



Bachelor Wirtschaftsingenieurwesen

R Tutorium

Amelie Schischke

UNIA Universität
Augsburg
University

MRM
Materials Resource
Management

wing
Wirtschaftsingenieur

Agenda

1 Allgemeines

2 Packages

3 Basics

4 Datenimport und Datenexport

5 Deskriptive Statistik

6 Plots

7 Regressionsanalyse

8 Weiterführende Literatur

Agenda

1 Allgemeines

2 Packages

3 Basics

4 Datenimport und Datenexport

5 Deskriptive Statistik

6 Plots

7 Regressionsanalyse

8 Weiterführende Literatur

Allgemeines

Was ist R?

- Kostenfreie Open-Source-Software
- Für statistische Probleme, insbesondere zu Zeitreihen- und Regressionsanalysen
- Packages für verschiedene Anwendungen/Problemstellungen ((nicht-)lineare Modellierung, statistische Tests, Zeitreihenanalyse, Klassifikation, Clustering,...)
- Ständige Weiterentwicklung durch Community
- Keine offiziellen Handbücher/Support
- ABER: Vignetten/Paper/Dokumentationen der einzelnen Packages sowie Tutorials/Foren

Allgemeines

Was ist R?

- Übersicht unter Informationen zum Projekt R unter: <https://www.r-project.org/>
- Downloadlink (unbedingt Version beachten): <https://ftp.fau.de/cran/>
- R ist ein ausführbares Programm und kann in dieser Form bereits verwendet werden.
- ABER: es gibt graphische Oberflächen, die den Umgang mit R deutlich erleichtern:
 - Download von RStudio unter: <https://www.rstudio.com/products/rstudio/download/#download>
- ACHTUNG: Unbedingt zuerst R installieren, danach erst RStudio!
- RStudio ist nur die Oberfläche (GUI), die die Sprache R als Basis zwingend benötigt.

Download and Install R

Precompiled binary distributions of the base system and

- [Download R for Linux](#)
- [Download R for \(Mac\) OS X](#)
- [Download R for Windows](#)

R is part of many Linux distributions, you should check

Installers for Supported Platforms

Installers	Size
RStudio 1.1.423 - Windows Vista/7/8/10	85.8 MB
RStudio 1.1.423 - Mac OS X 10.6+ (64-bit)	74.5 MB
RStudio 1.1.423 - Ubuntu 12.04-15.10/Debian 8 (32-bit)	89.3 MB
RStudio 1.1.423 - Ubuntu 12.04-15.10/Debian 8 (64-bit)	97.4 MB
RStudio 1.1.423 - Ubuntu 16.04+/Debian 9+ (64-bit)	65 MB
RStudio 1.1.423 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (32-bit)	88.1 MB
RStudio 1.1.423 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (64-bit)	90.6 MB

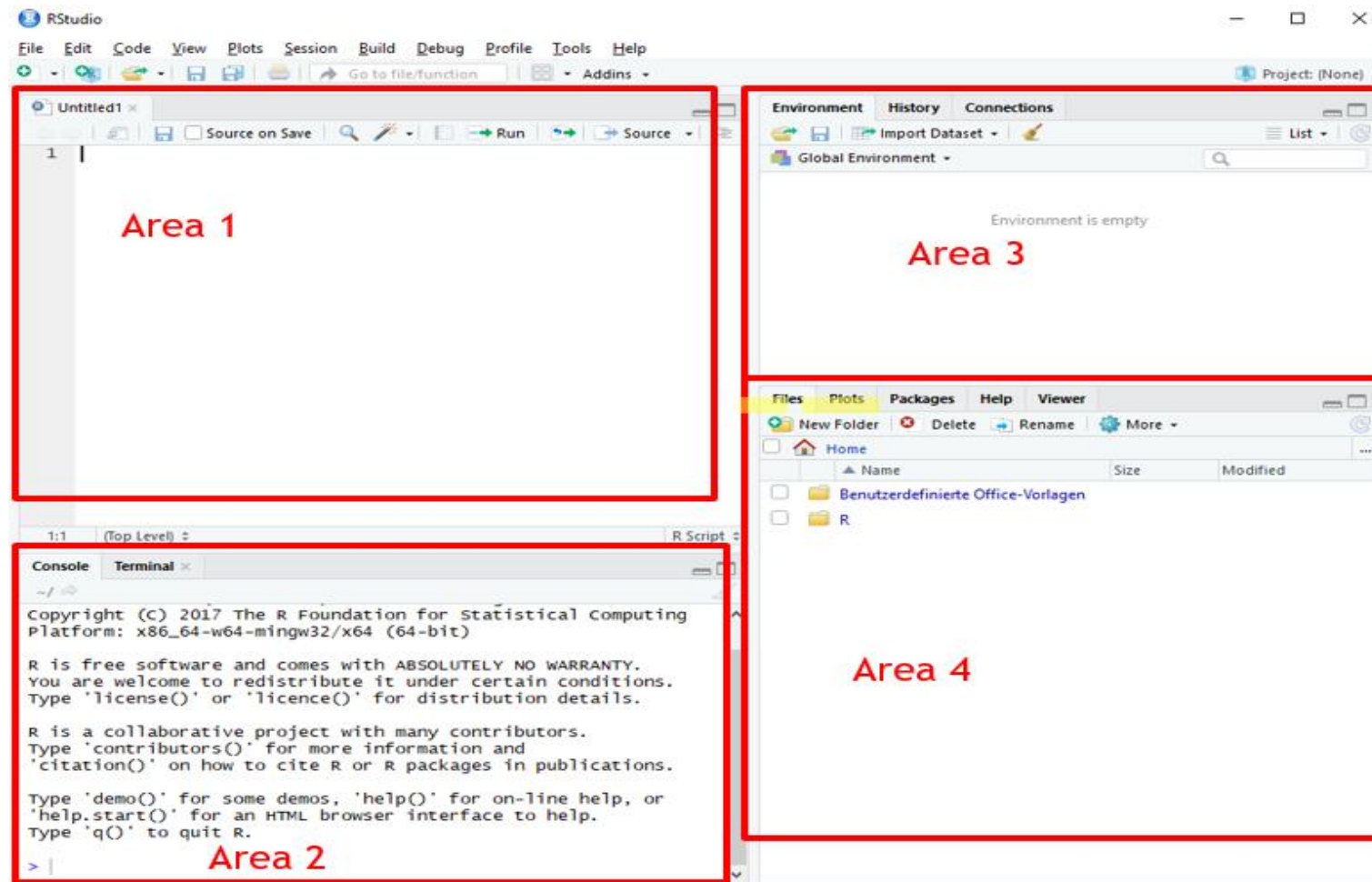
Allgemeines

Wie funktioniert R?

- Einen ersten Einblick zur Logik von Programmierung gibt folgendes Zitat:
 - *„Alles, was existiert, ist ein Objekt.
Alles, was passiert, ist ein Funktionsaufruf.“*
- John M. Chambers
- Erster Schritt nach Programmstart:
 - File → New File → R Script (alternativ Ctrl+Shift+N).
 - Im Folgenden ist eine genauere Erklärung der 4 Teilbereiche der Oberfläche von RStudio.

Allgemeines

Die Oberfläche in RStudio



Allgemeines

Die Oberfläche in Rstudio – Area 1 und Area 2

- Area 1 – Skript:
 - Dieser Bereich ist das „Hauptarbeitsfeld“, in dem der Code eines Programms verfasst wird.
 - Wird ein Skript abgespeichert, umfasst dies nur diesen Bereich.
 - Um einen Teil bzw. den gesamten Code auszuführen, Bereich markieren und mit „Ctrl+Enter“ starten.
- Area 2 – Konsole:
 - Hier wird der Code angezeigt, der tatsächlich in R berechnet wird.
 - Bsp.: Beim Import von Daten sieht man in der Konsole Befehle, die in RStudio durch „Klicken“ aufgerufen werden.
 - Kommandos können hier manuell eingegeben werden, diese werden aber nur einmalig ausgeführt und können nicht abgespeichert werden.

Allgemeines

Die Oberfläche in Rstudio – Area 3 und Area 4

- Area 3 – Datenbereich von RStudio:
 - Geringfügig abweichende Struktur zwischen Mac/Windows (voll kompatibel).
 - Zeigt alle Objekte, die derzeit verfügbar sind und verwendet werden können.
 - Separat speicherbar im Format .Rdata.
 - ABER: Datensätze im Regelfall NICHT speichern, sondern den Code (Area 1) zur Rekonstruktion der Ergebnisse speichern (Nachvollziehbarkeit).
 - Import von Excel Files hier per Klick möglich.
- Area 4 – Zielverzeichnis:
 - Plots können angesehen und ggf. direkt exportiert/gespeichert werden.
 - Packages können heruntergeladen, installiert und aktiviert werden.
 - Hilfedateien/Erklärungen hier einsehbar (auch Hilfedateien der Packages).

Agenda

1 Allgemeines

2 Packages

3 Basics

4 Datenimport und Datenexport

5 Deskriptive Statistik

6 Plots

7 Regressionsanalyse

8 Weiterführende Literatur

Packages

Übersicht (I)

- Ein Package ist eine vordefinierte Sammlung von Methoden, die von Mitgliedern der Community erstellt wurden.
- Für viele Standardanwendungen sind Packages verfügbar, welche meist vor dem Release geprüft werden → Die Funktionen eines Package sollten aber immer selbst überprüft werden!
- Packages können mittels Button in RStudio oder im Code installiert werden:
 - `install.packages("Name Package")`
- Packages müssen vor Verwendung geladen werden:
 - `library("Name Package")`
- Für wissenschaftliche Arbeiten ist die Methode im Code wg. Nachvollziehbarkeit immer zu präferieren.
- Vorgehen: Package installieren → Package laden → Package verwenden
- Achtung: Package muss nach jedem Neustart von Rstudio neu geladen, aber nicht neu installiert werden!

Packages

Übersicht (II)

- Zu einigen Packages sind sog. „CheatSheets“ verfügbar, welche die wichtigsten Funktionen und Kommandos auf einen Blick darstellen.
- Die Dokumentation eines Package gibt einen Überblick über die bereitgestellten Funktionen, insbesondere zu ihren notwendigen Inputvariablen, Output, Details zur Berechnung sowie Beispiele zur Verwendung.
- Vignetten beschreiben die Berechnung der implementierten Funktionen, genauere Hintergründe sowie die Anwendung der zur Verfügung gestellten Funktionen.
- Nützliche Packages:
 - ggplot2 Erstellung von Plots
 - dplyr Datenaufbereitung
 - e1071 Berechnung von Skewness/Kurtosis
 - readxl Importieren und Exportieren von Excel-Dateien in/aus R

Agenda

1 Allgemeines

2 Packages

3 **Basics**

4 Datenimport und Datenexport

5 Deskriptive Statistik

6 Plots

7 Regressionsanalyse

8 Weiterführende Literatur

Basics

Kommentar

- Kommentare dienen der Übersichtlichkeit und zum besseren Verständnis eines Codes.
- Kommentare helfen einem selbst, den Überblick im Code zu behalten, aber auch anderen, den Code nachvollziehen zu können.
- Mit „#“ startet ein Kommentar in R.
- Alles, was hinter „#“ steht, wird bei Ausführung des Codes NICHT mit ausgeführt und insbesondere führt es nicht zu Fehlern.
- Im Allgemeinen sollte ein Code immer (ausführlich) kommentiert werden.
- Nach gängigen Code conventions sollten Kommentare auf Englisch verfasst werden.

Basics

Variablen & Datentypen

- Name der Variable muss mit einem Buchstaben beginnen (Code convention: Kleinbuchstaben).
- Datentyp (= Art einer Variablen) wird im Allgemeinen durch zugewiesene Daten bestimmt.
- Das Auswerten eines Begriffs ohne Zuweisung gibt den Wert auf der Konsole aus und speichert diesen nicht ab.

Datentyp	Beschreibung
Numeric (skalar)	Kommazahl
Vector	Durchnummerierte Kette von Objekten eines Datentyps
Matrix	Kette von Elementen eines Datentyps in 2 Dimensionen
Logical (boolean)	Wahr oder Falsch
Character (String)	Wörter/Zeichenketten
Date	Datum
Dataframe	Tabellen, bei der jede Spalte von einem Datentyp ist.
List	Durchnummerierte Kette von Objekten verschiedener Datentypen

Basics

Datentypen – Character & Skalar

- Character

- `x <- "Hello World!"` # assign a string to a variable x

- Skalar

- `x <- 10` # assign a scalar to a variable x
wichtig:
bei Neuordnung eines Werts zur Variable x
wird der vorherige Wert überschrieben
insbesondere ändert sich hier der Datentyp

Basics

Datentypen – Vektor

- Vektor aus mehreren Elementen

- `x <- c(1, 2, 3, 4, 5)` # c ist die Kurzform für concatenate
- `x <- seq(1, 10, 2)` # Sequenz von 1 bis 10, in Schrittweite 2
- `x <- 1 : 10` # Kurzform für Sequenz der Schrittweite 1

- Selektion von Werten in Vektoren

- `x[1]` # Wert des Vektors x an der Stelle 1
Indizes starten bei 1 in R
das Element x[0] existiert nicht!
- `x <- x[-1]` # Vektor x wird überschrieben, das Element an
Stelle 1 dabei ignoriert
- `x[x < 3]` # Alle Elemente von x, deren Wert kleiner 3 ist

Basics

Datentypen – Matrix

- Erstellen einer Matrix

- `d <- c(1 : 10)` # Sequenz von 1 bis 10
- `x <- matrix(data = d, nrow = 5, ncol = 2, byrow = TRUE)`
Erstellt eine Matrix aus den Daten d, welche 5 Zeilen und 2 Spalten hat
- `View(x)` # Öffne x zur Ansicht
- `x` # print x in console

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

Basics

Datentypen – Dataframe (I)

- Liste von Vektoren gleicher Länge.
- Verschiedene Spalten können verschiedene Datentypen besitzen.
- Erstellen eines data.frame
 - `x_d <- as.data.frame(x)`
Erstellt einen data.frame aus der Matrix x (vorherige Folie)
- Benennung der Spalten des data.frame
 - `colnames(x_d) <- c("Spalte1", "Spalte2")`
- Ansprechen von Werten eines data.frame
 - `x_d[,1]` # alle Werte der ersten Spalte
 - `x_d$Spalte2` # alle Werte der Spalte 2
 - `x_d$Spalte1[1 : 3]` # Werte der ersten drei Zeilen von Spalte 1

Basics

Datentypen – Dataframe (II)

- Nützliche Befehle für data.frame
 - `x_d$Spalte3 <- x_d$Spalte1 + 2 * x_d$Spalte2`
Hinzufügen einer neuen Spalte „Spalte3“
 - `x_d$Spalte2 <- NULL` # Löschen von Spalte2
 - `x_d[x_d$Spalte3 > 5,]` # Filtern der Zeilen
 - `x_d[order(x_d$Spalte3),]` # Sortieren der Zeilen nach Spalte2

Basics

Bedingungen – Übersicht

- Normalerweise wird Code zeilenweise ausgeführt. Es kann jedoch sinnvoll sein, Code Blöcke nur dann auszuführen, wenn eine bestimmte Bedingung erfüllt ist.
- Schema: „Wenn-Dann-Ansonsten“
- Syntax:
 - ```
if(condition) {
 # Code, falls condition wahr / erfüllt ist
}
```
  - ```
if(condition) {  
    # Code, falls condition wahr / erfüllt ist  
}else if(condition2){  
    # Code, falls condition2 wahr / erfüllt ist  
}else{  
    # Code, falls keine der oberen Bedingungen wahr / erfüllt ist  
}
```

Basics

Bedingungen – Beispiele

- Falls das Klausurergebnis `examResult` besser 4.0, gib "Prüfung bestanden" aus
 - `examResult <- 1.0`
 - ```
if(examResult <= 4.0){
 print("Prüfung bestanden")
}
```
- Wenn die Klausur mit 1.0 bestanden ist, beglückwünsche die Kandidatin, wenn sie bestanden wurde, teile ihr ihre Note mit und ansonsten teile der Kandidatin mit, dass sie in die Nachklausur muss:
  - ```
if(examResult == 1.0){  
    print(„Glückwunsch zur 1.0!“)  
}else if(examResult <= 4.0){  
    print(examResult)  
}else{  
    print(„Leider musst du in die Nachklausur.“)  
}
```

Basics

Schleifen – Übersicht

- Schleifen ermöglichen es, Code wiederholt auszuführen.
- Schleifen sind Kontrollstrukturen (siehe if und else), die einen Codeblock wiederholen, solange eine Bedingung eingehalten wird.
- Vorgehen:
 - Überprüfe, ob die gegebene Bedingung eintritt.
 - Einmalige Ausführung des Codeblocks.
 - Erneute Überprüfung der Bedingung.
 - Dies wird solange wiederholt, bis die Bedingung nicht mehr zutrifft.
(Ausnahme: bei Programmierfehler kann es zu einer Endlosschleife kommen)
- Faustregel:
 - Sobald man Code mehr als 2 mal untereinander kopiert, um inhaltlich das Selbe auf unterschiedliche Objekte anzuwenden, kann eine Schleife sinnvoll sein und (Kopier-)Fehler minimieren.

Basics

Schleifen – for Schleife (I)

- Ziel: einen Code $n \in \mathbb{N}$ mal auszuführen
- Syntax:
 - # Die Schleife geht über die Zahlen von startwert bis endwert
i.A. kann auch rückwärts gezählt werden, wenn startwert > endwert
`for(nCount in startwert : endwert){`
 - # in die geschweiften Klammern schreibt man nun Code,
 - # der wiederholt ausgeführt wird`}`

Basics

Schleifen – for Schleife (II)

- Beispiel:

- ```
x <- c(1, 3, 5, 10, 15) # Initialisierung der Vektoren x, z
z <- c(2, 7, 8, 12, 17)
```
- ```
for(nExample in 1 : 5){
  print(nExample)
  z[nExample] <- z[nExample] + x[nExample]
  # Addiere zu jedem Vektorelement von z das Vektorelement von x
}
```
- ```
print(z)
[1] 3 10 13 22 32 # Ausgabe von z in der Konsole
```

# Basics

---

## Schleifen – while Schleife

- Ziel: einen Code so lange auszuführen, wie eine bestimmte Bedingung erfüllt ist.
- Syntax:
  - # Die Schleife wird solange ausgeführt, wie die Bedingung wahr ist  
`while(Bedingung) {`  
    # in die geschweiften Klammern schreibt man nun Code  
`}`
- Beispiel:
  - `nCount <- 1`  
`while(nCount < 5) {`  
    `z[nCount] <- z[nCount] + x[nCount]`  
    `nCount <- nCount + 1` # die Zählvariable muss manuell verändert werden  
`}`
- Meist sind while Schleifen für Bedingungen sinnvoller, for Schleifen, wenn etwas  $n \in \mathbb{N}$  mal ausgeführt werden soll.

# Basics

---

## Funktionen – Übersicht

- Eine Funktion fasst Codeabschnitte zusammen, welcher nur mit Aufruf der Funktion ausgeführt wird.
- R hat viele Funktionen bereits vorimplementiert und Packages bieten weitere Funktionen an.
- Beispiel zur Berechnung des Mittelwerts mit Hilfe der Funktion mean
  - `x <- c(1, 2, 3, 4, 5)`
  - `mean(x, na.rm = TRUE)` # Objekt x als notwendiger Inputparameter  
[1] 3 # weitere mögliche Inputparameter sind bspw. na.rm  
# hierbei werden mögliche NA Werte entfernt

# Basics

## Funktionen – eigene Funktionen (I)

- Zusätzlich zu den bereitgestellten Funktionen, kann man selbst Funktionen schreiben.
- Syntax:
  - # mit dem Schlüsselwort `function` sagen wir, dass `myFunction` eine Funktion sein soll  
# in den runden Klammern geben wir Inputparameter `input1, ..., inputN` ein  

```
myFunction <- function(input1, input2, ..., inputN){
 # Code der innerhalb der Funktion ausgeführt werden soll
 return(output) # die Funktion gibt output zurück
}
```
- Beispiele:
  - ```
greetings <- function(name){  
  print(paste("Hallo ", name, sep = ""))  
}  
greetings("Patric")      # Aufruf der Funktion mit Inputwert "Patric"  
[1] "Hallo Patric"
```

Basics

Funktionen – eigene Funktionen (II)

- Beispiele:
 - Berechnung des Kapitalwerts eines Zahlungsstroms cashFlow (Vektor) mit Kalkulationszinssatz discountRate
 - ```
calculatePresentValue <- function(cashFlow, discountRate){
 pv <- 0
 for(t in 1 : length(cashFlow)){
 pv <- pv + cashFlow[t] / (1 + discountRate)^t
 }
 return(pv)
}
```
  - Aufruf der Funktion für Zahlungsstrom cashFlow1 und Kalkulationszinssatz discountRate1
  - ```
cashFlow1 <- c(100, 120, 160, 130, 145, 95)  
discountRate1 <- 0.05  
calculatePresentValue(cashFlow1, discountRate1)  
[1] 633.7487
```

Agenda

1 Allgemeines

2 Packages

3 Basics

4 **Datenimport und Datenexport**

5 Deskriptive Statistik

6 Plots

7 Regressionsanalyse

8 Weiterführende Literatur

Datenimport & Datenexport

Datenimport

- Import einer .csv Datei
 - `data <- read.csv("data.csv", header = TRUE, sep = ",")`
- Import einer .xlsx Datei
 - `library("readxl")` # Laden des Packages readxl
 - `data <- read_excel("C:/Users/papenfpa/Documents/data.xlsx")`
- Ansicht der eingelesenen Daten
 - `View(data)`
- Anstatt den Pfad im Befehl „read_excel“ direkt anzugeben, kann er auch vorab gesetzt werden:
 - `getwd()`
 - `setwd("C:/Users/papenfpa/Documents/")`
 - `data <- read_excel("data.xlsx")`

Datenimport & Datenexport

Datenexport

- Speichern von Objekten als Excel Files
 - `write.xlsx(data, file = "data_excel.xlsx", sheetName = "HelloWorld", col.names = TRUE, row.names = FALSE, showNA = TRUE)`
- Achtung: R überschreibt vorhandene Excel-Dateien, ohne vorherige Abfrage!
- Export als .csv Datei ebenfalls möglich
 - `write.csv(data, file = "data_Excel.xlsx", row.names = FALSE)`
- Plots können direkt mit R exportiert werden
 - `png(filename = „plot_data.png“)`
`plot()`
`dev.off()`
- Plots, die über das Package ggplot2 erstellt wurden, können mit dem Befehl ggsave gespeichert werden.
- Alle Dateien werden, falls nicht anders spezifiziert, im working directory abgelegt.

Agenda

1 Allgemeines

2 Packages

3 Basics

4 Datenimport und Datenexport

5 **Deskriptive Statistik**

6 Plots

7 Regressionsanalyse

8 Weiterführende Literatur

Deskriptive Statistik

Überblick über die Daten

- Nützliche Befehle

- `str(data)` # Struktur des Datensets
- `names(data)` # Spaltennamen
- `head(data)` # Ausgabe der obersten Zeilen
- `data[10 : 20,]` # Ausgabe der Zeilen 10 bis 20
- `nrow(data)` # Zeilenanzahl
- `ncol(data)` # Spaltenanzahl
- `dim(data)` # Dimensionen (Zeilenanzahl und Spaltenanzahl)

Deskriptive Statistik

Berechnung von Kenngrößen

- Lagemaße, Streuungsmaße, Korrelationen

- `mean(data)` # Berechnung des Erwartungswerts $\left(\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i\right)$
- `median(data)` # Berechnung des Medians
- `var(data)` # Berechnung der Varianz $\left(\sigma^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2\right)$
- `sd(data)` # Berechnung der Standardabweichung
- `quantile(data, q)` # Berechnung des q-ten Quantils
- `summary(data)` # Summary des Datensatzes
- `cov(data, data2)` # Berechnung der (Stichproben-) Kovarianz zwischen
den Datensätzen data, data2 $\left(\text{Cov}(X, Y) = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{n-1}\right)$
- `cor(data, data2)` # Berechnung der Korrelation $\left(\text{Corr}(X, Y) = \frac{\text{Cov}(X, Y)}{\sigma_x \cdot \sigma_y}\right)$

Agenda

1 Allgemeines

2 Packages

3 Basics

4 Datenimport und Datenexport

5 Deskriptive Statistik

6 **Plots**

7 Regressionsanalyse

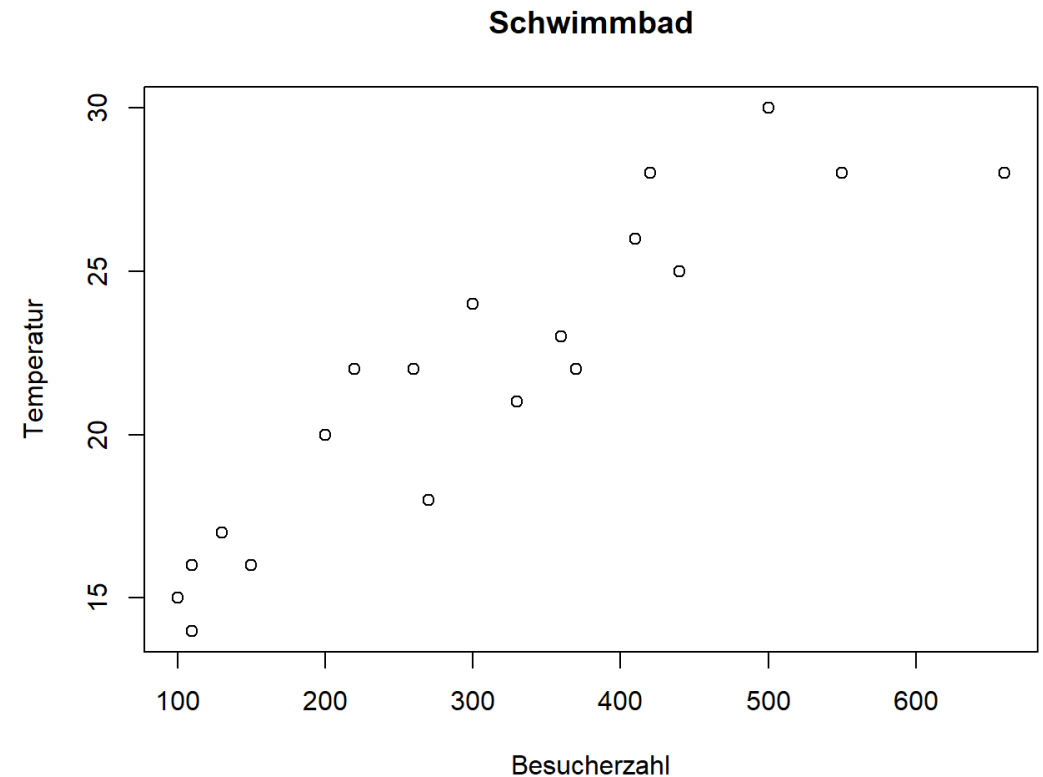
8 Weiterführende Literatur

Plots

Scatterplot

- Plotten zweier Variablen gegeneinander
 - ```
plot(data$Besucheranzahl,
 data$Temperatur,
 col = "black",
 main = "Titel",
 ylab = "y-Beschriftung",
 xlab = "x-Beschriftung")

Scatterplot der Variable size
des Dataframes data
zur Variable size2
des Dataframes data2
```
  - ```
abline(intercept, slope, col =  
"red")  
# Hinzufügen einer Linie mit  
# Achsenabschnitt intercept und  
# Steigung slope
```



Plots

Linienplot

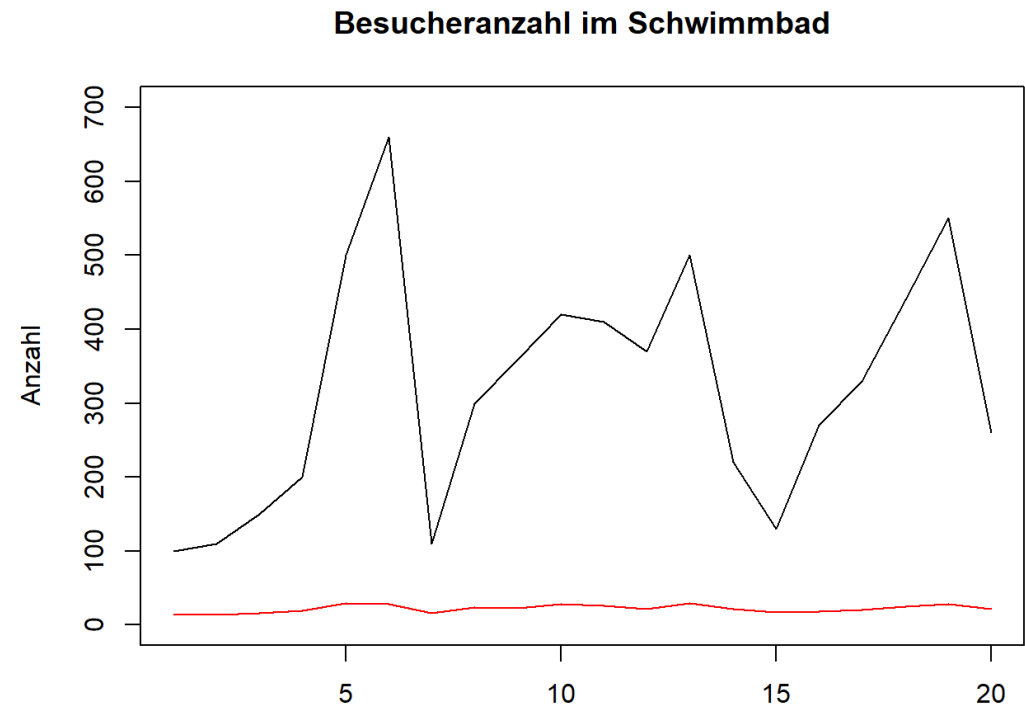
- Plotten zweier Variablen gegeneinander

- ```
plot(data$Besucheranzahl,
 type = "l",
 xlab = "x-Beschriftung",
 ylab = "y-Beschriftung",
 main = "Titel",
 ylim = c(1,700))
```

```
Linienplot der Variable
Besucheranzahl
des Dataframes data
Bestimme das Intervall [1,700]
für die y-Achse
```

- ```
lines(data$Temperatur, type = "l",  
      col = "red")
```

```
# Hinzufügen einer Linie für die  
# Variable Temperatur
```

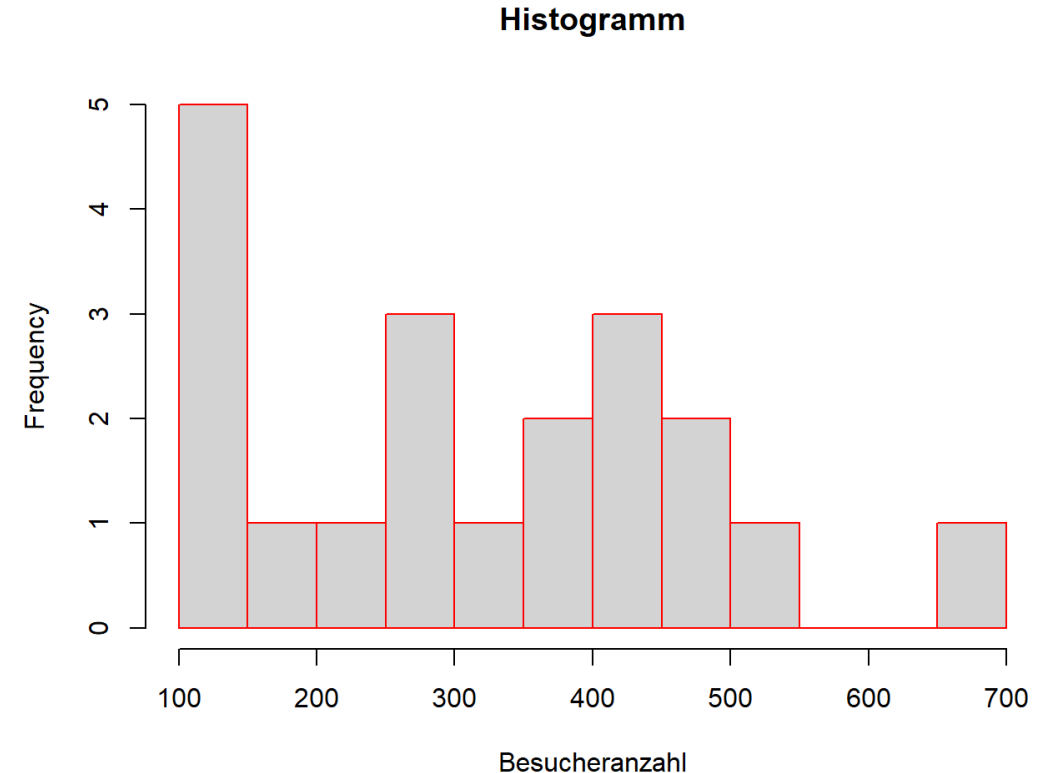


Plots

Histogramm

- Darstellung der Häufigkeitsverteilung
 - ```
hist(data$Besucheranzahl,
 breaks = 12,
 plot = TRUE,
 xlab = "x-Beschriftung",
 main = "Titel",
 freq = TRUE,
 col = "lightgrey",
 border = "red")

Histogramm für die Variable size
mit freq = FALSE bekommt man die
Wahrscheinlichkeitsverteilung
anstelle der Häufigkeiten
```



# Agenda

---

**1** Allgemeines

**2** Packages

**3** Basics

**4** Datenimport und Datenexport

**5** Deskriptive Statistik

**6** Plots

**7** **Regressionsanalyse**

**8** Weiterführende Literatur



# Regressionsanalyse

---

## Übersicht

- Ziel: Identifikation und Modellierung von Zusammenhängen zwischen abhängigen und unabhängigen Variablen.
  - Gibt es eine Beziehung zwischen den Variablen?
  - Schätzung des Einflusses der unabhängigen Variable auf die abhängige Variable.
  - Zur Prognose/Vorhersage der abhängigen Variable.
- Wichtig: bei der linearen Regression werden nur lineare Zusammenhänge zwischen den Daten modelliert!
- Beispiel:
  - Gibt es einen Zusammenhang zwischen der Tagesdurchschnittstemperatur und den Besucherzahlen im Schwimmbad?
  - Kann die Besucherzahl abgeschätzt werden, wenn die Temperatur bekannt ist?

# Regressionsanalyse

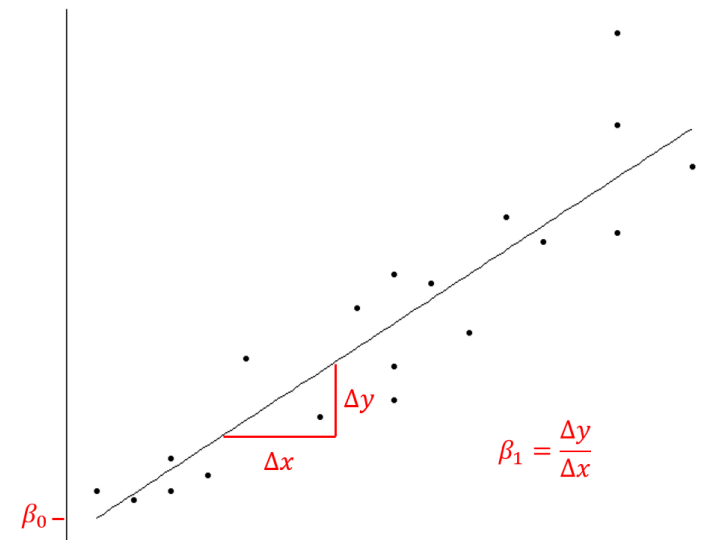
## Univariate lineare Regression (I)

- Abhängige/zu beschreibende Variable  $Y$  mit  $n \in \mathbb{N}$  Beobachtungen  $y_1, \dots, y_n$
- Unabhängige/ beschreibende Variable  $X$  mit  $n \in \mathbb{N}$  Beobachtungen  $x_1, \dots, x_n$
- Zusammenfassung der  $n \in \mathbb{N}$  Beobachtungen der beiden Variablen in Tupeln:  $(x_1, y_1), \dots, (x_n, y_n)$
- Allgemeine Regressionsgleichung:  $\eta(x) = E[Y|x]$
- Bei der linearen Regression wird ein linearer Zusammenhang zwischen der abhängigen und unabhängigen Variable angenommen:

$$Y = \beta_0 + \beta_1 X + \varepsilon$$

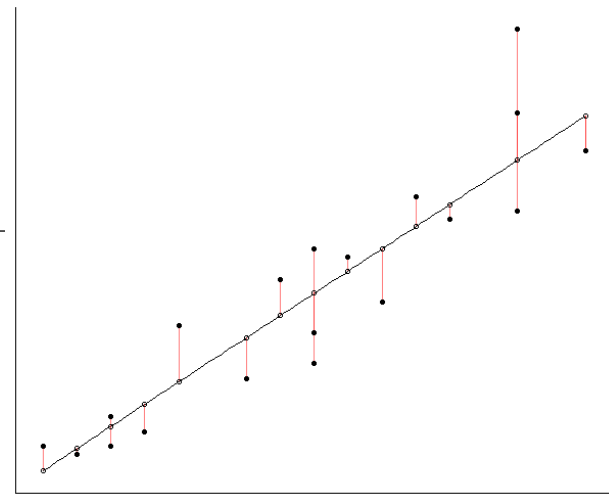
mit

- y-Achsenabschnitt  $\beta_0$
- Steigung  $\beta_1$ , Einfluss der unabhängigen Variable auf die abhängige Variable
- Fehlerterm  $\varepsilon$



# Regressionsanalyse

## Univariate lineare Regression (II)



- Lineare Regression:

$$Y = \beta_0 + \beta_1 X + \varepsilon$$

- Im Allgemeinen ist der wahre Zusammenhang zwischen den Variablen  $X$  und  $Y$  und damit die Parameter  $\beta_0$  und  $\beta_1$  unbekannt und muss mit Hilfe von Daten geschätzt werden!
- Wie müssen die Parameter  $\beta_0$  und  $\beta_1$  aussehen, sodass die Gerade am besten zu den beobachteten Werte von  $X$  und  $Y$  (s. Plot) passt und damit den beobachteten Zusammenhang am besten abbildet?

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 \hat{x}$$

⇒ Wahl der Parameter so, dass der Fehler  $\varepsilon$  (die Abweichung) minimiert wird!

- Unter bestimmten Voraussetzungen gibt der „kleinste Quadrate Schätzer“ die besten Schätzwerte für die Parameter. Es wird dabei die Summe der quadratischen Abweichungen der Datenpunkte von der Regressionsgeraden minimiert:

$$\min \sum_i^n \varepsilon_i = \min \sum_i (y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i))^2 \Rightarrow \hat{\beta}_0, \hat{\beta}_1$$

- Hinweis: R berechnet für uns den kleinsten Quadrate Schätzer über die Funktion [lm](#)

# Regressionsanalyse

## Multivariate lineare Regression

- Bisher haben wir den Zusammenhang einer unabhängigen Variable auf die abhängige Variable betrachtet.
- Allerdings können mehrere unabhängige Variablen  $X_1, \dots, X_K$  die abhängige Variable beeinflussen.  
→ Erweiterung der univariaten linearen Regression zur multivariaten linearen Regression:  
$$Y = \beta_0 + \beta_1 X_1 + \dots + \beta_K X_K + \varepsilon$$
  
→ der Parameter  $\beta_k, k = 1, \dots, K$ , beschreibt hierbei den Einfluss, den die unabhängige Variable  $X_k$  auf die abhängige Variable  $Y$  hat.
- Die Schätzung folgt auch hier der Methode der kleinsten Quadrate Schätzung (Minimierung der quadratischen Abweichung) und wir erhalten die „besten“ Parameterschätzer  $\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_K$ .
- Beispiel (fortgesetzt):
  - Neben der Temperatur ist den BesucherInnen des Schwimmbads auch die Wassertemperatur wichtig. Beeinflusst die Wassertemperatur die Besucheranzahl?

# Regressionsanalyse

## Güte der Regression – Bestimmtheitsmaß

- Bestimmtheitsmaß  $R^2$ 
  - Kennzahl zur Beurteilung der Anpassungsgüte einer Regression
  - „Der Anteil, der durch die Regression erklärten Quadratsumme an der zu erklärenden totalen Quadratsumme“
  - Wie viel Streuung in den Daten kann durch ein vorliegendes lineares Regressionsmodell „erklärt“ werden?
- $$R^2 = \frac{\sum_i (\hat{y}_i - \bar{y})^2}{\sum_i (y_i - \bar{y})^2} = \frac{\text{Summe der Quadrate der erklärten Abweichung}}{\text{totale Quadratsumme}} \in [0,1]$$
- $R^2 = 1$ : das Modell kann die Zusammenhänge perfekt erklären.
- $R^2 = 0$ : es gibt keinen linearen Zusammenhang/ das Modell kann die Zusammenhänge nicht erklären.
- Im Allgemeinen gilt, je höher der Wert des Bestimmtheitsmaßes  $R^2$  ist, desto besser ist das Modell.
- Adjusted  $R^2$  ist ein erweitertes Maß, welches die Anzahl an Variablen  $K$  mitberücksichtigt.

# Regressionsanalyse

---

## Güte der Regression – Test der Parameter

- Die Koeffizienten/Parameter  $\beta_0, \beta_1, \dots, \beta_K$  werden auf ihre Signifikanz getestet.
  - hat der geschätzte Parameter  $\beta_k$  einen signifikant unterschiedlichen Wert von 0?
  - falls ja, weist das auf einen signifikanten, linearen Einfluss von Variable  $X_k$  auf  $Y$  hin!
- T-Test:
  - $H_0: \beta_k = 0$
  - $H_1: \beta_k \neq 0$

# Regressionsanalyse

---

## Regression in R (I)

- Für eine lineare Regression berechnet R automatisch die meisten, notwendigen Informationen.
- Beispiel im univariaten Fall:
  - `data_lm <- read_excel("C:/Users/papenfpa/Documents/data.xlsx")`
  - `# Berechnung einer linearen Regression mit Besucheranzahl als abhängige,  
# Temperatur als unabhängige Variable, die Daten sind im dataframe data_lm  
lin_model <- lm("Besucheranzahl ~ Temperatur", data = data_lm)`
  - `# Ausgabe der Parameter, Teststatistiken, Bestimmtheitsmaße etc.  
summary(lin_model)`

# Regressionsanalyse

## Regression in R (II)

- Beispiel im univariaten Fall:
  - Geschätzter Beta-Koeffizient für Temperatur von 29.254
  - Dieser Koeffizient ist signifikant auf dem 0.001 Niveau mit einer Teststatistik von 9.879 und ein p-Wert von 1.08e-08.
  - → Eine Erhöhung der Temperatur um 1 Grad führt durchschnittlich zu 29.254 mehr BesucherInnen.  
→ Die Temperatur beeinflusst die Besucheranzahl signifikant.
  - Der  $R^2$  Wert von 0.8443 ist relativ hoch.  
→ Das Modell beschreibt den Zusammenhang relativ gut.

```
> summary(lin_model)
```

```
Call:
lm(formula = "Besucheranzahl ~ Temperatur", data = data_lm)

Residuals:
 Min 1Q Median 3Q Max
-92.19 -47.71 -13.31 41.81 172.29

Coefficients:
 Estimate Std. Error t value Pr(>|t|)
(Intercept) -331.394 67.514 -4.908 0.000113 ***
Temperatur 29.254 2.961 9.879 1.08e-08 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 65.93 on 18 degrees of freedom
Multiple R-squared: 0.8443, Adjusted R-squared: 0.8356
F-statistic: 97.6 on 1 and 18 DF, p-value: 1.076e-08
```



# Regressionsanalyse

## Regression in R (III)

- Beispiel im multivariaten Fall:
  - # Einlesen der Daten y als abhängige, a, b, c als unabhängige Variablen  
`data_lm <- read_excel("C:/Users/papenfpa/Documents/data.xlsx")`
  - # Berechnung einer linearen Regression mit y als abhängige, alle anderen  
# Spalten des Dataframes data\_lm als unabhängige Variablen  
`lin_model <- lm("Besucheranzahl ~ . ", data = data_lm)`
  - # Berechnung einer linearen Regression mit y als abhängige, b, c als  
# unabhängige Variablen  
`lin_model <- lm(y ~ b + c, data = data_lm)`
  - # Ausgabe der Parameter, Teststatistiken, Bestimmtheitsmaße etc.  
`summary(lin_model)`

# Regressionsanalyse

## Regression in R (IV)

- Beispiel im multivariaten Fall:
  - Geschätzter Beta-Koeffizient für Temperatur von 34.189. Dieser Koeffizient ist signifikant auf dem 0.001 Niveau.
    - Eine Erhöhung der Temperatur um 1 Grad führt durchschnittlich zu 34.189 mehr BesucherInnen.
    - Die Temperatur beeinflusst die Besucheranzahl signifikant.
  - Geschätzter Beta-Koeffizient für Wassertemperatur von -69.162. Dieser Koeffizient ist nicht signifikant und hat einen p-Wert von 0.242.
    - Die Wassertemperatur beeinflusst die Besucheranzahl nicht signifikant!

```
> summary(lin_model)
```

```
Call:
```

```
lm(formula = "Besucheranzahl ~ .", data = data)
```

```
Residuals:
```

```
 Min 1Q Median 3Q Max
-83.426 -54.509 2.731 34.564 165.700
```

```
Coefficients:
```

```
 Estimate Std. Error t value Pr(>|t|)
(Intercept) 1100.076 1182.830 0.930 0.365
Temperatur 34.189 5.012 6.821 2.98e-06 ***
Wassertemperatur -69.162 57.058 -1.212 0.242
```

```

```

```
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 65.09 on 17 degrees of freedom
Multiple R-squared: 0.8567, Adjusted R-squared: 0.8398
F-statistic: 50.81 on 2 and 17 DF, p-value: 6.741e-08
```

# Regressionsanalyse

## Regression in R (V)

- Beispiel im multivariaten Fall:
  - Der  $R^2$  Wert von 0.8567 ist relativ hoch.
    - Das Modell beschreibt den Zusammenhang relativ gut.
    - der adjusted  $R^2$  Wert ist niedriger als im univariaten Fall
    - das univariate Modell performt besser.

```
> summary(lin_model)
```

```
Call:
lm(formula = "Besucheranzahl ~ .", data = data)
```

```
Residuals:
 Min 1Q Median 3Q Max
-83.426 -54.509 2.731 34.564 165.700
```

```
Coefficients:
 Estimate Std. Error t value Pr(>|t|)
(Intercept) 1100.076 1182.830 0.930 0.365
Temperatur 34.189 5.012 6.821 2.98e-06 ***
Wassertemperatur -69.162 57.058 -1.212 0.242
```

```

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 65.09 on 17 degrees of freedom
Multiple R-squared: 0.8567, Adjusted R-squared: 0.8398
F-statistic: 50.81 on 2 and 17 DF, p-value: 6.741e-08
```

# Agenda

---

**1** Allgemeines

**2** Packages

**3** Basics

**4** Datenimport und Datenexport

**5** Deskriptive Statistik

**6** Plots

**7** Regressionsanalyse

**8** Weiterführende Literatur

# Weiterführende Literatur

---

## Nützliche Websites & Tutorials

- Websites
  - <https://www.statmethods.net/r-tutorial/index.html>
  - <https://stackoverflow.com/>
  - <https://cran.r-project.org/>
  - <https://stats.stackexchange.com/questions>
  - <https://www.datacamp.com/courses/free-introduction-to-r>
- Tutorials
  - [Multiple Linear Regression in R | R Tutorial 5.3 | MarinStatsLectures - YouTube](#)
  - [R - Multiple Regression \(part 2\) - YouTube](#)

# Weiterführende Literatur

---

## Zusätzliche Literatur

- Lehrbücher
  - Grundlagen der Datenanalyse mit R - Eine anwendungsorientierte Einführung, Daniel Wollschläger, Berlin, Springer Spektrum, 2017
  - Angewandte Zeitreihenanalyse mit R, Rainer Schlittgen, München, Oldenbourg, 2012
  - Einführung in die Statistik mit R, Andreas Behr, München, Vahlen, 2011
- Reference Cards
  - <https://cran.r-project.org/doc/contrib/Short-refcard.pdf>
- Online-Skripte
  - [www.wiwi.uni-bielefeld.de/lehrebereiche/emeriti/jfrohn/Upload/statskript.pdf](http://www.wiwi.uni-bielefeld.de/lehrebereiche/emeriti/jfrohn/Upload/statskript.pdf)
  - [www.uni-muenster.de/Stochastik/lehre/SS17/PrakStat/Materialien/Skript.pdf](http://www.uni-muenster.de/Stochastik/lehre/SS17/PrakStat/Materialien/Skript.pdf)

Vielen Dank für Ihre Aufmerksamkeit!



Wirtschaftsingenieurwesen

Institut für Materials Resource Management

Universität Augsburg

[wing@mrm.uni-augsburg.de](mailto:wing@mrm.uni-augsburg.de)

[www.uni-augsburg.de/wing](http://www.uni-augsburg.de/wing)