Anhang A
Julia und Pluto

Markus Lippitz
26. September 2021

Wir benutzen in dieser Veranstaltung die Programmiersprache *Julia*[1] für graphische Veranschaulichungen und numerische 'Experimente'. Ich bin überzeugt, erst wenn man einen Computer überreden kann, etwas zu tun, ein Model darzustellen, einen Wert auszurechnen, erst dann hat man es wirklich verstanden. Vorher hat man nur die ganzen Probleme noch nicht gesehen.

Man kann Julia mit verschiedenen Benutzeroberflächen verwenden. Wir benutzen *Pluto*.[2]

[1] https://julialang.org

[2] https://github.com/fonsp/Pluto.jl

Julia

Julia ist eine Programmiersprache, die für Numerik und wissenschaftliches Rechnen entwickelt wurde. Sie ist ein Mittelding zwischen Matlab, Python und R. Aus meiner Sicht übernimmt sie jeweils das Beste aus diesen Welten und eignet sich so gerade für Einsteiger. Wir werden im Laufe des Semesters verschiedene Beispiel-Skripte zusammen besprechen, und es wird auch numerische Übungsaufgaben geben.

Ein Beispiel

Lassen Sie uns zunächst ein einfaches Beispiel betrachten.

```
using Plots
x = range(0, 2 * pi; length=100)
plot(x, sin.(x); label="ein Sinus")
```

Für manche Dinge benötigt man Bibliotheken, die man mit `using` laden kann. Halten sie sich bei der Auswahl der Bibliotheken zunächst an die Beispiele, die ich zeige.

Dann definieren wir eine Variable `x` (einfach durch benutzen) als äquidistanter 'Zahlenstrang' zwischen 0 und 2π mit 100 Werten. Funktionen wie `range` haben immer benötigte Parameter, die über ihre Position in der Parameterliste definiert sind (hier: Anfangs- und End-Wert), sowie weitere optionale. Diese folgen nach einem Semikolon in der Form `<Parameter>=<Wert>`.

Schließlich zeichnen wir die Sinus-Funktion über diesen Wertebereich. Beachten sie den Punkt in `sin.(x)`. Er bedeutet 'wende `sin` auf alle Elemente von `x` an'. Das ist sehr praktisch.

Informationsquellen

Aktuell ist die Version 1.6.2. Mit der Version 1.0 hat sich einiges geändert. Ignorieren sie Webseiten, die älter als 2 Jahre sind, bzw. die sich auf eine Version vor 1.0 beziehen.

Offizielle Dokumentation auf der website[3]. Oder fragen Sie google mit 'Julia' als Stichwort oder mit der Bibliothek / Funktion und angehängter Endung '.jl' .

Beispiele Julia by example[4], Think julia[5], Introduction to Computational Thinking[6]

Unterschiede Vergleich[7] mit Matlab, Python und anderen Sprachen. Und als Übersichtstabelle[8]

Cheat Sheets Allgemein[9] und für Plots[10]

[3] https://docs.julialang.org/en/v1/

[4] https://juliabyexample.helpmanual.io/

[5] https://benlauwens.github.io/ThinkJulia.jl/latest/book.html

[6] https://computationalthinking.mit.edu/Spring21/

[7] https://docs.julialang.org/en/v1/manual/noteworthy-differences/

[8] https://cheatsheets.quantecon.org/

[9] https://juliadocs.github.io/Julia-Cheat-Sheet/

[10] https://github.com/sswatson/cheatsheets/

Benutzeroberflächen

Es gibt verschiedene Möglichkeiten, wie man kürzere oder längere Programme in Julia schreiben kann. Hier eine Auswahl

Kommandozeile und Editor Man kann Julia interaktiv an der Kommandozeile (REPL, read-eval-print loop) benutzen. In einem externen Editor könnte man wiederholende Kommandos in Skript-Dateien schreiben.

IDE Das geht komfortabler mit einer integrierten Umgebung, beispielsweise Juno[11], oder einer Julia-Erweiterung [12] für Visual Studio Code. Das ist sicherlich die Herangehensweise bei größeren Projekten.

[11] https://junolab.org/

[12] https://www.julia-vscode.org/

Jupyter notebook Jupyter[13] setzt sich zusammen aus Julia, Python und R. Diese drei Sprachen kann man in einem Notebook-Format benutzen. Programmcode steht dabei in Zellen, die Ausgabe und auch beschreibender Text und Grafiken dazwischen. Das eignet sich besonders, wenn Rechnungen von Beschreibungen oder Gleichungen begleitet werden sollen, beispielsweise in (Praktikums-)Protokollen oder Übungsaufgaben.

[13] https://jupyter.org/

Mathematica hat ein ähnliches Zellen-Konzept. Ein Nachteil ist, dass Zellen den Zustand des Kernels in der Reihenfolge ihrer Ausführung beeinflussen. Die Reihenfolge muss aber nicht der in der Datei entsprechen; insbesondere ändert ein Löschen der Zellen den Kernel nicht. Das kann sehr verwirrend sein, oder man muss der Kernel oft neu starten.

Pluto Man kann auch in Pluto[14] Programmcode, Text und Grafik mischen. Das Zellen-Konzept von Pluto ist das aber von Excel, limitiert auf eine Excel-Spalte. Die Anordnung der Gleichungen in den Zellen spielt keine Rolle. Alles wird nach jeder Eingabe neu evaluiert. Eine Logik im Hintergrund sorgt dafür, dass nur unbedingt notwendige Berechnungen neu ausgeführt werden. Aus meiner Sicht sollte das für Anfänger intuitiv zu bedienen sein und für kleiner Projekte völlig ausreichen sein. *Wir benutzen Pluto als Oberfläche in dieser Veranstaltungen.*

[14] https://github.com/fonsp/Pluto.jl

Installation

Server von EP III Um Ihnen die ersten Schritte zu vereinfachen können sie den Jupyter & Pluto-Server[15] von EP III benutzen. Dazu müssen Sie innerhalb der Universität sein oder via VPN verbunden sein. Zugangsdaten erhalten Sie in der ersten Semesterwoche. Melden Sie sich mit diesen am Server an. Sie gelangen auf eine Juypter-Oberfläche, auf der sie beispielsweise Dateien auf dem Server verwalten könne. Über das Pluto-Icon starten sie eine Pluto-Oberfläche im web browser.

Gehen Sie bitte rücksichtsvoll mit diesem Server um. Seine Ressourcen sind eher begrenzt.

[15] http://jupyter.ep3.uni-bayreuth.de

Lokale Installation Insbesondere wenn Ihnen der EP III Server zu langsam wird sollten Sie Julia und Pluto lokal installieren. Eine gute Anleitung ist am MIT[16]. Kurzfassung: Julia vom website installieren, dann in Julia das Pluto-Paket installieren (`import Pkg; Pkg.add("Pluto")`) und aufrufen via `using Pluto; Pluto.run()`.

[16] https://computationalthinking.mit.edu/Spring21/installation/

Benutzung von Pluto

Eine schöne Einführung in Pluto (und Julia) gibt es auf der Pluto homepage[17], am MIT (hier[18] bzw eigentlich die ganze site) und am WIAS.[19]

- Shift-Enter führt eine Zelle aus
- Der Ausführungs-Optimierer verlangt, dass jede Zelle einen geschlossenen Block bildet. Also darf da nur ein Kommando stehe, oder mehrere müssen mit `begin` ... `end` geschachtelt werden.
- Jede Zelle hat nur eine Ausgabe, die der letzten Zeile. Die Ausgabe steht über der Zelle selbst.
- Pluto verwaltet Bibliotheken selbständig, einfach mit `using` benutzen, installiert wird automatisch.
- Pluto speichert automatisch alles. Man kann aber die Datei umbenennen / bewegen.

[17] https://github.com/fonsp/Pluto.jl/wiki

[18] https://computationalthinking.mit.edu/Spring21/basic_syntax/

[19] https://www.wias-berlin.de/people/fuhrmann/SciComp-WS2021/assets/nb01-first-contact-pluto.html