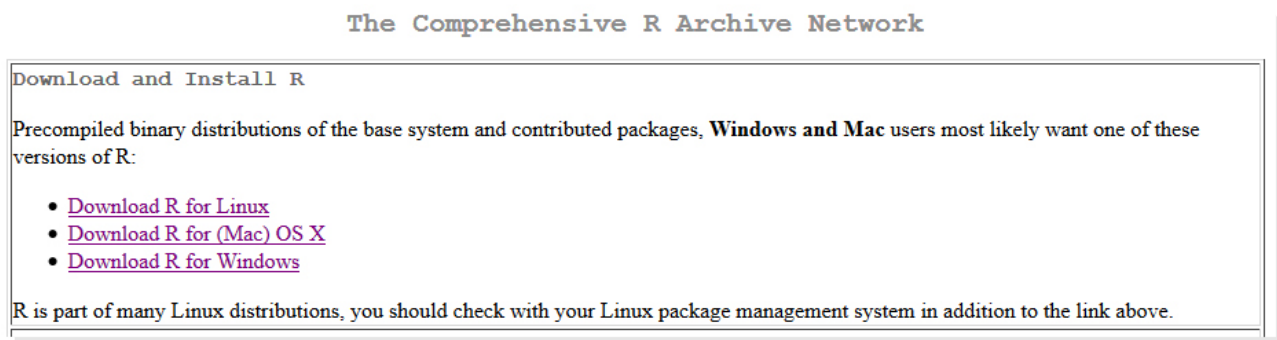
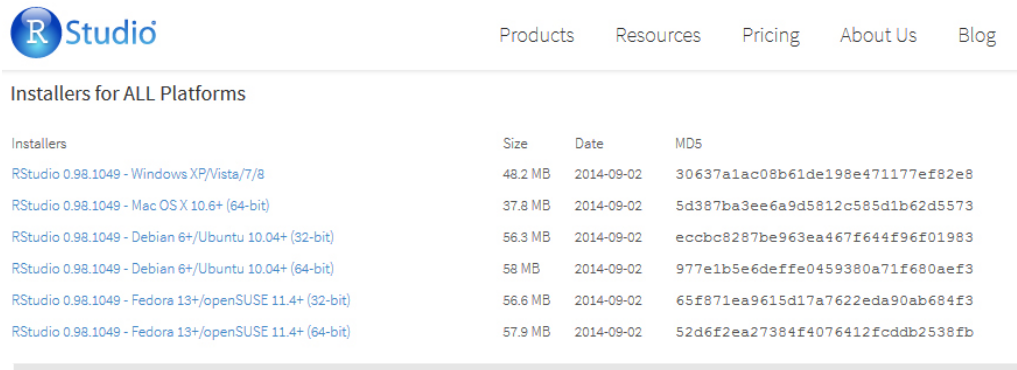


Abbildung 1: Der Downloadbereich der R-Project Homepage

## 2. Die Editor-Software RStudio:

<http://www.rstudio.com/products/rstudio/download/>

Auch hier laden Sie die entsprechende Datei herunter.



Installers for ALL Platforms			
Installers	Size	Date	MD5
<a href="#">RStudio 0.98.1049 - Windows XP/Vista/7/8</a>	48.2 MB	2014-09-02	30637a1ac08b61de198e471177ef82e8
<a href="#">RStudio 0.98.1049 - Mac OS X 10.6+ (64-bit)</a>	37.8 MB	2014-09-02	5d387ba3ee6a9d5812c585d1b62d5573
<a href="#">RStudio 0.98.1049 - Debian 6+/Ubuntu 10.04+ (32-bit)</a>	56.3 MB	2014-09-02	eccbc8287be963ea467f644f96f01983
<a href="#">RStudio 0.98.1049 - Debian 6+/Ubuntu 10.04+ (64-bit)</a>	58 MB	2014-09-02	977e1b5e6deffe0459380a71f680aef3
<a href="#">RStudio 0.98.1049 - Fedora 13+/openSUSE 11.4+ (32-bit)</a>	56.6 MB	2014-09-02	65f871ea9615d17a7622eda90ab684f3
<a href="#">RStudio 0.98.1049 - Fedora 13+/openSUSE 11.4+ (64-bit)</a>	57.9 MB	2014-09-02	52d6f2ea27384f4076412fcddb2538fb

Abbildung 2: Der Downloadbereich der RStudio Homepage

Als erstes wird **R** installiert. Der Installationspfad kann beliebig gewählt werden. Shortcuts sind nicht nötig. Nachdem die Installation von **R** erfolgreich war, installieren Sie **RStudio**. Hier ist ein Shortcut auf dem Desktop oder im Startmenü sinnvoll.

## 1.3 Die Benutzeroberfläche

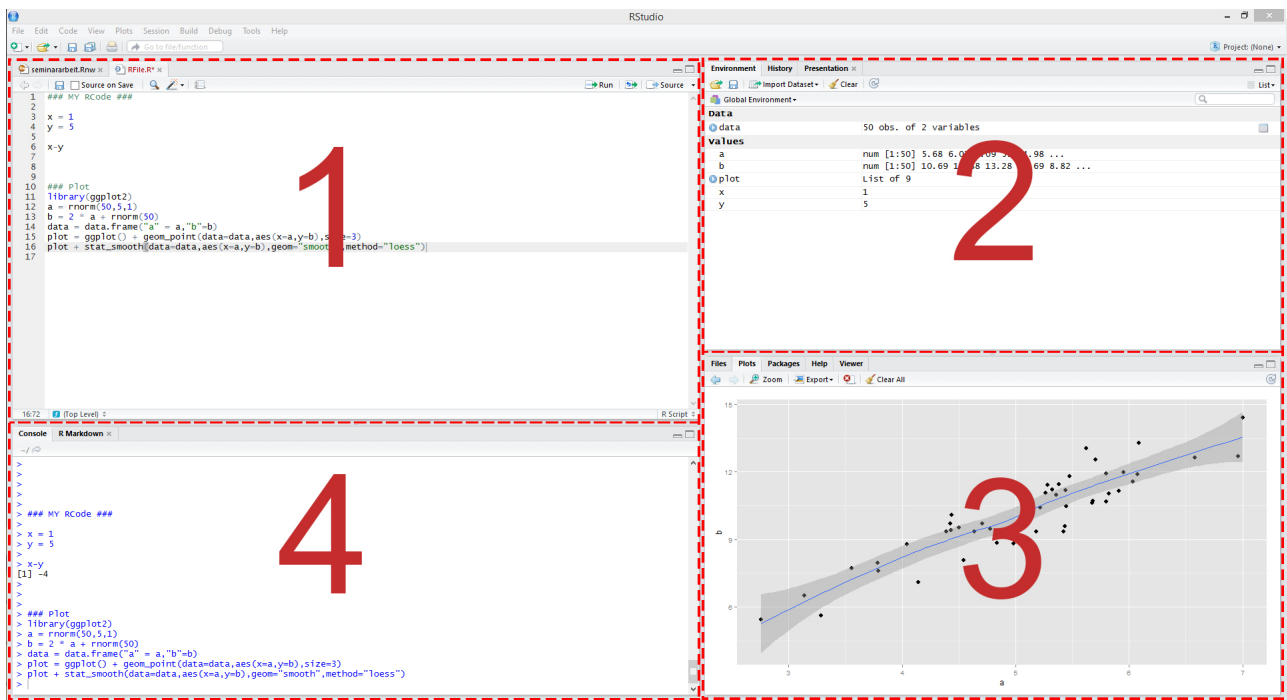


Abbildung 3: Die Benutzeroberfläche von RStudio

Starten Sie **RStudio**. Die Benutzeroberfläche ist in vier Bereiche eingeteilt:

### 1. Der Editor

Erfüllt die gleichen Funktionen wie das Programm *Editor* unter Windows.

### 2. Protokoll-Fenster

Im Reiter *Environment* findet man eine Übersicht aller gespeicherten Objekte (Variablen und Funktionen). Unter *History* wird der Verlauf des bisher ausgeführten Codes angezeigt (quasi wie die Chronik im Browser). *Presentation* ist für unsere Zwecke unwichtig. Es sei jedoch erwähnt, dass es möglich ist, Slides mit Codes und Plots direkt in **RStudio** zu erstellen und anzuzeigen.

### 3. Grafische Ausgabe, Dokumentation und Packages

Der Reiter *Plots* dient der grafischen Ausgabe. Benötigt man Hilfe zu einer bestimmten Funktion, kann man unter *Help* im Suchfeld oben rechts nach ihr suchen. Eine Übersicht aller installierten Pakete findet sich unter dem Reiter *Packages*.

### 4. Die Konsole

Wollen wir unseren im Editor geschriebenen Code ausführen, senden wir ihn über den Button `Source` (oben rechts im Editorfenster) zur Konsole. Alternativ kann man seinen Code auch direkt in der Konsole eingeben.

## 1.4 Die Dokumentation: wie war das noch gleich?

**R** verfügt über eine sehr gut strukturierte Dokumentation, mit der sich die Mehrzahl der Probleme während des Arbeitens lösen lässt. Benötigen wir Hilfe zu einer bestimmten Funktion, können wir die dazugehörige Seite mittels `help(Funktionsname)` aufrufen. Dabei folgen alle Funktionsbeschreibungen der Dokumentation demselben Aufbau. Im Abschnitt *Description* erfahren wir, welchen Zweck die jeweilige Funktion erfüllt. Unter *Usage* sehen wir den typischen Aufbau der Funktion und unter *Arguments* die detaillierte Beschreibung der Funktionsargumente. Am Ende jeder Seite findet man simple Anwendungsbeispiele. Eine Kurzschreibweise ist durch `?NameDerFunktion` gegeben.

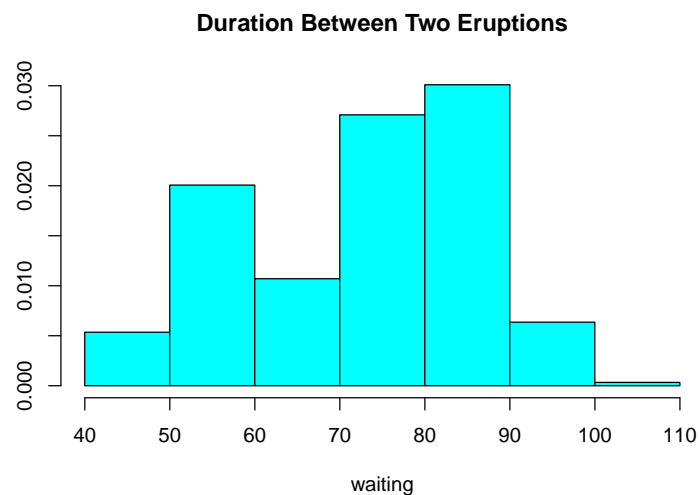
```
help(rnorm)      ## Hilfe zu der Funktion rnorm()
?help           ## Hilfe zu der Funktion help()
??randomnumbers ## Hilfe zum Stichwort "randomnumbers"
```

Die Dokumentation kann auch nach einzelnen Stichwörtern durchsucht werden. Dazu verwendet man einfach zwei Fragezeichen anstelle von einem.

## 1.5 Installation und Verwendung zusätzlicher Pakete

Neue Pakete können innerhalb von RStudio über den Reiter *Packages* oder alternativ über das Menü `Tools` `Install Packages` installiert werden. Die Standardquelle für neue Pakete ist dabei die "CRAN" Bibliothek. Fortgeschrittene Nutzer können Pakete auch direkt in der Konsole installieren. Dazu nutzt man den Befehl `install.packages()`. Möchte man ein neu installiertes Paket verwenden, so muss man es in der aktuellen **R**-Session noch importieren. Dies geschieht mit Hilfe der Funktion `library(Paketname)`. Beispielsweise beinhaltet das Paket "MASS" unter anderem einen Datensatz namens `geyser` und die Funktion `truehist()`, mit deren Hilfe das Plotten von Histogrammen verbessert werden kann.

```
# Methoden und Daten des Pakets MASS laden/importieren!
# Die auftretende Warnung kann ignoriert werden.
library(MASS)
# Variablen des Datensatzes importieren
attach(geyser)
# Histogramm der durchschnittlichen Wartezeit zwischen
# zwei Ausbrüchen des Geysirs plotten.
truehist(waiting, main="Duration Between Two Eruptions")
```



## 2 Stock and Watson - Chapter 2

## 3 Stock and Watson - Chapter 3

## 4 Stock and Watson - Chapter 4

```
library(AER) # contains the dataset
```

```
## Loading required package: car
## Warning: package 'car' was built under R version 3.2.4
## Loading required package: lmtest
## Loading required package: zoo
## Warning: package 'zoo' was built under R version 3.2.5
##
## Attaching package: 'zoo'
## Die folgenden Objekte sind maskiert von 'package:base':
##
##   as.Date, as.Date.numeric
## Loading required package: sandwich
## Loading required package: survival
## Warning: package 'survival' was built under R version 3.2.5
```

```
data(CASchools)

CASchools$tsratio <- CASchools$students/CASchools$teachers # teacher-student-ratio
CASchools$score   <- (CASchools$read + CASchools$math)/2 # average test-score

mean(CASchools$tsratio)
```

```
## [1] 19.64043
```

```
mean(CASchools$score)
```

```
## [1] 654.1565
```

```
sd(CASchools$tsratio)
```

```
## [1] 1.891812
```

```
sd(CASchools$score)
```

```
## [1] 19.05335
```

```
quantiles      <- c(0.10, 0.25, 0.4, 0.5, 0.6, 0.75, 0.9)
quantile(CASchools$tsratio, quantiles)
```

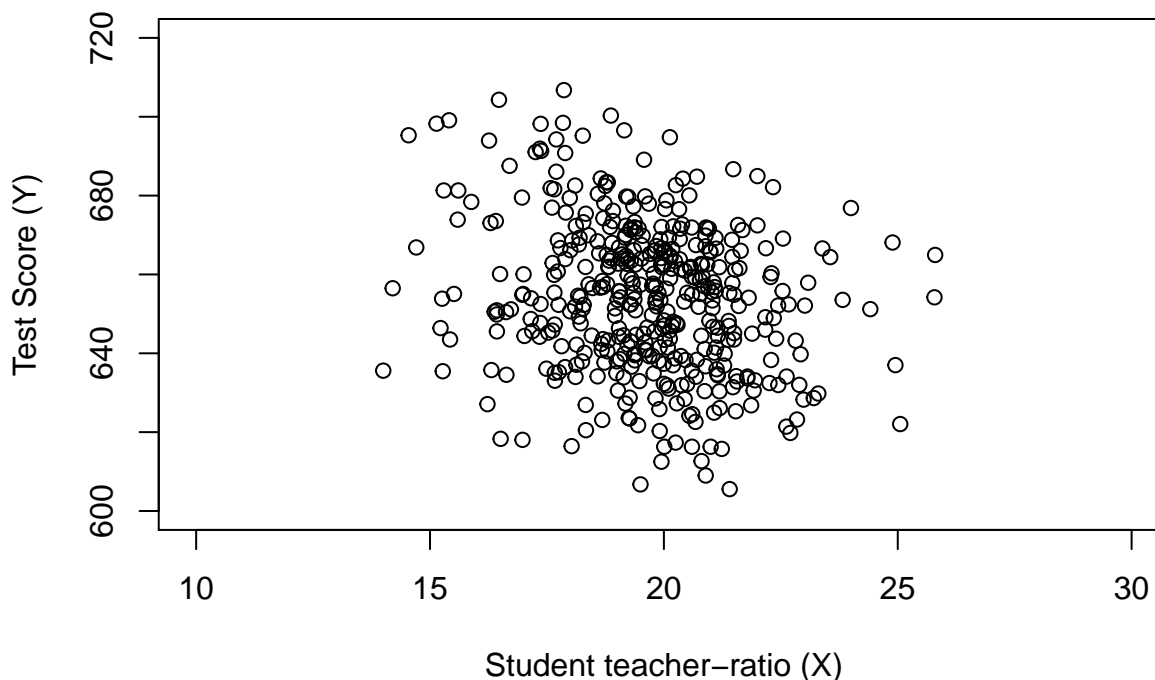
```
##      10%      25%      40%      50%      60%      75%      90%
## 17.34860 18.58236 19.26618 19.72321 20.07830 20.87181 21.86741
```

```
quantile(CASchools$score, quantiles)
```

```
##      10%      25%      40%      50%      60%      75%      90%
## 630.3950 640.0500 649.0700 654.4500 659.4000 666.6625 678.8600
```

```
plot(score ~ tsratio,
     data = CASchools,
     main = "Scatterplot of Test Score vs. Student-Teacher Ratio",
     xlab = "Student teacher-ratio (X)",
     ylab = "Test Score (Y)",
     xlim = c(10,30),
     ylim = c(600, 720))
```

## Scatterplot of Test Score vs. Student-Teacher Ratio



```
attach(CASchools)
```

```
beta_1 <- sum((tsratio - mean(tsratio))*(score - mean(score))) / sum((tsratio - mean(tsratio))^2)
beta_0 <- mean(score) - beta_1 * mean(tsratio)
```

```
library(xtable)
```

```
plot(score ~ tsratio,
     data = CASchools,
```



```

main = "Scatterplot of Test Score vs. Student-Teacher Ratio",
xlab = "Student teacher-ratio (X)",
ylab = "Test Score (Y)",
xlim = c(10,30),
ylim = c(600, 720))

```

```

linear_model <- lm(score ~ tsratio, data = CASchools)
linear_model

```

```

##
## Call:
## lm(formula = score ~ tsratio, data = CASchools)
##
## Coefficients:
## (Intercept)      tsratio
##      698.93       -2.28

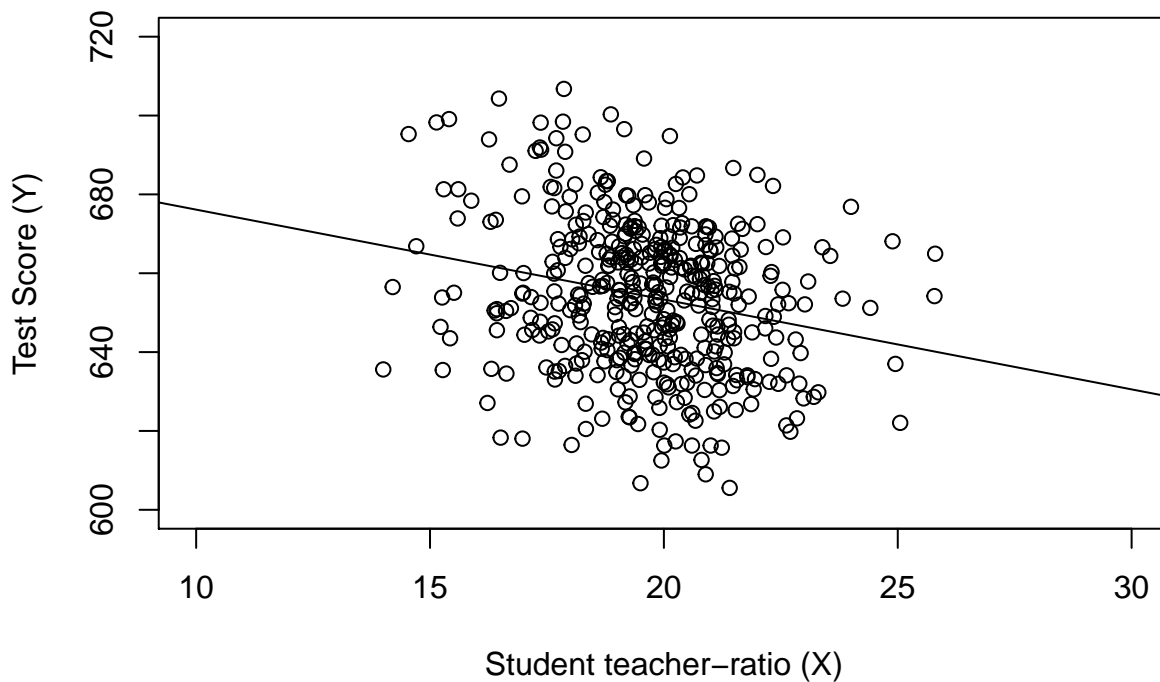
```

```

abline(a = beta_0, b = beta_1)

```

### Scatterplot of Test Score vs. Student-Teacher Ratio



```

linear_model <- lm(score ~ tsratio, data = CASchools)
linear_model

```

```

##
## Call:
## lm(formula = score ~ tsratio, data = CASchools)
##
## Coefficients:
## (Intercept)      tsratio
##      698.93       -2.28

```

```

library(xtable)
plot(score ~ tsratio,
      data = CASchools,
      main = "Scatterplot of Test Score vs. Student-Teacher Ratio",
      xlab = "Student teacher-ratio (X)",

```

```
ylab = "Test Score (Y)",  
xlim = c(10,30),  
ylim = c(600, 720))
```

```
abline(linear_model)
```

### Scatterplot of Test Score vs. Student-Teacher Ratio

