Installation & Setup

Grundlagen Operations Research

Auf dieser Seite richten Sie alles ein, was Sie brauchen, um die Julia-Beispiele des Kurses selbst auszuführen: die Programmiersprache **Julia**, den Editor **VS Code** und die drei Kurs-Pakete. Der ganze Weg dauert etwa 15 Minuten. Danach läuft jede Code-Zelle dieses Kurses auch auf Ihrem Rechner.

> ℹ Brauche ich das zwingend?
>
> Nein. Sie können alle Vorlesungen und Übungen auch ohne eigene Installation lesen: sämtliche Ergebnisse sind auf den Seiten bereits ausgeführt. Für die Übungsaufgaben mit Selbstkontrolle (und für die Klausurvorbereitung) lohnt sich die Installation aber sehr: Optimierung lernt man durch Ausprobieren.

Schritt 1: Julia installieren

Wir empfehlen die Installation über **juliaup**, den offiziellen Versionsmanager. Er installiert immer die aktuelle stabile Version und hält sie automatisch aktuell.

Windows (im Terminal bzw. der PowerShell):

```
winget install julia -s msstore
```

macOS und Linux (im Terminal):

```
curl -fsSL https://install.julialang.org | sh
```

Alternativ finden Sie fertige Installationspakete auf julialang.org/downloads.

> 💡 Tipp
>
> Falls das Installationsprogramm fragt, ob es etwas zu Ihrem „PATH" hinzufügen darf: **ja, unbedingt**. Sonst finden Terminal und Editor die Julia-Installation später nicht.

Prüfen Sie die Installation, indem Sie ein **neues** Terminal öffnen und

```
julia --version
```

eingeben. Erscheint eine Versionsnummer (der Kurs ist mit Julia 1.12 getestet), hat alles geklappt.

Schritt 2: VS Code installieren

Als Editor verwenden wir im Kurs **Visual Studio Code**:

1. Laden Sie VS Code von code.visualstudio.com herunter und installieren Sie es.
2. Starten Sie VS Code und öffnen Sie links in der Seitenleiste den Bereich **Extensions** (Tastenkürzel `Strg+Umschalt+X` bzw. `Cmd+Umschalt+X`).
3. Suchen Sie nach **„Julia"** und installieren Sie die Erweiterung *Julia (Julia Language Support)*.

> Tipp
>
> Wer Wert auf eine Variante ohne Telemetrie legt, kann alternativ VSCodium verwenden. Die Julia-Erweiterung funktioniert dort genauso.

Schritt 3: Kurs-Pakete installieren

Für diesen Kurs brauchen Sie drei Pakete: **JuMP** (Modellierungssprache), **HiGHS** (Open-Source-Solver) und **Plots** (Grafiken). Öffnen Sie ein Terminal, starten Sie mit `julia` die interaktive Konsole (die *REPL*) und tippen Sie dort **eine schließende eckige Klammer** `]`. Die Eingabezeile wechselt in den blauen Paketmodus `pkg>`. Installieren Sie dann:

```
] add JuMP HiGHS Plots
```

Die Installation lädt einmalig einige hundert Megabyte herunter und dauert ein paar Minuten. Mit der Rücktaste (`Backspace`) verlassen Sie den Paketmodus wieder, mit `exit()` die REPL.

> Optional: Jupyter-Notebooks (.ipynb)
>
> Zu jeder Übung können Sie neben der Julia-Datei (Button **Julia**) auch ein Jupyter-Notebook herunterladen (Button **Jupyter**). Wenn Sie Notebooks direkt in VS Code ausführen möchten, installieren Sie zusätzlich das Paket `IJulia` (`] add IJulia`) sowie die VS-Code-Erweiterung **Jupyter**. Für den Kurs ist das optional. `.jl`-Dateien reichen völlig aus.

Schritt 4: Testlauf

Legen Sie in VS Code eine neue Datei an (z. B. `test.jl`, am besten in einem eigenen Ordner für diesen Kurs) mit folgendem Inhalt:

```
println("Hallo, Operations Research!")
```

```
Hallo, Operations Research!
```

Führen Sie die Datei aus: entweder über das Dreieck („Run") oben rechts oder zeilenweise mit `Strg+Enter` (bzw. `Cmd+Enter`, macOS). Unten öffnet sich eine Julia-REPL und gibt den Text aus. Wenn Sie das sehen, ist Ihr Setup vollständig.

> ⚠ Probleme beim Testlauf?
>
> - **„Julia executable not found":** VS Code findet die Installation nicht. Öffnen Sie die Einstellungen (`Strg+,`), suchen Sie nach `julia.executablePath` und tragen Sie den Pfad zur Julia-Programmdatei ein (Terminal: `which julia` bzw. Windows: `where julia`). Danach VS Code neu starten.
> - **Der erste Start dauert lange:** Das ist normal: Julia kompiliert beim ersten Ausführen. Ab dem zweiten Lauf innerhalb derselben REPL geht es deutlich schneller. Lassen Sie die REPL deshalb geöffnet, statt sie nach jedem Lauf zu schließen.
> - **`Package JuMP not found`:** Schritt 3 wiederholen und darauf achten, dass die Installation ohne Fehlermeldung durchläuft.

Arbeitsdateien des Kurses

Auf jeder Übungsseite finden Sie rechts oben Download-Buttons: **Julia** lädt die Seite als ausführbare `.jl`-Datei, **Jupyter** als Notebook und **PDF** als Ausdruck. Laden Sie die Datei herunter, öffnen Sie sie in VS Code und arbeiten Sie die Aufgaben direkt darin durch. Die Selbstkontroll-Blöcke am Ende jeder Aufgabe sagen Ihnen, ob Ihre Lösung stimmt.

Best Practices und Tipps

Diese Tipps gehen über die reine Installation hinaus und ersparen Ihnen die häufigsten Anfangshürden. Zwingend sind sie für den Kurs nicht.

Reproduzierbare Paketumgebungen

Die Installation aus Schritt 3 landet in der globalen Umgebung und genügt für alle Kursaufgaben. Sobald Sie eigene, größere Projekte anlegen, sind **Projektumgebungen** die bessere Wahl: Jedes Projekt erhält eine eigene `Project.toml` (welche Pakete) und `Manifest.toml` (welche exakten Versionen). Wechseln Sie dazu in den Projektordner und aktivieren Sie ihn im Paketmodus:

```
] activate .
] add JuMP HiGHS Plots
```

Geben Sie ein solches Projekt weiter oder holen es von einer anderen Person, stellt `] instantiate` mit einem einzigen Befehl exakt dieselben Paketversionen wieder her. So läuft Ihr Code auf jedem Rechner identisch.

Julia und Pakete aktuell halten

`juliaup` aktualisiert Julia selbst:

```
juliaup update
```

Die installierten Pakete bringen Sie im Paketmodus auf den neuesten Stand:

```
] update
```

Zügig arbeiten

Julia kompiliert Ihren Code beim ersten Ausführen, deshalb dauert der allererste Aufruf (etwa der erste Plot) spürbar länger. Danach geht es schnell. Lassen Sie die REPL deshalb geöffnet und führen Sie Ihre Datei mehrfach in derselben Sitzung aus, statt Julia jedes Mal neu zu starten.

Typische Stolperfallen

> Tipp
>
> - **Indizes beginnen bei 1**, nicht bei 0: `v[1]` ist das erste Element.
> - **Elementweise Operationen brauchen einen Punkt**: `preise .* mengen` multipliziert Eintrag für Eintrag, `value.(X)` liest alle Variablenwerte eines Vektors aus.
> - **`/` liefert immer eine Kommazahl**: `6 / 2` ergibt `3.0`. Ganzzahlig teilen Sie mit `div(6, 2)`.
> - **Zuweisung `=` und Vergleich `==` nicht verwechseln**: In Gleichungs-Nebenbedingungen schreiben Sie `==`, nicht `=`.

Mehr dazu finden Sie unter Erste Schritte und in den Cheatsheets.

Wie geht es weiter?

- Erste Schritte: die Julia-Grundlagen, die dieser Kurs voraussetzt (Variablen, Vektoren, Schleifen, Funktionen).
- Pakete & JuMP: wie ein Optimierungsmodell in JuMP aufgebaut, gelöst und ausgewertet wird.
- Zum Nachschlagen: Cheatsheet Julia und Cheatsheet JuMP.

Bibliografie