



Programmierung und Deskriptive Statistik

BSc Psychologie WiSe 2024/25

Belinda Fleischmann

Herzlich Willkommen!



Aufnahme läuft.

(1) Einführung

Formalia

Grundbegriffe der Informatik

R und Visual Studio Code

Aufgaben und Selbstkontrollfragen

Formalia

Grundbegriffe der Informatik

R und Visual Studio Code

Aufgaben und Selbstkontrollfragen

Homepage der Abteilung für Methodenlehre I



INSTITUT FÜR PSYCHOLOGIE

INSTITUT | STUDIUM | FORSCHUNG | PERSONEN

Home > Institut > Abteilungen des Ins... > Methodenlehre I : Experimentelle und Neuro... > Forschung | Lehre | Team

Methodenlehre I : Experimentelle und Neurowissenschaftliche Psychologie

Forschung



Lehre



Team



C1. Programmierung und Deskriptive Statistik

- Einführung in die datenanalytische Programmierung mit R in Visual Studio Code
- Einführung in die Auswertung von deskriptiven Statistiken

C2. Analyse und Dokumentation

- Praktische Analyse empirischer Daten
- Dokumentation empirischer Studien und Analysen

Programmierung

- Einführung in die Programmiersprache R und die Entwicklungsumgebung Visual Studio Code (VS Code).
- Grundlegende Konzepte der datenanalytischen Programmierung in R, einschließlich arithmetischer Operationen, logischer Operationen, Variablen, Datenstrukturen und Kontrollstrukturen.
- Einführung in wesentliche Werkzeuge und Pakete für die Durchführung deskriptiver statistischer Analysen mit R.

Deskriptive Statistik

- Grundlegender Methoden der deskriptiven Statistik zur Analyse und Beschreibung quantitativer Daten.
- Praktische Übung anhand von Anwendungsbeispielen.

Seminar und Präsenzübung

- Theoretische und praktische Einführung in die wesentlichen Konzepte der Programmierung und deskriptiven Statistik.
- Anwendung der erlernten Konzepte in praktischen Übungen, unterstützt durch Hilfestellung von Lehrpersonen und Tutor:innen während der Präsenzübungen.

Programmierübungen

- Neben den Präsenzzeiten werden wöchentliche Vor- und Nachbereitung der Programmierübungen und Selbstkontrollfragen dringend empfohlen!
- Die Computerplätze am URZ sind außerhalb der Belegzeiten (siehe [G26.1-006](#) oder [G26.1 007](#)) öffentlich zugänglich.

Leistungsnachweis

- Selbstständige Bearbeitung von Übungsaufgaben und deren Einreichung in Form von R Skripten.
- Leistungsnachweis gilt als erbracht, wenn mind. 50 % der abgegebenen Leistungen als bestanden bewertet wurden.

- Termine: Dienstags
 - Gruppe 1: 11-13 Uhr in G26.1 007 (Persönlicher Laptop optional)
 - Gruppe 2: 13-15 Uhr in G16-215 (Persönlicher Laptop notwendig)
- Kursmaterialien (Folien, Videos) auf der [Kurswebseite](#)
- Codes und tagesaktuelle Folien auf [GitHub](#)
- Vorherige Iteration des Kurses: [PDS \(WS 2023/24\)](#)
- Empfohlene Vorbereitung: [Vorkurs "Mathematische Grundlagen"](#)
- Ankündigungen und Fragen-Forum im [Mattermost-Channel](#)
- Abgabe und Evaluation des Leistungsnachweises über den [Moodlekurs](#)

Termine

Datum	Einheit	Thema	Form
15.10.24	R Grundlagen	(1) Einführung	Seminar
22.10.24	R Grundlagen	(2) R und Visual Studio Code	Seminar
29.10.24	R Grundlagen	(2) R und Visual Studio Code	Übung
05.11.24	R Grundlagen	(3) Vektoren, (4) Matrizen	Seminar
12.11.24	R Grundlagen	(5) Listen und Dataframes	Seminar
	<i>Leistungsnachweis 1</i>		
19.11.24	R Grundlagen	(6) Datenmanagement	Seminar
26.11.24	R Grundlagen	(2)-(6) R Grundlagen	Übung
03.12.24	Deskriptive Statistik	(7) Häufigkeitsverteilungen	Seminar
10.12.24	Deskriptive Statistik	(8) Verteilungsfunktionen und Quantile	Seminar
	<i>Leistungsnachweis 2</i>		
17.12.24	Deskriptive Statistik	(9) Maße der zentralen Tendenz und Datenvariabilität	Seminar
	Weihnachtspause		
07.01.25	R Grundlagen	(10) Strukturiertes Programmieren: Kontrollfluss, Debugging	Seminar
14.01.25	Deskriptive Statistik	(11) Anwendungsbeispiel	Übung
	<i>Leistungsnachweis 3</i>		
21.01.25	Deskriptive Statistik	(11) Anwendungsbeispiel	Seminar
28.01.25	Deskriptive Statistik	(11) Anwendungsbeispiel, Q&A	Seminar

Webseite des Kurses (Folien, Videos)



OTTO VON GUERICKE
UNIVERSITÄT
MAGDEBURG

INSTITUT FÜR PSYCHOLOGIE

Sitemap Impressum Kontakt

Suchbegriff



INSTITUT | STUDIUM | FORSCHUNG | PERSONEN

DIREKTLINKS ▾

Home >> Methodenlehre I >> Lehre >> Wintersemester >> Programmierung und Deskriptive Stati...

Programmierung und Deskriptive Statistik

Dieser Kurs gibt eine Einführung in die datenanalytische Programmierung und die Auswertung deskriptiver Statistiken mit [R](#) in [Visual Studio Code](#).

Nach der [Studien- und Prüfungsordnung für den BSc Psychologie \(06/2020\)](#) und dem [Modulhandbuch für den BSc Psychologie \(09/2020\)](#) entspricht dieser Kurs dem Modul [CT Computergestützte Datenanalyse](#).

Der Quarto Code der Vorlesungsfolien ist auf [github](#) verfügbar.

Zur Überprüfung der erfolgreichen Teilnahme an diesem Kurs wird ein Leistungsnachweis in zwei Teilen abgegeben.

Als weiterführende Literatur zur Programmierung mit R werden empfohlen:

- Sauer, S. (2019) *Moderne Datenanalyse mit R*
- Cotton, R. (2013) *Learning R*.
- Wickham, H. (2019) *Advanced R*
- Wickham, H., Çetinkaya-Rundel, M., & Grolemund, G. (2023) *R for Data Science*
- Murrell, P. (2021) *R Graphics*

Als Einstieg in die Deskriptive Statistik bieten sich an

- Henze, N. (2016) *Deskriptive Statistik*
- Fahrmeier et al. (2016) *Statistik, Kapitel 1-3*

Vorlesungseinheiten

(1) Einführung



Kontakt

Abteilungsleitung

• Prof. Dr. Dirk Ostwald

✉ dirk.ostwald@ovgu.de

Tel.: + 49 391 67 5370

Abteilungsassistentz

• Birgit Müller

✉ birgit.mueller@ovgu.de

Tel.: +49 391 67 58464

Anschrift

Otto-von-Guericke-Universität

Magdeburg

Institut für Psychologie

Universitätsplatz 2

Gebäude 24

39106 Magdeburg

• [Anfahrt](#)

Git-repository des Kurses (Code und Folien)

The screenshot shows the GitHub interface for the repository 'progund-deskr-stat-25'. The repository is public and has 21 commits. The file list includes folders for course topics and various configuration files.

File/Folder	Description	Time Ago
1_Einfuehrung	Rephrase exercise section headings to use "Programmie...	2 weeks ago
2_R_und_VSCode	Update Programmieraufgaben and small files	last week
3_Vektoren	Add instructions and exercises for help and viewer in [3] ...	2 weeks ago
4_Matrizen	Add instructions and exercises for help and viewer in [3] ...	2 weeks ago
5_Listen_und_Dataframes	Add screenshots of list and table viewer to [5]	5 days ago
6_Datenmanagement	Small Files (6)	5 days ago
7_Haeufigkeitsverteilungen	Add script for demonstrating VSCode Table Viewers	5 days ago
8_Verteilungsfunktionen_und_Quartile	Add [8]	5 days ago
9_Masse_der_Zentralen_Tendenz_und_Varia...	Add [9]	5 days ago
Abbildungen	Add images and data	5 days ago
Daten	Add images and data	5 days ago
.gitignore	chore: Add initial project files and configuration	last month
Headstex	chore: Add initial project files and configuration	last month
README.md	chore: Add initial project files and configuration	last month
R_common.R	chore: Add initial project files and configuration	last month
Referenzen.bib	chore: Add initial project files and configuration	last month

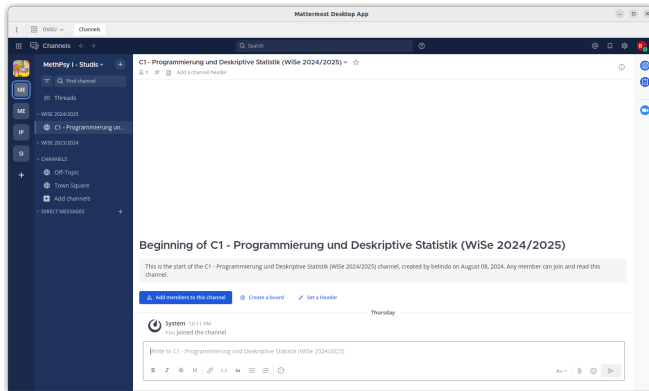
About
No description, website, or topics provided.

Releases
No releases published
[Create a new release](#)

Packages
No packages published
[Publish your first package](#)

Languages

- TeX 99.7%
- R 0.3%



- Registrierung und Beitritt zum **Team MethPsy I - Studis**.
- Beitritt zum **Channel C1 - Programmierung und Deskriptive Statistik (WiSe 2024/2025)**
- Zugang über **Web-App** (gleicher Link wie oben), **Desktop-App** oder **Mobile App** möglich.

Forum als Community Tool

Wenn ihr Fragen habt, die für alle relevant sind oder auch andere beschäftigen könnten, stellt sie bitte im Forum. Schaut vorher, ob jemand schon eine ähnliche Frage gestellt hat. Das vermeidet Dopplungen und fördert den Austausch! Beantwortet gegenseitig eure Fragen – anderen etwas zu erklären, ist oft der beste Weg, um das eigene Verständnis zu schärfen.

Seid präzise und spezifisch

Formuliert eure Frage bitte möglichst spezifisch. Statt „Ich verstehe das Thema X nicht.“ formuliert spezifische Fragen wie „Ist im dritten Punkt zum Thema X gemeint, dass...?“. Präzise Fragen führen zu klaren Antworten und helfen euch zudem, das Thema bereits beim Formulieren besser zu durchdringen. Oft lösen sich Fragen dabei auch von selbst.





Kontext ist Key

Verweist bitte immer auf das Thema, die Foliennummer oder die Übungsaufgabe. Am besten direkt mit der Verlinkungsfunktion in Mattermost. So können wir die Frage schneller im richtigen Zusammenhang verstehen und beantworten.

Nutzt **Markdown in Mattermost**, um Inhalte übersichtlicher zu formatieren!

Use Markdown

You can also format your messages in Mattermost using Markdown to control [text styling](#), [links](#), [headings](#), [lists](#), [code blocks](#), [in-line code](#), [in-line images](#), [horizontal lines](#), [block quotes](#), [tables](#), and [math formulas](#). Markdown makes it easy to format messages: type a message as you normally would, then use formatting syntax to render the message a specific way. For a guide to using Markdown in Mattermost, [see this blog post](#).

Text Entered	How It Appears
<code>_italics_</code>	<i>italics</i>
<code>**bold**</code>	bold
<code>~~strikethrough~~</code>	strikethrough
<code>"In-line code"</code>	<code>In-line code</code>
<code>[hyperlink](http://mattermost.org)</code>	hyperlink
<code></code>	 build passing
<code>:smile: :sheep: :alien:</code>	  

Text style

You can use either `_` or `*` around a word or phrase to make it italic, or `__` or `***` around a word or phrase to make it bold.

Tip

Common formatting keyboard shortcuts are supported. Bold text by pressing `Ctrl` + `B` on Windows and Linux, or `Cmd` + `B` on Mac. Italicize text by pressing `Ctrl` + `I` on Windows or Linux, or `Cmd` + `I` on Mac.

- `*italics*` (or `_italics_`) renders as *italics*
- `***bold***` renders as **bold**
- `***bold-italics***` renders as ***bold-italics***
- `~~strikethrough~~` renders as ~~strikethrough~~

Q & A

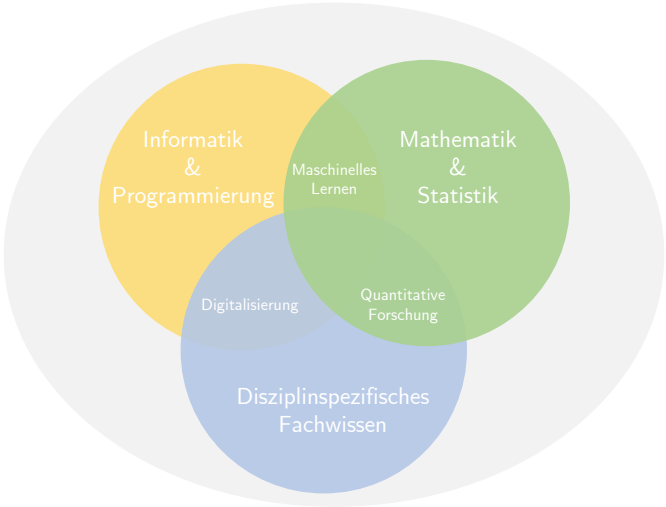
Formalia

Grundbegriffe der Informatik

R und Visual Studio Code

Aufgaben und Selbstkontrollfragen

Zentrale Komponenten der Datenwissenschaft



Formalia

Grundbegriffe der Informatik

- Datenanalyse
- Informatik
- Rechnerarchitektur
- Algorithmen und Programme

R und Visual Studio Code

Aufgaben und Selbstkontrollfragen

Formalia

Grundbegriffe der Informatik

- **Datenanalyse**
- Informatik
- Rechnerarchitektur
- Algorithmen und Programme

R und Visual Studio Code

Aufgaben und Selbstkontrollfragen

- Wissenschaftliche Daten liegen heutzutage als digitale Daten vor.
- Digitale Daten werden mit Hilfe eines Computers analysiert.
- Zur Analyse von digitalen Daten schreibt man Computerprogramme.
- Diese Computerprogramme heißen Datenanalyseskripte.

1. Einlesen und Bereinigen eines digitalen Datensatzes.
2. Berechnung und Visualisierung deskriptiver Statistiken.
3. Probabilistische Modellierung und Inferenz.
4. Dokumentation und Präsentation der Ergebnisse.

Typische Werkzeuge zur Analyse psychologischer Daten

- **R** (frei, Datenwissenschaft, Statistik, Psychologie)
- **Python** (frei, Datenwissenschaft, Anwendung)
- **Matlab** (kommerziell, Engineering, Neuroimaging)

Altmodisch

- **SPSS** (kommerziell, Sozialwissenschaften, Psychologie)
- **JMP** (kommerziell, Biologie, Psychologie)
- **STATA** (kommerziell, Wirtschaftswissenschaften)

Programmiersprachen Trends

PYPL Index (Stand: August 2024)

Worldwide, Aug 2024 :

Rank	Change	Language	Share	1-year trend
1		Python	29.6 %	+1.7 %
2		Java	15.51 %	-0.3 %
3		JavaScript	8.38 %	-1.0 %
4		C#	6.7 %	-0.0 %
5		C/C++	6.31 %	-0.2 %
6	↑	R	4.6 %	+0.2 %
7	↓	PHP	4.35 %	-0.6 %
8		TypeScript	2.93 %	-0.1 %
9		Swift	2.76 %	+0.1 %
10	↑	Rust	2.58 %	+0.5 %
11	↓	Objective-C	2.4 %	+0.2 %
12		Go	2.14 %	+0.2 %
13		Kotlin	1.94 %	+0.2 %
14		Matlab	1.5 %	-0.0 %

- PopularitY of Programming Language
- Basierend auf GoogleSuchanfragen zu Programmiersprachentutorials

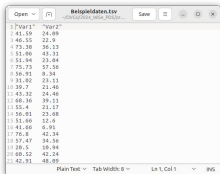
- Dokumentation aller Schritte von Rohdaten bis zur Datenvisualisierung.
- Reproduktion wissenschaftlicher Ergebnisse durch Dritte.
- Essentieller Teil wissenschaftlicher Publikationen.
- Essentieller Teil täglicher wissenschaftlicher Arbeit.

Beispiel

```
Datenanalysekript_Beispiel.R - 2024_WiSe_PDS - Visual Studio Code
Datenanalysekript_Beispiel.R U X
prog-und-deskr-stat-25 > 1_Einfuehrung > Datenanalysekript_Beispiel.R > --
1 # Beispiel eines einfachen Datenanalysekripts
2 # -----
3 # Dieses Skript lädt den Datensatz "Beispieldaten_1.csv", und berechnet
4 # Mittelwerte und Korrelation.
5 # -----
6 # Seminar: "Programmierung und Deskriptive Statistik" WiSe 2024/2025
7 # Autorin: Belinda Fleischmann
8 # Datum: --28.08.2024
9
10 # Daten von Festplatte einlesen
11 daten <- read.csv("prog-und-deskr-stat-25/1_Einfuehrung/Beispieldaten.csv")
12
13 # Mittelwerte der Variablen berechnen
14 mittelwert_1 <- mean(daten$Variable1)
15 mittelwert_2 <- mean(daten$Variable2)
16
17 # Korrelation der Variablen berechnen
18 korrelation <- cor(daten$Variable1, daten$Variable2)
19
20 # Ausgabe der Ergebnisse
21 cat("Der Mittelwert der ersten Variable beträgt", mittelwert_1, "\n")
22 cat("Der Mittelwert der zweiten Variable beträgt", mittelwert_2, "\n")
23 cat("Die Korrelation beträgt", korrelation, "\n")
24
25 # Visualisierung der Daten
26 plot(daten$Variable1, daten$Variable2)
27 barplot(c("Var1" = mittelwert_1, "Var2" = mittelwert_2))
28
```

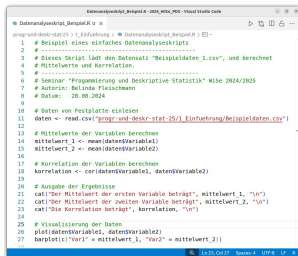
Intuition zur Struktur der Computergestützten Datenanalyse

Datensatz



	Var1	Var2
1	41.59	24.89
2	46.55	22.9
3	72.38	36.15
4	51.86	43.31
5	51.94	23.94
6	72.73	27.56
7	56.91	8.34
8	31.82	23.11
9	39.7	21.46
10	43.32	24.46
11	66.36	39.11
12	55.4	21.17
13	56.81	23.88
14	51.66	12.4
15	41.66	6.91
16	76.8	42.24
17	57.47	34.56
18	28.5	18.84
19	60.52	42.24
20	42.91	48.89

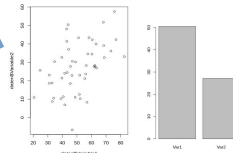
Datenanalysekript



```
1 # Beispiel eines einfachen Datenanalysekripts
2 # .....
3 # Dieses Skript lädt den Datensatz "Beispieldaten_1.csv", und berechnet
4 # Mittelwerte und Korrelation.
5 # .....
6 # Seminar "Programmierung und Deskriptive Statistik" WiSe 2024/2025
7 # Autor:in: Belinda Fleischmann
8 # Datum: 28.08.2024
9
10 # Daten von Festplatte einlesen
11 daten <- read.csv("prog-und-destr-stat-25/1_Einfuehrung/Beispieldaten.csv")
12
13 # Mittelwerte der Variablen berechnen
14 mittelwert_1 <- mean(daten$Variable1)
15 mittelwert_2 <- mean(daten$Variable2)
16
17 # Korrelation der Variablen berechnen
18 korrelation <- cor(daten$Variable1, daten$Variable2)
19
20 # Ausgabe der Ergebnisse
21 cat("Der Mittelwert der ersten Variable beträgt", mittelwert_1, "\n")
22 cat("Der Mittelwert der zweiten Variable beträgt", mittelwert_2, "\n")
23 cat("Die Korrelation beträgt", korrelation, "\n")
24
25 # Visualisierung der Daten
26 plot(daten$Variable1, daten$Variable2)
27 barplot(c("Var1" = mittelwert_1, "Var2" = mittelwert_2))
28
```

Der Mittelwert der ersten Variable beträgt 50.516
Der Mittelwert der zweiten Variable beträgt 27.1598
Die Korrelation beträgt 0.4851987

Ergebnisse als Text



Ergebnisse als Visualisierungen

- Die Digitalisierung betrifft insbesondere auch die Wissenschaft.
- Forschungsdatenmanagement ist eine akute Herausforderung.
- Programmierung als zentrales Handwerkszeug wissenschaftlicher Arbeit.
- Informatikkenntnisse sind in der Arbeitswelt unverzichtbar.
- Dies gilt auch für Psychotherapeut:innen (z.B. Online-Intervention).

Formalia

Grundbegriffe der Informatik

- Datenanalyse
- **Informatik**
- Rechnerarchitektur
- Algorithmen und Programme

R und Visual Studio Code

Aufgaben und Selbstkontrollfragen

Informatik (engl. Computer Science)

Bei der Informatik handelt es sich um die Wissenschaft von der systematischen Darstellung, Speicherung, Verarbeitung und Übertragung von Informationen, wobei besonders die automatische Verarbeitung mit Computern betrachtet wird. Sie ist zugleich Grundlagen- und Formalwissenschaft als auch Ingenieurdisziplin.

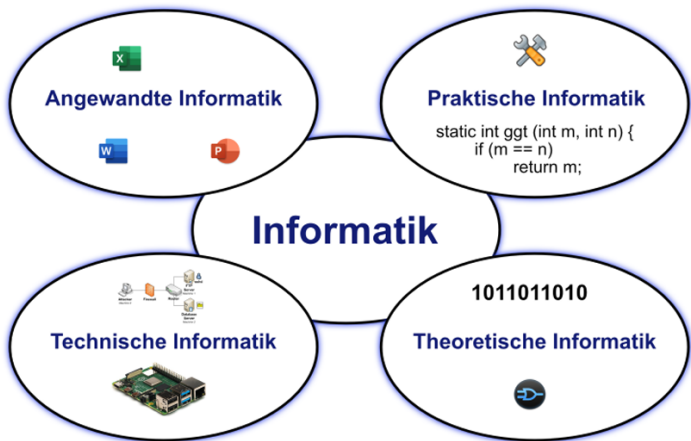
Wikipedia

Computer

- Maschinen zum Datenspeichern und Ausführen einfacher Datenoperationen.
- Einfache Operationen mit extrem hoher Geschwindigkeit.
- Universalität durch Speicherung von Daten und Programmen.

Algorithmen und Programme

- *Programme* sind in einer *Programmiersprache* verfasste *Algorithmen*.
- Algorithmen sind Folgen von Anweisungen durchzuführender Operationen.
- Bei Algorithmen unterscheidet man
 - Beschreibung (Kochrezept, IKEA Bauanleitung, R Skript)
 - Anweisungen (“Mehl und Wasser vermengen”, $o - - -, x = c(1,2,3)$)
 - Durchführung (Kochvorgang, Zusammenbau, R Skript laufen lassen)



Hattenhauer (2020) Informatik

Technische Informatik

- Mikroprozessortechnik, Rechnerarchitektur, Netzwerktechnik

Theoretische Informatik

- Automatentheorie, Berechenbarkeitstheorie, Komplexitätstheorie

Praktische Informatik

- Programmierung, Algorithmen, Datenbanken

Angewandte Informatik

- Anwendungssoftware, Human-Computer-Interaction, Informatik und Gesellschaft

Maschinelles Lernen und Künstliche Intelligenz

- Datenanalyse aus Sicht der Informatik

Computervisualistik

- Bilderkennung und Bildsynthese, Virtuelle Realität, Augmented Reality

Computerlinguistik

- Spracherkennung und Sprachsynthese

Bioinformatik

- Lebenswissenschaften, Genomik, Bildgebende Verfahren der Medizin

Formalia

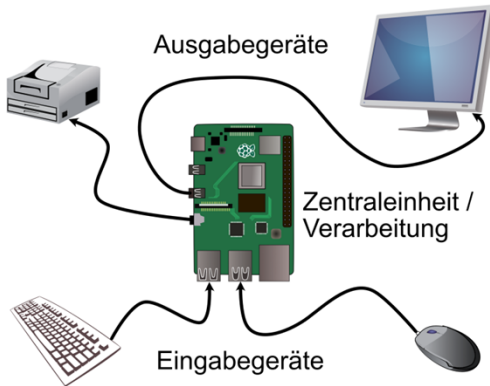
Grundbegriffe der Informatik

- Datenanalyse
- Informatik
- **Rechnerarchitektur**
- Algorithmen und Programme

R und Visual Studio Code

Aufgaben und Selbstkontrollfragen

EVA-Prinzip: Eingabe → Verarbeitung → Ausgabe



Hattenhauer (2020) Informatik

Zentraleinheit eines Computers

Auch Hauptplatine, Motherboard oder Mainboard genannt

High Performance Gaming
Unsere Top-Konfiguration zum selbst Zusammenbauen.

Component	Price
Hochleistungsprozessor	359.-
High Performance Grafikkarte	379.-
DVD Multiform Laufwerk	15.99
Arbeitsspeicher	64.90
'State of the Art' Mainboard	229.-
Lesens Netzteil	54.99
Mid Tower ATX „Silent“	64.90
Hochleistungs HDD	139.-
SanDisk schnelle SSD	80.-
WLAN Adapter	22.45
Alle Komponenten in den Warenkorb	1473.73

Figure 1: Hattenhauer (2020) Informatik

Hattenhauer (2020) Informatik

Zentraleinheit eines Computers

CPU (Central Processing Unit/Mikroprozessor)

- Rechenwerk, Steuerwerk, und Leitwerk des Systems
- Cache (flüchtiger schneller Speicher)
- Intel(R) Core(TM) i5-7300HQ CPU @ 2.50GHz

RAM (Random Access Memory)

- Temporärer, flüchtiger Arbeitsspeicher des Systems
- Begrenzt, z.B. 16 GB

Massenspeicher

- Stationärer Speicher des Systems
- SSD (Solid State Drive), Cloudspeicher

GPU (Graphical Processing Unit)

- Leistungsstarke, speziell für Visualisierung optimierte Prozessoren
- Unterstützung der CPU in manchen Anwendungen, z.b. Neuronale Netze

Von Neumann-Architektur

Abstraktion eines Rechensystems mit wohldefinierten Komponenten und Datenflüsse.

Rechner := Steuerwerk, Rechenwerk, Speicher, Eingabewerk, Ausgabewerk.

Zentrale Eigenschaften

- Struktur des Rechners unabhängig von dem zu bearbeitenden Problem.
- Daten, Programme, Zwischen- und Endergebnisse liegen im gleichen Speicher.
- Speicher ist in gleichgroße nummerierte (adressierte) Zellen unterteilt.
- Über die Adresse einer Speicherzelle kann deren Inhalt abgerufen verändert werden.
- Ein Programm ist eine Reihe von Befehlen.
- Aufeinanderfolgende Befehle eines Programms liegen in benachbarten Speicherzellen und werden entsprechend nacheinander aufgerufen.

→ **Die Architektur eines Rechners impliziert das Grundprinzip der Programmierung:
Befehle werden streng sequentiell abgearbeitet.**

Formalia

Grundbegriffe der Informatik

- Datenanalyse
- Informatik
- Rechnerarchitektur
- **Algorithmen und Programme**

R und Visual Studio Code

Aufgaben und Selbstkontrollfragen

Vom Realweltproblem zum Programm

Realweltproblem

- Das Problem, das mithilfe eines Computers gelöst werden soll.
- z.B. Auswertung von Fragebogendaten einer psychologischen Studie.

Problemspezifikation

- Genaue sprachliche Fassung des Realweltproblems.
- z.B. Methodenteil einer wissenschaftlichen Publikation.

Algorithmus

- Folge von Anweisungen zur Lösung des Problems.
- z.B. Dateneinlesen, deskriptive Statistiken berechnen, T-Test durchführen.

Programm

- Ein Algorithmus, der von einem Computer ausgeführt werden kann.
- Eine in einer Programmiersprache verfasste Textdatei.

Definition (Algorithmus)

Ein *Algorithmus* ist eine Folge von Anweisungen, um aus gewissen Eingabedaten bestimmte Ausgabedaten herzuleiten, wobei folgende Bedingungen erfüllt sein müssen

- *Fintheit*. Die Anweisungsfolge muss in einem endlichen Text vollständig beschrieben sein.
- *Effektivität*. Jede Anweisung muss tatsächlich ausführbar sein.
- *Terminierung*. Der Algorithmus endet nach endlich vielen Anweisungen.
- *Determiniertheit*. Der Ablauf des Algorithmus ist zu jedem Punkt fest vorgeschrieben.

Wenn E die Menge der zulässigen Eingabedaten und A die Menge der zulässigen Ausgabedaten bezeichnet, dann ist ein Algorithmus eine Funktion

$$f : E \rightarrow A, e \mapsto f(e) \quad (1)$$

Umgekehrt heißen Funktionen, die durch einen Algorithmus beschrieben werden können, *berechenbare Funktionen*.

Bemerkung

- Effektivität sollte nicht mit Effizienz verwechselt werden.

Eine Programmiersprache

- ... bestimmt die Regeln, denen ein Programm gehorchen muss.
- ... definiert eine Syntax, also Vokabular und Programmaufbau.
- ... definiert Semantik, also die Bedeutung der erlaubten Anweisungen.

```
#if [ -z "$USER_NAME" -o -z "$USER_TYPE" -o -z "$GROUP" ]
if [ -z "$USER_NAME" -o -z "$USER_TYPE" ]
then
#     echo "Please set the user name, type and group"
     echo "Please set the user name and type"
     exit 1
fi

# generate a random password
# -y: include special characters
# -n: include numbers
# -l: one generated passwords per Line
#pwgen -y 15 -n 5 -l
echo "Propositions for random passwords to use in next step:"
pwgen -s -n -l 15 5

# add the user
# requires password to be given via input
adduser --firstuid 1000 --lastuid 9999 --no-create-home ${USER_NAME}
```

Maschinensprache

- Elementare Operationsbefehle (z.B. Speichern, Vergleichen, Addieren)
- Elementare Operationsbefehle werden als Binärzahlen kodiert

Addiere Inhalt R1 zu Inhalt R2 \Rightarrow 1001 0010

Erhöhe Inhalt R1 um 1 \Rightarrow 1001 0110

Übertrage Inhalt R1 nach R3 \Rightarrow 0010 0011

- Programme in Maschinensprache heißen *Maschinenprogramme*.
- De facto führt ein Computer nur Maschinenprogramme aus.
- Für Menschen ist die Programmierung in Maschinensprache mühselig.

Höhere Programmiersprache

- An die menschliche Sprache angelehnte Wörter und Sätze
- Interpreter oder Compiler übersetzen Programme in Maschinensprache
- R, Python, Matlab, C++, Java, FORTRAN, COBOL,...

Generationen von Programmiersprachen

1. Generation (1GL)

- Maschinensprachen
- 10110000 01100001 (hexadezimaler Darstellung des Ausdrucks "B0 61")

2. Generation (2GL)

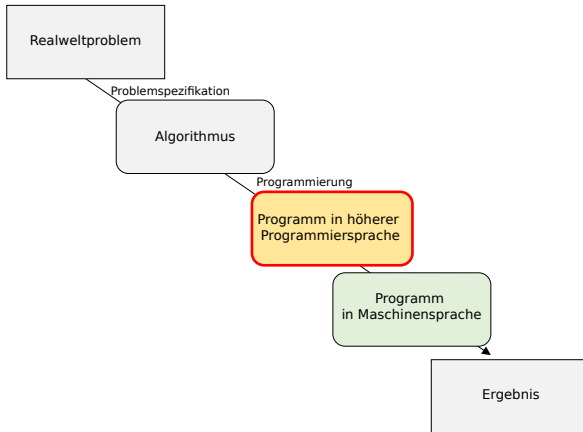
- Assemblersprachen ab 1950, erste Form der symbolischen Programmierung
- Bspw. "MOV AI, 61H" # Intel-Prozessor-spezifische Sprache

3. Generation (3GL)

- Höhere Programmiersprachen ab 1970 wie FORTRAN, C, C++, Java
- Programmierfreundlich, prozessor-unabhängig

4. Generation (4GL)

- Höhere Programmiersprachen ab 1980 wie Python, Matlab, R
- Codeoverhead Minimierung, Automation, Flexibilität, Multiparadigmatisch



Imperative Programmierung

- Problemlösungsweg wird als Folge von *Anweisungen (Befehlen)* vorgegeben.
- Befehle verarbeiten Daten, die mithilfe von *Variablen* adressiert werden.
 - **Prozedurale imperative Programmierung**
 - Daten und sie manipulierende Befehle werden separat behandelt.
 - Prozeduren (Funktionen) bilden das zentrale Strukturkonzept.
 - **Objektorientierte imperative Programmierung**
 - Daten und manipulierende Befehle werden als *Objekte* zusammengefasst.
 - Objekte bilden das zentrale Strukturkonzept.
- Praktisch liegen oft Mischformen vor.

Kompilierte Programmiersprachen

- Gesamter Quellcode wird *vor der Ausführung* in Maschinensprache übersetzt.
- Das Übersetzungsprogramm heißt *Compiler*.
- Der übersetzte Maschinencode wird vom Prozessor ausgeführt.
- Das ausführbare Programm wird nicht übersetzt und läuft schnell.
- Bei Änderungen des Quellcodes muss neu kompiliert werden.
- Beispiele für kompilierte Sprachen sind Java, C, C++.

Interpretierte Programmiersprachen

- Quellcode wird *während der Ausführung* in maschinennahe Sprache übersetzt.
- Das Ausführungsprogramm heißt *Interpreter*.
- Das Programm läuft aufgrund der Interpretation langsamer.
- Bei Änderungen des Quellcodes muss nicht neu interpretiert werden.
- Beispiele für interpretierte Sprachen sind Python und R.

Formalia

Grundbegriffe der Informatik

R und Visual Studio Code

Aufgaben und Selbstkontrollfragen

Was ist R?

- Eine Programmiersprache und ein Softwarepaket.
- Entwickelt von Ihaka and Gentleman (1996).
- Freier Dialekt der proprietären Software S (Becker, Chambers, and Wilks (1988)).
- Weiterentwickelt und gepflegt durch [R Core Team](#) und [R Foundation](#)
- Interpretierte imperative 4GL Sprache.
- Optimiert und populär für statistische Datenanalysen.
- Große Community mit etwa 20.000 beigetragenen [R Paketen](#) (Erweiterungen)
- Evolviert und konservativ im Kern, konsistent und progressiv in R Paketen.

Wie bekomme ich R?

Über cran.r-project.org die geeignete Version herunterladen und installieren.



CRAN
Home
What's new?
Search
CRAN Team

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Link Views
Other

Documentation
Manuals
FAQs
Contributed

Download and Install R
Precompiled binary distributions of the base system and contributed packages, Windows and Mac users most likely want one of these versions of R: <ul style="list-style-type: none">• Download R for Linux (Debian, Fedora/Redhat, Ubuntu)• Download R for macOS• Download R for Windows
R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.
Source Code for all Platforms
Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it! <ul style="list-style-type: none">• The latest release (2022-06-23, Funny-Looking Kid) R-4.2.1.tar.gz, read what's new in the latest version.• Sources of R.alpha and beta releases (daily snapshots, created only in time periods before a planned release).• Daily snapshots of current patched and development versions are available here. Please read about new features and bug fixes before filing corresponding feature requests or bug reports.• Source code of older versions of R is available here.• Contributed extension packages
Questions About R
<ul style="list-style-type: none">• If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

What are R and CRAN?

R is 'GNU S', a freely available language and environment for statistical computing and graphics which provides a wide variety of statistical and graphical techniques: linear and nonlinear modelling, statistical tests, time series analysis, classification, clustering, etc. Please consult the [R project homepage](#) for further information.

CRAN is a network of ftp and web servers around the world that store identical, up-to-date, versions of code and documentation for R. Please use the CRAN [mirror](#) nearest to you to minimize network load.

Submitting to CRAN

To "submit" a package to CRAN, check that your submission meets the [CRAN Repository Policy](#) and then use the [web form](#).

If this fails, send an email to CRAN-submissions@R-project.org following the policy. Please do not attach submissions to emails, because this will clutter up the mailboxes of half a dozen people.

Was können wir mit R machen?

Was kann R?

- Datensätze laden, manipulieren, und speichern.
- Eine Vielzahl von Berechnungen an verschiedenen Datenstrukturen durchführen.
- Eine Vielzahl statistischer Analysemethoden auf Daten anwenden.
- Datenanalyseskripte schreiben und Abbildungen generieren.
- Präsentationen [RMarkdown](#) und Bücher [RBookdown](#) erstellen.
- Wissenschaftliche Berichte mit [Quarto](#) erstellen.

Was kann R (bisher) nicht so gut?

- In einer ansprechenden Umgebung programmieren (⇒ Visual Studio Code).
- Scientific Computing (⇒ Python, Matlab, Julia).
- Psychologische Experimente programmieren (⇒ Python, Matlab)

Wie bekomme ich Hilfe zu R?

- Während der Programmierung und bei bekanntem Funktionsnamen über die Kommandozeile:

```
?mean           # Zeigt Hilfe zu der Funktion "mean()"
help(mean)      # Zeigt Hilfe zu der Funktion "mean()"
browseVignettes() # Zeigt Vignetten aller installierten Pakete im Browser
browseVignettes("knitr") # Zeigt Vignetten des Paktes "knitr"
```

- Googlen
- stackoverflow.com
- r-project.org/help.html
- rseek.org
- rstudio.com/resources/cheatsheets
- r-bloggers.com

Was ist Visual Studio Code (VSCode)?

- VSCode ist ein kostenloser Quelltext-Editor von Microsoft.
- VSCode ist eine Softwareentwicklungsumgebung (Integrated Development Environment, IDE)
- Seit 2015 für Windows, macOS und Linux verfügbar.
- Seit 2018 ist VSCode der beliebteste Editor laut jährlicher [stackoverflow Umfragen](#).
- Ein Microsoftprodukt ist damit auch der beliebteste Editor der Linuxwelt.
- Über Extensions kann VSCode als IDE für beliebige Sprachen genutzt werden.
- Zum Beispiel funktioniert VSCode als IDE für R, Python, Julia, Shell, Quarto, etc.
- VSCode ist Community-based und sehr konfigurierbar.
- VSCode ist über Microsoft's GitHub über Endgeräte synchronisierbar.

Wie bekomme ich VSCode?

Über <https://code.visualstudio.com> heruntergeladen und installiert.

Visual Studio Code Docs Updates Blog API Extensions FAQ Learn

Version 1.83 is now available! Read about the new features and fixes from September.

Code editing. Redefined.

Free. Built on open source. Runs everywhere.

Download .deb (Debian, Ubuntu, ...) or .rpm (Red Hat, Fedora, ...)

Web, Insiders edition, or other platforms

By using VS Code, you agree to its license and privacy statement.

IntelliSense Run and Debug Built-in Git Extensions

Wie benutze ich VSCode?

Online Dokumentation: <https://code.visualstudio.com/docs>

The screenshot shows the Visual Studio Code documentation website. At the top, there is a navigation bar with links for 'Visual Studio Code', 'Docs', 'Updates', 'Blog', 'API', 'Extensions', 'FAQ', and 'Learn'. A search bar and a 'Download' button are also present. Below the navigation bar, a banner for 'Version 1.82' is visible. The main content area is titled 'Getting Started' and contains a paragraph describing Visual Studio Code as a lightweight but powerful source code editor. Below this, there is a section titled 'Visual Studio Code in Action' which features a code editor snippet showing JavaScript code for an Express.js server. The code is:

```
4 var server = express();
5 server.use(bodyParser.json);
6
7 server.get
8
9
10
11
12
13
14
15
16
17
18
```

 The code editor shows an autocomplete menu for the `server.get` method, listing properties like `get`, `getMaxListeners`, `arguments`, `engine`, `length`, `merge`, `purge`, `settings`, `toString`, and `defaultConfiguration`. Below the code editor, there is a section titled 'Intelligent Code Completion' with the text: 'Code smarter with IntelliSense - completions for variables, methods, and imported modules.' The right sidebar contains a 'GETTING STARTED' section with links for 'VS Code in Action', 'Top Extensions', 'First Steps', 'Keyboard Shortcuts', 'Downloads', 'Privacy', 'Subscribe', 'Ask questions', 'Follow @code', 'Request features', 'Report issues', and 'Watch videos'. A left sidebar contains a navigation menu with categories like 'OVERVIEW', 'SETUP', 'GET STARTED', 'USER GUIDE', 'SOURCE CONTROL', 'TERMINAL', 'LANGUAGES', 'NODE.JS / JAVASCRIPT', 'TYPESCRIPT', 'PYTHON', 'JAVA', 'C++', 'CF', 'DOCKER', 'DATA SCIENCE', 'AZURE', 'REMOTE', and 'DEV CONTAINERS'.

Visual Studio Code

Docs Updates Blog API Extensions FAQ Learn

Search Docs Download

Version 1.82 is now available! Read about the new features and fixes from September.

Getting Started

Visual Studio Code is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS and Linux. It comes with built-in support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages and runtimes (such as C++, C#, Java, Python, PHP, Go, .NET). Begin your journey with VS Code with these [introductory videos](#).

Visual Studio Code in Action

```
4 var server = express();
5 server.use(bodyParser.json);
6
7 server.get
8
9
10
11
12
13
14
15
16
17
18
```

Intelligent Code Completion

Code smarter with IntelliSense - completions for variables, methods, and imported modules.

GETTING STARTED

- VS Code in Action
- Top Extensions
- First Steps
- Keyboard Shortcuts
- Downloads
- Privacy
- Subscribe
- Ask questions
- Follow @code
- Request features
- Report issues
- Watch videos

OVERVIEW

- SETUP
- GET STARTED
- USER GUIDE
- SOURCE CONTROL
- TERMINAL
- LANGUAGES
- NODE.JS / JAVASCRIPT
- TYPESCRIPT
- PYTHON
- JAVA
- C++
- CF
- DOCKER
- DATA SCIENCE
- AZURE
- REMOTE
- DEV CONTAINERS

Formalia

Grundbegriffe der Informatik

R und Visual Studio Code

Aufgaben und Selbstkontrollfragen

Aufgaben bis nächste Woche

1. R installieren.
2. VSCode installieren.
3. VSCode für R startklar machen (Konsultiere dafür die [Online-Doku](#)).
4. Selbstkontrollfragen bearbeiten.
5. Mit dem eigenen oder Labor-Rechner vertraut machen.
 - Wie viel Festplatten- und Arbeitsspeicher besitzt der Rechner?
 - Welches Betriebssystem in welcher Version ist installiert?
 - Erstelle einen Ordner für alle Materialien und Übungen dieses Kurses (z.B. 'PDS_2024') an einem selbst gewählten Speicherort.
 - Wie lautet die vollständige Speicheradresse (Pfad) zu diesem Kursordner?
 - Lade die Dateien "Beispiel_Datenanalyseskript.R" und "Beispieldaten.csv" herunter und speichere sie im Kursordner.
 - Probiere aus, mit welchen Programmen du diese Dateien jeweils öffnen kannst und mit welchen nicht.

Selbstkontrollfragen

1. Gib die typische Struktur einer computergestützten Datenanalyse wieder.
2. Erläutere den Begriff "Datenanalyseskript".
3. Definiere den Begriff "Informatik".
4. Erläutere die Akronyme CPU, RAM, SSD, und GPU.
5. Nenne die wesentlichen Aspekte der Von-Neumann Rechnerarchitektur.
6. Definiere den Begriff Algorithmus.
7. Erläutere den Zusammenhang von Algorithmen und Programmen.
8. Was bezeichnen Syntax und Semantik einer Programmiersprache?
9. Differenziere die Begriffe "Maschinensprache" und "höhere Programmiersprache".
10. Skizziere Prinzipien der prozeduralen und objektorientierten imperativen Programmierung.
11. Skizziere die Entwicklung der Programmiersprachen der ersten bis vierten Generation.
12. Differenziere die Begriffe der kompilierten und der interpretierten Programmiersprachen.

- Becker, Richard A., John M. Chambers, and Allen Reeve Wilks. 1988. *The New S Language: A Programming Environment for Data Analysis and Graphics*. Reprint. London: Chapman & Hall.
- Ihaka, Ross, and Robert Gentleman. 1996. "R: A Language for Data Analysis and Graphics." *Journal of Computational and Graphical Statistics* 5 (3): 2999–2314.