Seminararbeit

Julia zur Verarbeitung und Visualisierung von Daten

Autor: Theodor Schwarzrock

Lehrer: Herr Möller

Seminarkurs Informatik

Friedrich-Schiller-Gymnasium

08.04.2024

# 1 Einleitung

*„Visualization gives you answers to the questions you didn’t know you had.”  
- Ben Schneiderman*

Unser heutiges Zeitalter ist geprägt von Informationen und Daten. In fast jeder alltäglichen Situation, welche sich mit Technologie werden unzählbar viele Daten ausgetauscht, ausgewertet und wiederum ausgetauscht. Diese Einsen und Nullen sind jedoch nicht für den Menschen verständlich, selbst wenn sie in lesbare Rohdaten umgewandelt werden. Um eine Intuition zu diesen Datenmengen zu erschaffen, müssen sie anschaulich gemacht werden. Darum kümmert sich die *Data-Science*.

Es wurden schon etliche Programmiersprachen entwickelt, die sich dieser Aufgabe widmen. Unter anderem *Python*, *C++*, *MATLAB*, *R* oder *FOTRAN*. Die genannten Sprachen sind Giganten in der Data-Science. Vor allem Python etablierte sich im Laufe der Zeit als die Beliebteste der vielen Optionen. Vor etwa 12 Jahren wurde ein neuer Konkurrent veröffentlicht. Die Programmiersprache *Julia*. Sie wurde erschaffen, um die besten Eigenschaften ihrer Vorgänger ein einer Sprache zu vereinen.

Das Ziel dieser Seminararbeit ist es einen umfassenden Vergleich zwischen Julia und Python zu erschaffen, wodurch schließlich auf Julias Relevanz und Daseinsberechtigung geschlossen wird. In dem Vergleich steht die Data-Science und darin vor allem die Visualisierung von Daten im Vorschein. So werden die beiden Programmiersprachen nicht nur auf Syntax, Semantik und weitere technische Details untersucht, sondern auch anhand eines praktischen Teils veranschaulicht. Der Fokus liegt währenddessen jedoch durchgehend auf Julia. In dieser Seminararbeit soll Python *nicht* bis auf das letzte Detail analysiert werden.

Die Relevanz der Forschung liegt im Innovationspotenzial von Julia. Für Entwickler und Datenwissenschaftler, welche stets nach effizienteren Werkzeugen suchen, ist ein Vergleich der vielen Möglichkeiten von großer Bedeutung. So gibt es zudem praktische Anwendungsbereiche dieser Untersuchung.

# 2 Die Programmiersprache Julia

## 2.1 Entstehung und Geschichte

Jeff Bezanson, Alan Edelman, Viral B. Shah und Stefan Karpinski hatten in den 2000ern ihr Informatikstudium beendet und verfolgten Karrieren in der Informatik. Sie waren in Sprachen wie *MATLAB*, *Python*, *Ruby*, *R* und *C* erfahren und sich deren jeweiligen Stärken und Schwächen bewusst. Jedoch waren die studierten Informatiker nicht zufrieden. Die verschiedenen Programmiersprachen deckten zwar gegenseitig ihre Schwächen ab, aber es gab nie eine Codiersprache, welche alle Vorteile mit möglichst wenig Nachteilen gleich vereinte. Die Gruppe tat sich somit mit dem Ziel zusammen, eine Sprache zu kreieren, die Leistung, Simplizität, Effizienz und Geschwindigkeit bietet.

Im Jahr 2009 begann schließlich die Entwicklung von Julia. Der Name hat keine Bedeutung. Bezanson wollte lediglich einen kurzen, prägnanten Namen. „Julia“ gefiel dem Team schlicht. Den Entwicklern war es wichtig, dass Julia Open Source ist und unter einer liberalen Lizenz veröffentlicht wird. Auf Benutzer-Ebene sollte Julia einfach zu erlernen sein, aber auch erfahrene Programmierer zufriedenstellen. Zudem war Interaktivität trotz Kompilierung ein sehr wichtiger Anhaltspunkt. Auf technischer Ebene sollte Julia so schnell sein wie *C*, aber dynamisch wie *Ruby*, eine mathematische Notation besitzen, ähnlich wie *MATLAB*, für die allgemeine Programmierung nutzbar sein, wie *Python* aber gleichzeitig auch für das wissenschaftliche Programmieren geeignet sein.

Das Team strebte außerdem grundlegende Funktionen an. Einfache Skalarschleifen sollen in Maschinencode kompiliert werden, um Effizienz zu steigern. Zudem soll eine Skalierung von Einzelprozessor-Berechnungen auf verteile Berechnungen über viele Maschinen hinweg möglich sein. So werden die Rechenleistung und Kapazität erheblich gesteigert. Um die gewollte Nutzerfreundlichkeit und Dynamik zu erreichen wurde großer Wert auf Typenflexibilität gelegt. Sie werden also nur dann spezifiziert, wenn es wirklich notwendig ist. „We are greedy: we want more.“ [[1]](#footnote-1) schrieb das Team passend in einem Artikel über die Schaffung von Julia.

Am 14. Februar 2012 wurde die erste Version von Julia schließlich offiziell veröffentlicht. Daraufhin durchging die Programmiersprache mehrere pre-1.0 Versionen, bis sechs Jahre später die Veröffentlichung von Julia 1.0 folgte. Damit war die erste stabile Version öffentlich.

# 2.2 Anwendungen

Der grundlegende Aufbau Julias verspricht ihr von Natur aus einen Platz in der Data-Science. Vor allem für Big-Data-Anwendungen und maschinelles Lernen eignet sich Julia aufgrund ihrer hohen Geschwindigkeit. Ähnlich werden ihre Fertigkeiten auch im Finanzwesen angewendet, um Märkte zu modellieren, Risiken zu bewerten oder komplexe Finanzinstrumente zu berechnen. Dabei wird wieder auf die Geschwindigkeit, aber besonders auch auf die Genauigkeit Julias Wert gelegt.

Die effiziente Ausführung mathematischer Operationen

## 2.3 Konkurrenten

1. Bezanson, J., Karpinski, S., Shah, V., & Edelman, A. (2012, 14. Februar). “Why We Created Julia”. https://julialang.org/blog/2012/02/why-we-created-julia/ [↑](#footnote-ref-1)