

Installparty macOS

Bioinformatik · Wintersemester 2026/27

Willkommen auf der Installparty!

Dieses 'Handout' erklärt, wie die Installation abläuft, was dabei installiert wird und wie ihr alles benutzt und aktuell haltet. Behaltet es gern für später, vieles davon braucht ihr in den ersten Semestern immer wieder.

Vor der Installparty

Bitte noch nichts selbst mit der Installation starten Klar, ihr freut euch darauf bereits richtig loszulegen und wollt vielleicht schon sofort selbst mit der Installation starten. Bitte wartet trotzdem auf die Installparty. Unser Skript sorgt dafür, dass alle Programme zusammenpassen und für eure Module funktionieren. Findet es ältere Installationen (z. B. Anaconda, Python von python.org oder mit pip installierte Pakete), müssen wir die erst gemeinsam aufräumen, und das dauert oft länger als die ganze Installation. Wenn ihr direkt zu uns kommt, seid ihr am schnellsten fertig und habt mehr Zeit für den gemütlichen Teil.

Damit auf der Installparty alles glattläuft, checkt bitte vorher Folgendes:

- ☐ **Es ist euer eigenes Gerät.** Die Installation sollte auf dem Gerät stattfinden, mit dem ihr im Studium arbeitet. Der Laptop, den ihr euch für einen Tag bei euren Eltern leiht, nützt euch nach der Installparty leider nichts. Das Skript braucht einen Mac mit Apple-Silicon-Chip (M1, M2, ...) und macOS 15 oder neuer. Nachschauen könnt ihr das unter *Apple-Menü* → *Über diesen Mac*. Ihr habt einen älteren Intel-Mac? Kommt trotzdem, dann installieren wir von Hand. Da es auf diesen einfacher war, geht es oft sogar schneller.
- ☐ **Admin-Konto und Passwort.** Euer Benutzerkonto braucht Administratorrechte und ihr müsst das Passwort kennen. Ohne ist eine Installation nicht möglich.
- ☐ **Backup.** Bei der Installation geht normalerweise nichts verloren, aber sicher ist sicher, und ein Backup solltet ihr ohnehin regelmäßig machen: *Systemeinstellungen* → *Allgemein* → *Time Machine* mit einer externen Festplatte.
- ☐ **macOS aktualisieren.** Installiert vorher alle Updates unter *Systemeinstellungen* → *Allgemein* → *Softwareupdate*. Das dauert manchmal etwas, wenn länger nicht geschehen. Aber Erfahrungsgemäß ließen sich einige Probleme im Installationsprozess durch Software-Updates lösen und sicherer ist es sowieso.
- ☐ **Speicherplatz.** Mindestens 15 GB sollten frei sein (*Systemeinstellungen* → *Allgemein* → *Speicher*). Mit weniger Platz installieren wir eine kleinere $\text{\LaTeX}$ -Variante, unter 8 GB geht es gar nicht.
- ☐ **FU-Account.** Ihr braucht euren FU-Benutzernamen und das Passwort, damit das Skript das schnelle WLAN *eduroam* einrichten kann. Wenn ihr noch keinen FU-Account habt (ein häufiges Problem - die Immatrikulation dauert), dann geben wir euch auf der Installparty ein LAN-Kabel, denn alles über das langsame Free-Wifi-Berlin runterzuladen dauert ewig.

- ❑ **Netzteil und Zeit.** Bringt euer Ladekabel mit und plant etwas Zeit ein. Die meiste Zeit laden Programme herunter, dabei könnt ihr quatschen und Kaffee trinken und müsst zwischendurch nur gucken, dass ihr euer Passwort eingibt.

So läuft die Installation

Die ganze Installation erledigt ein Skript. Ihr beantwortet nur ein paar Fragen, der Rest läuft von allein.

1. Terminal öffnen und das Skript starten

Öffnet das Terminal: **Cmd + Leertaste** drücken, „Terminal“ tippen, **Enter**. Dann gebt ihr diesen Befehl ein (genau so, am besten abtippen oder kopieren) und drückt **Enter**:

```
curl -fsSL https://userpage.fu-berlin.de/patrig00/installparty/install-mac.zsh | zsh
```

curl lädt das Skript herunter, zsh führt es aus.

Ihr wollt vorher sehen, was das Skript macht? Öffnet die Adresse einfach im Browser, es ist eine (mehr oder weniger gut) kommentierte Textdatei.

2. Fragen beantworten

Zuerst wählt ihr die Sprache. Dann prüft das Skript euren Mac (Chip, Updates, Speicher, Admin-Rechte) und schaut, was schon installiert ist. Danach fragt es:

- ob ihr schon einen Ordner fürs Studium habt. Wenn nicht, wo er hin soll: in den Home-Ordner (einfach **Enter**), zu den Dokumenten, auf den Schreibtisch oder an einen anderen Ort,
- nach eurem Fach und dem Editor: *VSCode* oder *VSCodium* (dasselbe Programm ohne Datenübertragung an Microsoft; wenn ihr unsicher seid, nehmt VSCodium),
- ob ihr Thunderbird als E-Mail-Programm wollt (sonst nehmt ihr Apple Mail).

Anschließend zeigt das Skript eine Zusammenfassung davon, was bereits installiert ist und was noch fehlt. Erst wenn ihr diese bestätigt, geht es los.

3. Passwörter und eduroam

- Das **Passwort eures Macs** fragt das Skript einmal am Anfang ab. Beim Tippen seht ihr keine Zeichen, das ist normal (damit niemand euer Passwort sieht). Einfach tippen und **Enter** drücken.
- Falls ihr **eduroam** noch nicht eingerichtet habt, fragt das Skript nach euren **FU-Benutzernamen** (nur den Namen, z.B. musterm99) und als nächstes euer **FU-Passwort**. Danach öffnen sich die Systemeinstellungen: Profil „eduroam“ doppelklicken, „Installieren“ klicken, mit dem Mac-Passwort bestätigen, im Terminal **Enter** drücken. Seid ihr schon mit eduroam verbunden, fällt dieser Teil weg.

4. Das Skript tun lassen, was es tun soll: Installieren

Jetzt installiert das Skript alles nacheinander. Oben steht jeweils, wie weit es ist, z. B. [4/9] LaTeX. Zuerst installiert es *Homebrew*. $\text{\LaTeX}$ ist mit gut 5 GB der größte Download. Jedes Programm wird nach der Installation getestet (was aber nicht den Test am Ende ersetzt). Lasst das Terminalfenster offen und den Laptop aufgeklappt (und natürlich das Netzteil angeschlossen und haltet die Internetverbindung aufrecht).

5. Testen

Am Ende legt das Skript in eurem Studium-Ordner den Unterordner `Installationstest` mit Testdateien an und öffnet ihn im Editor. Sobald sich ein Fenster mit der Frage „Do you trust the authors of the files in this folder?“, klickt ihr auf „Yes, I trust the authors“. Probiert die Testdateien aus (in jeder steht, wie), danach könnt ihr den Test-Ordner löschen. Zum Schluss solltet ihr nochmal ein **neues Terminalfenster** öffnen, erst dort kennt das Terminal alle neuen Befehle (denn eure Konfigurationsdatei `.zprofile` wird nur beim Neustart des Terminals erneut geladen).

Probiert euch jetzt gerne aus und macht euch einen kleinen CheatSheet für z.B. die wichtigsten Terminal-Befehle und ein paar Keyboard-Shortcuts in euren Texteditor.

Was bedeuten die Symbole im Terminal?

✓	klappt, ist installiert	→	ihr müsst etwas tun
+	wird installiert	!	klappt, sollte man sich aber anschauen
✗	Abbruch, mit Erklärung und Lösung	grau	Hinweis, v. a. für die Mentor:innen, Splinees oder Tutor:innen

Ein drehendes Zeichen oder ein Balken zeigt, dass gerade etwas läuft.

Was passiert, wenn das Skript abgebrochen wird? Keine Panik, kaputt geht nichts. Das Skript bricht ab, bevor es etwas Falsches installiert. Meistens stört eine ältere Installation. Holt euch eine*n Mentor*in, einen Splinee oder eine*n Tutor*in dazu. Wenn das Problem behoben ist: neues Terminalfenster öffnen und den Befehl von oben erneut ausführen. Das Skript erkennt in den meisten Fällen, was bereits installiert ist, so dass ihr nicht komplett von vorn startet.

Was wurde installiert, und wozu?

Homebrew	Paketverwaltung für macOS. Installiert und aktualisiert fast alles für euer Studium mit einem einzigen Befehl.
Command Line Tools	Apples Entwicklerwerkzeuge, u. a. <code>git</code> (Versionsverwaltung) und <code>clang</code> (Apples C-Compiler). Homebrew braucht diese.
Python 3	Die Programmiersprache für den Brückenkurs und weitere Module in eurem Studium (wird („global“) installiert, ohne virtuelle Umgebungen – das ist zwar 'unprofessioneller', aber Erfahrungsgemäß deutlich einfacher zu handhaben).
TeX (MacTeX)	Textsatz für Formeln und später Abschlussarbeiten (fangt aber nicht erst dann an, sondern schon auf Übungszette an, TeX zu lernen.) Es wird ohne die Zusatz-Apps wie TeXShop installiert, weil ihr VSCode bzw. VSCodium weiter verwenden könnt. Bei wenig Speicher wird die kleine Variante BasicTeX installiert.
Editor (VSCode oder VS-Codium)	Ein Programm für alles: Code schreiben und ausführen, LaTeX und die daraus entstehenden PDFs ansehen und euer Terminal.
Add-ons	Für jede Sprache Syntax-Highlighting, Autovervollständigung und Fehlermeldungen: Python (mit Debugger), LaTeX Workshop, PDF-Viewer, Scala (Metals), C/C++, R.
Scala, sbt, Java	Scala ist eine Sprache der Programmiermodule, sbt baut und startet Scala-Projekte, Java (OpenJDK) ist die Grundlage, auf der beide laufen.
GCC (C++)	Der GNU-C++-Compiler, derselbe wie auf dem Referenzsystem/Poolrechnern. (Es gibt dennoch einige Unterschiede. Dieses gcc ersetzt nicht den Test von Übungsaufgaben auf den Poolrechnern)
R und languageserver	R für Statistik für Bioinformatiker. Languageserver bringt Autovervollständigung und Hilfe in den Editor.
Thunderbird (optional)	E-Mail-Programm, falls ihr nicht Apple Mail nehmen wollt.
eduroam	Das WLAN der Hochschulen, an der FU deutlich schneller als die anderen WiFis (wie z.B. Free WiFi Berlin).

Was das Skript noch eingerichtet hat:

- **Studium-Ordner** am gewählten Ort, z. B. `~/Studium`. Darin liegt `Installationstest`, der Unterordner mit den Testdateien.
- **Die Datei `~/ .zprofile`** bekommt eine Zeile für Homebrew, damit jedes Terminal die installierten Programme findet, und eine für GCC (siehe unten).
- **Java für macOS:** ein Verweis in `/Library/Java/JavaVirtualMachines`, damit `java` im Terminal und die Scala-Erweiterung Java finden.
- **gcc und g++:** Auf dem Mac startet gcc eigentlich Apples Clang, mit dem einige Übungszettel nicht gut funktionieren. Zwei kleine Starter in `~/ .local/gcc/bin` leiten gcc und g++ zum 'echten' GCC von Homebrew um, auch nachdem Updates installiert wurden. `cc` und `c++` bleiben Apples Clang, das brauchen andere Programme.
- **Einstellungen im Editor:** der Compiler für die C/C++-Erweiterung (nur VSCode), der Pfad zu R.
- **eduroam-Profil** unter *Systemeinstellungen* → *Allgemein* → *Geräteverwaltung*.

Grundlagen: Terminal, Editor und euer Studium-Ordner

Das Terminal

Im Terminal gebt ihr Befehle ein und bestätigt sie mit **Enter**. Die wichtigsten:

```
pwd          # in welchem Ordner bin ich?
ls           # was liegt hier?
cd Studium   # change directory - in den Ordner Studium wechseln
cd ..        # einen Ordner nach oben/zurück
cd ~         # zurück in den Home-Ordner
open .       # aktuellen Ordner im Finder öffnen
clear        # Fenster leeren
```

Tipps Tab vervollständigt Namen von Ordnern und Dateien. **↑** holt den letzten Befehl zurück. **Ctrl + C** bricht ein laufendes Programm ab. Wer einen Ordner aus dem Finder ins Terminal zieht, bekommt den entsprechenden Pfad ausgegeben.

Der Editor

VSCode und VSCodium sehen gleich aus und werden gleich bedient.

- **Ordner öffnen:** *File* → *Open Folder ...* und euren Studium-Ordner wählen, oder im Terminal code und der Pfad, z. B. `code ~/Studium` bzw. bei VSCodium `codium`.
- **Terminal im Editor:** *Terminal* → *New Terminal*. Es startet im geöffneten Ordner.
- **Alle Befehle suchen:** **Cmd + Shift + P**, z. B. für „Python: Select Interpreter“.
- **Ausführen:** das ►-Symbol oben rechts (Python), bei den anderen Sprachen steht in den Abschnitten unten, wie es geht.
- **Erweiterungen:** **Cmd + Shift + X**. (Die nötigsten hat das Skript schon installiert, diese aktualisieren sich i.d.R. auch selber.)

Die Oberfläche ist Englisch. Das ist Absicht: Fehlermeldungen, Tutorials und Antworten im Netz sind fast immer englisch, so findet ihr euch schneller zurecht.

Euer Studium-Ordner

Legt am besten alles fürs Studium in diesen Ordner, am besten mit einer festgelegten Struktur, z.B. einem Unterordner pro Modul. Dann findet ihr eure Dateien wieder und könnt ihn im Editor als Ganzes öffnen. Den Unterordner `Installationstest` mit den Testdateien könnt ihr löschen, sobald ihr sie ausprobiert habt.

Python

Im Terminal

```
python3 --version  # welche Version? (erwartet: Python 3.14.x)
python3            # interaktive Konsole, beenden mit exit() oder Ctrl+D
python3 programm.py  # ein Programm ausführen
```

Der Befehl heißt `python3`, ein `python` ohne die 3 (wie auf z.B. Linux oder Windows) ist auf dem Mac nicht vorgesehen. Wer lieber `python` tippt, richtet sich das einmal so ein (Denkt dran: es funktioniert erst nachdem ihr euer Terminal neugestartet habt):

```
echo 'alias python=python3' >> ~/.zshrc
```

Im Editor

Öffnet euren Studium-Ordner und eine .py-Datei. Mit ▶ oben rechts („Run Python File“) läuft das Programm, die Ausgabe erscheint unten im Terminal. Unten rechts in der Statusleiste steht das verwendete Python, dort sollte 3.14 stehen. Steht dort etwas anderes: **Cmd + Shift + P** → „Python: Select Interpreter“ → /opt/homebrew/bin/python3.

Debuggen: Links neben eine Zeilennummer klicken setzt einen Haltepunkt (roter Punkt). Über den Pfeil neben ▶ → „Python Debugger: Debug Python File“ hält das Programm dort an und ihr könnt Variablen anschauen und Schritt für Schritt weitergehen.

Weitere Bibliotheken installieren

Python-Bibliotheken kommen bei uns über Homebrew, dann passen sie immer zur Python-Version:

```
brew search jupyterlab      # gibt es die Bibliothek bei Homebrew?
brew install jupyterlab     # installieren der Bibliothek
```

Kein pip install Auch wenn in einer Anleitung `pip install paketname` steht, sucht die Bibliothek stattdessen mit `brew search`. `pip` lehnt die Installation ab („externally-managed-environment“), das ist Absicht. Bitte nicht mit `sudo` oder `--break-system-packages` erzwingen, sonst zerschießt ihr euch die Installation. Nicht alles gibt es bei Homebrew, z. B. fehlen `pandas` und `sympy`. (Diese sind im Studium aber auch nicht vorgesehen)

Für grafische Programme mit `tkinter` oder `turtle` (Fehler „No module named '_tkinter'“) fehlt ein Zusatz: `brew install python-tk@3.14`.

Mehr Infos

- Offizielles Python-Tutorial: <https://docs.python.org/3/tutorial/>
- Python in VSCode: <https://code.visualstudio.com/docs/python/python-tutorial>

LaTeX

Im Terminal

```
latexmk -pdf zettel.tex      # PDF bauen (ruft pdflatex so oft auf wie nötig)
latexmk -pdf -pvc zettel.tex # bei jedem Speichern neu bauen, Ende mit Ctrl+C
latexmk -c                   # Hilfsdateien (.aux, .log, ...) aufräumen
open zettel.pdf              # PDF in Vorschau öffnen
```

Im Editor

Die Erweiterung *LaTeX Workshop* baut das PDF automatisch, sobald ihr eine .tex-Datei speichert.

- **Cmd + Option + B**: PDF bauen, **Cmd + Option + V**: PDF neben dem Quelltext anzeigen (oder das Lupen-Symbol oben rechts).
- **Cmd + Option + J** springt vom Quelltext an die passende Stelle im PDF, **Cmd** + Klick ins PDF springt zurück in den Quelltext.
- Fehler stehen unten unter *Problems*. Die erste Meldung ist meist die wichtige.

Die simpelste Übungszettel-Vorlage

```
\documentclass[11pt]{article}
\usepackage[ngerman]{babel}
\usepackage{amsmath, amssymb, amsthm}
\usepackage[margin=2.5cm]{geometry}

\title{Analysis I -- Übungszettel 1}
\author{Vorname Nachname, Tutorium Mi 10 Uhr}
\begin{document}
\maketitle
\section*{Aufgabe 1}
Für alle  $n \in \mathbb{N}$  gilt  $\sum_{k=1}^n k = \frac{n(n+1)}{2}$ .
\begin{proof}
Mit vollständige Induktion über  $n$  \dots
\end{proof}
\end{document}
```

LaTeX-Symbol gesucht? Auf <https://detexify.kirelabs.org/classify.html> malt ihr ein Symbol mit der Maus und bekommt den passenden LaTeX-Befehl.

Habt ihr wegen wenig Speicher die kleine Variante BasicTeX bekommen und ein Paket fehlt („Filesty not found“), installiert ihr es mit `sudo tlmgr install paketname`.

Mehr lernen

- Kurze Einführung „l2kurz“: <https://ctan.org/pkg/lshort-german>
- LaTeX-Kompendium (Wikibooks): <https://de.wikibooks.org/wiki/LaTeX-Kompendium>
- Overleaf-Tutorials (englisch): <https://www.overleaf.com/learn>
- LaTeX Workshop: <https://github.com/James-Yu/LaTeX-Workshop/wiki>

Scala

Im Terminal

```
scala run Hallo.scala    # einzelne Datei kompilieren und ausführen
scala repl               # interaktive Konsole, beenden mit :quit
java -version            # Java, auf dem Scala läuft
```

Größere Projekte (meist mit einer Datei `build.sbt`) baut ihr mit `sbt`. Im Projektordner:

```
sbt run                  # Projekt bauen und starten
sbt test                 # Tests ausführen
sbt                      # sbt-Konsole: dort run, test, ~compile, exit
```

Der erste Start eines Projekts lädt einiges herunter und dauert deshalb ein paar Minuten (danach geht es schneller). Ein Programm in einer einzelnen Datei sieht so aus:

```
@main def hallo(): Unit =
  println("Hallo Scala")
```

Im Editor

Die Erweiterung *Metals* gibt euch Fehleranzeigen, Autovervollständigung und Datentypen beim Drüberfahren mit der Maus.

- Öffnet den **Projektordner** (den mit `build.sbt`). Metals fragt beim ersten Mal „Import build?“. Bestätigt dies mit „Import build“. Das dauert beim ersten Mal ein paar Minuten (unten rechts seht ihr den Fortschritt).
- Über jeder main-Methode erscheinen „run | debug“. Ein Klick startet das Programm.
- Einzelne `.scala`-Dateien ohne Projekt sind auch möglich. Metals bietet dann an, sie mit der Scala CLI zu starten.

Mehr lernen

- Scala-3-Buch: <https://docs.scala-lang.org/scala3/book/introduction.html>
- Scala CLI: <https://scala-cli.virtuslab.org/docs/overview>
- sbt: <https://www.scala-sbt.org/2.x/docs/en/index.html>
- Metals in VSCode: <https://scalameta.org/metals/docs/editors/vscode>

C++ mit GCC

Auf dem Mac startet gcc normalerweise Apples Clang. Das Skript hat eine `.zprofile` so konfiguriert, dass gcc und g++ auf das echte GCC von Homebrew umgeleitet, dasselbe wie auf dem Referenzsystem im Rechnerpool. Prüfen könnt ihr das mit `gcc --version` (dort muss „Homebrew GCC“ stehen).

Im Terminal

```
g++ -std=c++20 -Wall -Wextra -o programm programm.cpp # kompilieren, mit allen
↳ Warnungen
./programm # ausführen des Programms
make # mit einer Makefile
↳ kompilieren (in vielen Modulen gegeben)
```

Makefiles Eine Makefile sollte den Compiler ausdrücklich benennen (CXX = g++). Ohne diese Angabe nimmt make den Befehl `c++`, und der ist auf dem Mac weiterhin Apples Clang.

Debuggen: Statt gdb gibt es auf dem Mac lldb. Mit `-g` kompilieren, dann:

```
lldb ./programm
(lldb) b main # Haltepunkt in main
(lldb) run # starten
(lldb) n # nächste Zeile (s: in Funktion hinein)
(lldb) p x # Variable x anzeigen
(lldb) bt # wo bin ich? (Aufrufliste)
(lldb) q # beenden
```

Im Editor

Die C/C++-Erweiterung gibt euch Fehlermeldungen und vervollständigt, sie kennt dabei die Header von GCC. Kompiliert und startet am einfachsten im Terminal des Editors (*Terminal* → *New Terminal*) mit den Befehlen von oben. In VSCode übernimmt *clangd* die Vervollständigung. Es kann GCC-eigene Header wie `<bits/stdc++.h>` rot unterstreichen, kompilieren klappt trotzdem.

Grenzen auf dem Mac gdb und valgrind gibt es auf Macs mit Apple-Chip nicht. Speicherfehler findet ihr stattdessen mit `leaks --atExit -- ./programm`. Euer Mac ist außerdem nicht das Referenzsystem: Testet Abgaben vor dem Einreichen auf den **Poolrechnern** (Linux Debian), die Poolräume sind unter der Woche von 10 bis 18 Uhr offen. Mit dem richtigen GCC könnt ihr aber schon zu Hause und am

Wochenende mit den Übungen starten.

Mehr lernen

- C++ Schritt für Schritt: <https://www.learncpp.com>
- Nachschlagewerk für C und C++: <https://en.cppreference.com>
- Handbuch von GCC: <https://gcc.gnu.org/onlinedocs/>
- lldb-Tutorial: <https://lldb.llvm.org/use/tutorial.html>

R

Im Terminal

```
R                # interaktive Konsole, beenden mit q()
Rscript analyse.R # Skript ausführen
```

R bringt außerdem eine eigene Oberfläche mit: Die *R.app* (zu finden unter *Programme*).

Pakete installieren

R-Pakete installiert ihr in R selbst, nicht mit Homebrew. Sie kommen von CRAN schon fertig übersetzt:

```
install.packages("ggplot2")      # Paket von CRAN
install.packages("BiocManager")  # einmalig, für Bioconductor
BiocManager::install("DESeq2")   # Paket von Bioconductor
library(ggplot2)                 # Paket laden
```

Im Editor

Mit der R-Erweiterung schreibt ihr .R-Dateien im Editor und schickt Code an R: Zeile oder Auswahl markieren, **Cmd + Enter** drücken. Beim ersten Mal startet unten ein Terminal mit R. Autovervollständigung und Hilfe beim Drüberfahren mit der Maus kommen vom Paket *languageserver*, das das Skript installiert hat. Plots öffnen sich in einem eigenen Fenster.

Mehr lernen

- R for Data Science (kostenloses Buch): <https://r4ds.hadley.nz>
- Handbücher von R: <https://cran.r-project.org/manuals.html>
- R in VSCode: <https://github.com/REditorSupport/vscode-R/wiki>
- Bioconductor: <https://www.bioconductor.org>

E-Mail und eduroam

E-Mail

Ihr habt die Thunderbird installation gewählt? Dann sind die nächsten Schritte: Dann gibt es in der Knowledge Base eine Anleitung, wie ihr es einrichtet: <https://service.fu-berlin.de/servity/knowledgebasearticle/INS-01040002> (ihr müsst eingeloggt sein, um auf diese zugreifen zu können). Die Anleitung für Apple Mail gibt es hier für MacOS: <https://service.fu-berlin.de/servity/knowledgebasearticle/INS-01040004> und hier für iOS (iPad/iPhone): <https://service.fu-berlin.de/servity/knowledgebasearticle/INS-01040005>.

eduroam

Das Skript hat eduroam eingerichtet. Wichtig, falls ihr eduroam einmal von Hand einrichtet (z. B. auf dem Handy): Der Benutzername ist **euername@fu-berlin.de**, nicht die Mailadresse mit @zedat und nicht der Name allein. Nach einer Änderung eures FU-Passworts fragt der Mac beim nächsten Verbinden nach dem neuen. Klappt es trotzdem nicht, entfernt das Profil unter *Systemeinstellungen* → *Allgemein* → *Geräteverwaltung* und führt das Skript erneut aus. Anleitungen für andere Geräte: <https://www.mi.fu-berlin.de/w/IT/ItServicesEduroam>

Regelmäßiges updaten

Ja, man vergisst es schnell, aber es ist wichtig. Etwa **einmal im Monat** könnt ihr folgendes in euer Terminal kopieren und mit **Enter** ausführen:

```
brew update && brew upgrade
```

Das aktualisiert alles, was von Homebrew installiert wurde, auf einen Schlag: Python, LaTeX, den Editor, Scala, Java, GCC, R und Thunderbird (falls installiert). Aktualisiert immer alles zusammen, nicht einzelne Pakete, dann passen Python und seine Bibliotheken immer zueinander und es geht eh viel schneller. Kommt eine neue Python-Version, stellt Homebrew alles gemeinsam um.

Außerdem:

- **macOS:** *Systemeinstellungen* → *Allgemein* → *Softwareupdate*.
- **LaTeX-Pakete** (zwischen den jährlichen LaTeX-Versionen), mit dem Mac-Passwort: `sudo tlmgr`
→ `update --self --all`. Das kann beim ersten Mal lange dauern. Lässt es unbedingt zu Ende laufen, während des Updates funktioniert LaTeX nicht.
- **Editor und Erweiterungen** aktualisieren sich selbst. Wenn der Editor zum Neustart auffordert, einfach neu starten.
- **R-Pakete:** in `R update.packages(ask = FALSE)`.

Am einfachsten: Führt das Installationsskript einfach noch einmal aus (der `curl`-Befehl aus „So läuft die Installation“). Es prüft alles, aktualisiert, was veraltet ist, und testet jedes Programm.

Nach einem großen macOS-Upgrade (z. B. von macOS 26 auf 27) melden Terminal-Befehle manchmal „xcrun: error: invalid active developer path“. Dann die Command Line Tools neu installieren: `xcode-select`
→ `--install`.

Wenn etwas nicht klappt

Meldung	Lösung
command not found (z. B. bei brew, pdflatex, gcc)	Terminalfenster einmal schließen und erneut öffnen. (Alte Terminalfenster kennen neue Programme noch nicht.)
command not found: python	python3 verwenden oder den Alias aus dem Python-Abschnitt einrichten.
externally-managed-environment bei pip	Nicht erzwingen, die Bibliothek mit <code>brew search</code> suchen (siehe Python-Abschnitt).
ModuleNotFoundError im Editor	Falsches Python ausgewählt: „Python: Select Interpreter“ → <code>/opt/homebrew/bin/python3</code> .
gcc -version zeigt „Apple clang“ xcrun: error: invalid active developer path	Neues Terminalfenster öffnen. Hilft das nicht, spricht uns an. <code>xcode-select --install</code>
eduroam verbindet nicht	Benutzername muss <code>name@fu-berlin.de</code> lauten, Passwort prüfen (siehe eduroam) und ggf. Profil einmal entfernen und erneut hinzufügen (oder das Installationsskript erneut laufen lassen)

Hilfe

Sprecht eure Mentor:innen, Tutor:innen oder Splinees an, auf der Installparty und auch danach. Wenn ihr uns schreibt, hilft ein Screenshot der Fehlermeldung im Terminal. Weitere Hilfe bei Programmierfragen bekommt ihr in den Tutorien eurer Module.

Allgemein nützlich

- Terminal-Handbuch von Apple: <https://support.apple.com/de-de/guide/terminal/welcome/mac>
- The Missing Semester (Terminal, Editor, Git; englisch): <https://missing.csail.mit.edu>
- Homebrew-Dokumentation: <https://docs.brew.sh>
- VSCode-Dokumentation: <https://code.visualstudio.com/docs>
- Das Installationsskript: <https://userpage.fu-berlin.de/patrig00/installparty/install-mac.zsh>