ReactiveRuntime

Schlussbericht

Zuwendungsempfänger:

Simon Danisch

Das diesem Bericht zugrundeliegende Vorhaben wurde mit Mitteln des Bundesministeriums für Bildung und Forschung unter dem Förderkennzeichen 01IS19S27 gefördert. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt beim Autor.

Kurze Darstellung der Aufgabenstellung und Motivation

Veröffentlichungen und Reproduzierbarkeit ist ein Grundstein der wissenschaftlichen Methodik. Zur Bereitstellung von Daten, Analysen und Code erfreuen sich Jupyter Notebooks steigender Beliebtheit. Notebooks sind interaktive Dokumente, in denen man Code entwickeln kann und die Resultate als multi-media Output angezeigt bekommt. Die Veröffentlichung dieser Notebooks und die Interaktivität der generierten Grafiken gestaltet sich jedoch als Anspruchsvoll.

Bei diesem Projekt, wird die Kalkulation ähnlich wie bei Spreadsheets (z. B. Excel) durch einen Graphen dargestellt, der die Abhängigkeiten zwischen den benutzten Variablen beschreibt.

Dies ermöglicht die effiziente, automatische Ausführung des gesamten Notebooks, z.B. wenn sich Inputs verändern. Zusammen mit interaktiven Widgets wie Slidern, kann man so spielerisch komplexe Daten und wissenschaftliche Modelle erforschen.

Eine Hauptinnovation dieses Projekts ist es, dass man durch das Erfassen des gesamten Ausführungsgraph alle Berechnungen speichern kann - das heißt, egal ob die Berechnungen auf einem Supercomputer gelaufen sind, oder auf einem Laptop mit einem Komplizierten Setup, ist es möglich die Resultate interaktiv im Internet einem breiten Publikum vorzustellen.

Meilensteine:

1. Verbesserung der Stabilität, Performance und Dokumentation des aktuellen Prototypen
2. Verbesserung der Integration von Javascript und Makie[1] für bessere Visualisierungen

[1] https://github.com/JuliaPlots/Makie.jl

3. Entwicklung der Zwischenspeicherung und Vorberechnung von Ergebnissen mittels des Ausführungsgraphes. Dadurch wird Interaktivität im Browser ohne laufenden Julia Prozess ermöglicht

4. Wrappen der Bibliothek für die Verwendung von Python, Release von Version 1 als Python Bibliothek

5. Erste Veröffentlichungen von interaktiven Modellen mit den wissenschaftlichen Partnern

Beitrag des Projektes zu den Zielen der Förderinitiative „Software-Sprint“

Die klassischen Zielgruppen von diesem Projekt sind Organisationen, die wissenschaftliche Artikel und Modelle veröffentlichen. Diese sind zum Beispiel das Umweltbundesamt, NASA, National Center for Environmental Information und Universitäten.

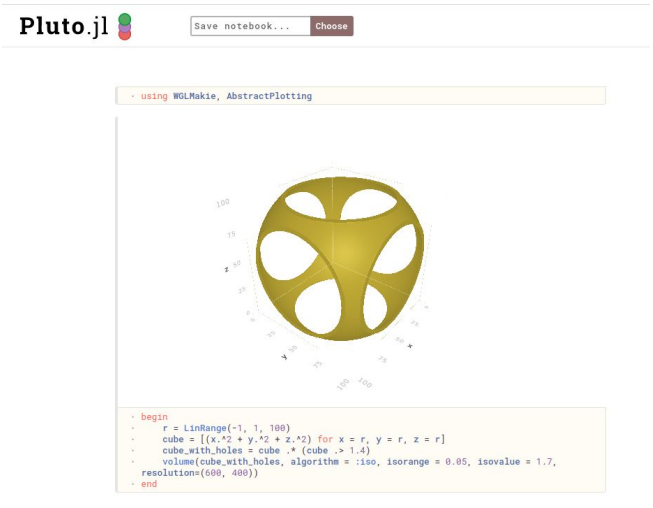
Eine einfache Veröffentlichung interaktiver, reproduzierbarer Grafiken hilft bei der Kommunikation der Wissenschaft. Wenn jeder Leser direkt mit den Modellen und Daten interagieren kann, hilft dies Vertrauen zu schaffen, da die Leser sicher sein können, dass Ihnen nicht nur handverlesene Parameter des Models präsentiert werden, um eine bestimmte narrative zu gestalten.

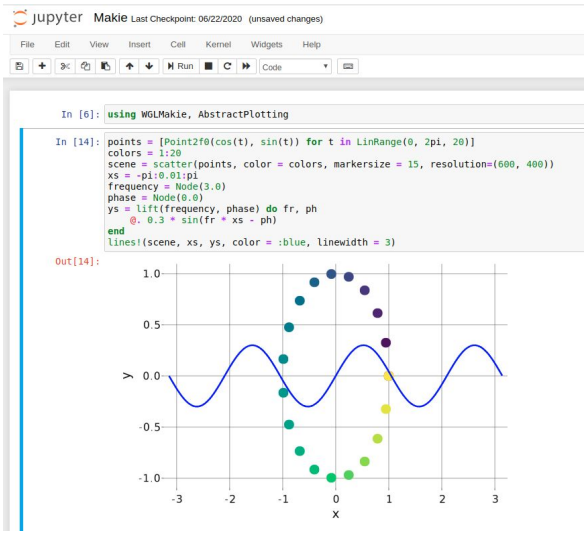
Wenn interaktive Veröffentlichung so zugänglich gemacht werden, dass selbst einzelne Forscher mit wenig Software Entwicklungserfahrung ihre Daten und Modelle veröffentlichen können, hilft dies auf breiter Basis das Internet verständlicher zu gestalten und lässt die Menschen mehr am wissenschaftlichen Prozess teilhaben - was wiederum hilft besser demokratisch Entscheidungen zu fällen.

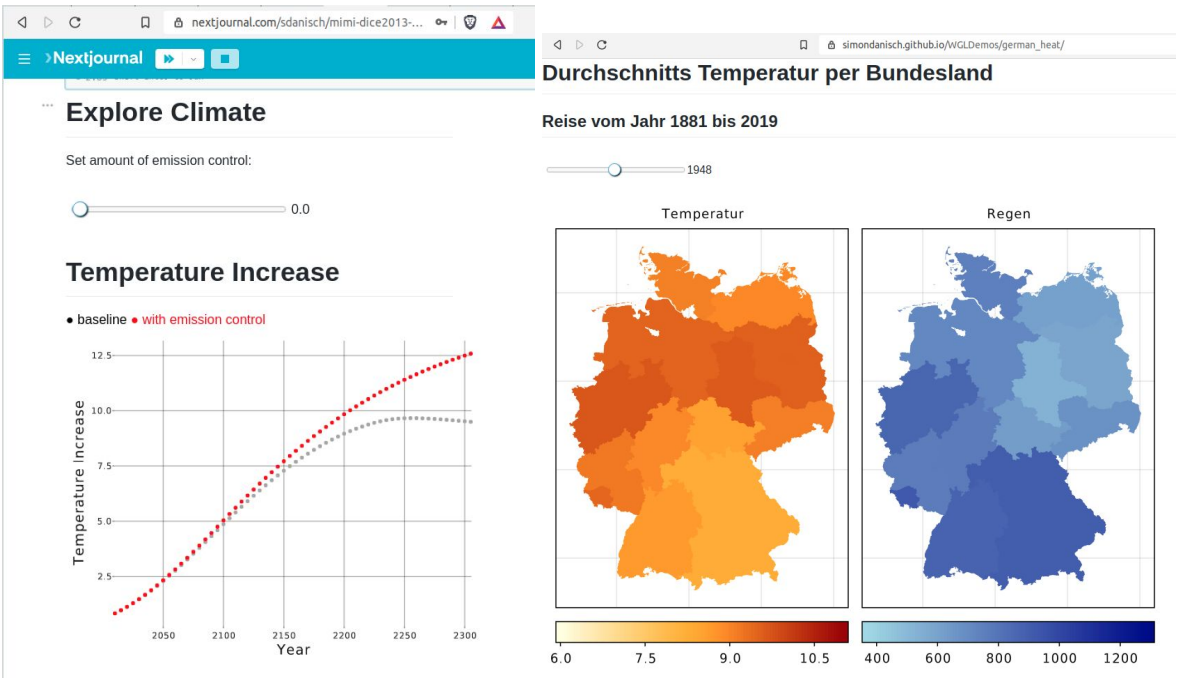
Diesen Beitrag leistet ReactiveRuntime mit Makie, da selbst Studenten ohne ein großes Team mit extra Web-Entwicklern ihre Forschung interaktiv im Internet präsentieren können.

Ausführliche Darstellung der Ergebnisse

Makie funktioniert dank des PrototypeFunds nun 100 % interaktiv in Jupyter[2], Pluto.jl[3], Nextjournal[4] und im Browser[5]:







Dies sind die meistgenutzten Plattformen für Code Explorationen. Mit ReactiveRuntime können diese Plattformen nun voll genutzt werden, und es wurde sehr einfach gemacht, von dort aus die interaktiven Visualisierungen zu publizieren.

---

[2] https://jupyter.org/
[3] https://github.com/fonsp/Pluto.jl
[4] https://nextjournal.com/sdanisch/mimi-dice2013-climate-impact/
[5] https://simondanisch.github.io/WGLDemos/german_heat/

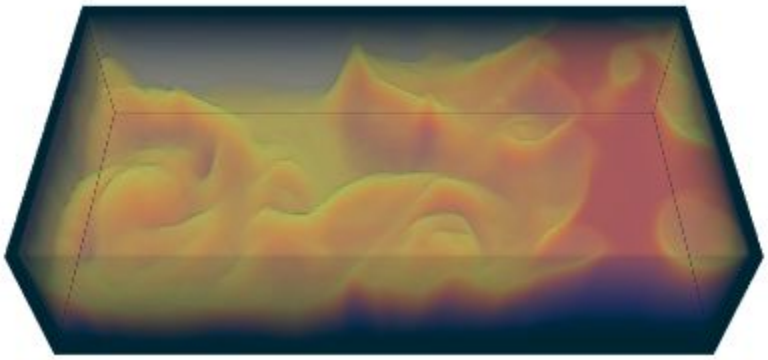
Pluto.jl ist eine sehr neue Julia Notebook Plattform, die es am Anfang meines Projektes noch nicht gab. Sie implementiert perfekt die Spreadsheet artige Notebook Ausführungsweise, die ich mir für ReactiveRuntime vorgestellt hatte. Dies hat Makie.jl mehr in den Mittelpunkt des Projektes gerückt und ich habe mehr Zeit investiert, dass Makie.jl besser im Web funktioniert und gut getestet und dokumentiert ist.

Der Rest ist in die Implementation des “offline-exports” geflossen. Dieser ermöglicht es ohne Laufenden Julia Prozess, auf jedem Webserver interaktive Datenanalysen/Grafiken einzubinden. Dies wird möglich gemacht, indem mithilfe des Ausführungsgraph alle möglichen Berechnungen gespeichert werden und so im Web verfügbar gemacht werden können.

Das Hauptergebnis dieses Projekts sind jedoch mehrere Veröffentlichungen mit den wissenschaftlichen Partnern!

CliMA[6]

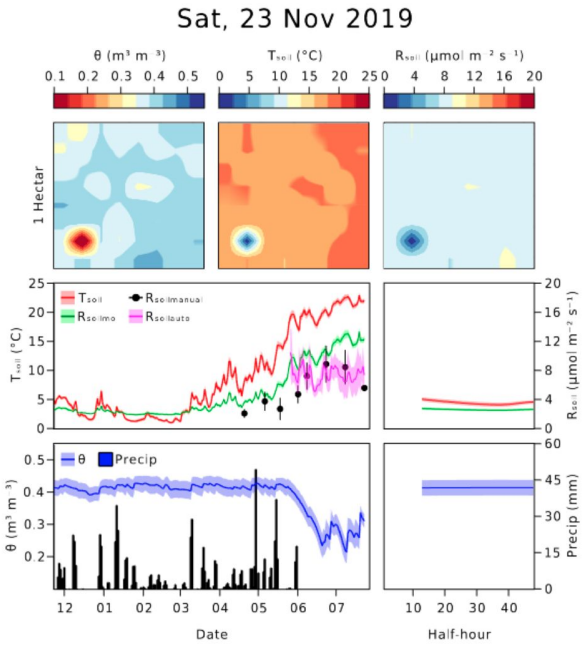
CliMA (Clima Modeling Aliance) ist ein großes Projekt von dem CALTECH (California Institute of Technology) um das Klima zu modellieren und so gut wie möglich Strategien für die Bekämpfung der Klimakrise zu entwickeln. Im Moment arbeite ich mit dem Unterprojekt Oceananigans[7] zusammen, um Meeres Simulationen interaktiv zu visualisieren[8]. Da alle Projekte der CliMA Organisation Julia benutzen, ist es zu erwarten, dass die Anwendung von Makie noch um einiges steigt.



[6] https://clima.caltech.edu/
[7] https://github.com/CliMA/Oceananigans.jl/
[8] https://simondanisch.github.io/WGLDemos/oceananigans/

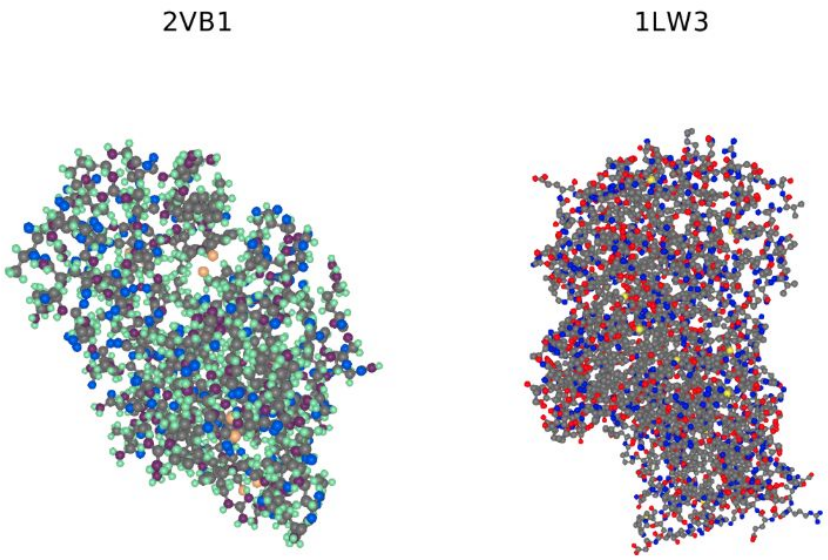
Argonne National Laboratory [9]

Benutzt Makie für eine interaktive Datenexploration. Dargestellt ist ein Datensatz bei dem die Feuchtigkeit, Temperatur und Niederschläge in heterogenem Grasland aufgezeichnet wurden. Es wird außerdem geplant, eine Webseite zu erstellen, um noch mehr Datensätze zu präsentieren und zu erklären.



Iowa State University [10]

Benutzt Makie für interaktive Molekül Simulationen und Explorationen:

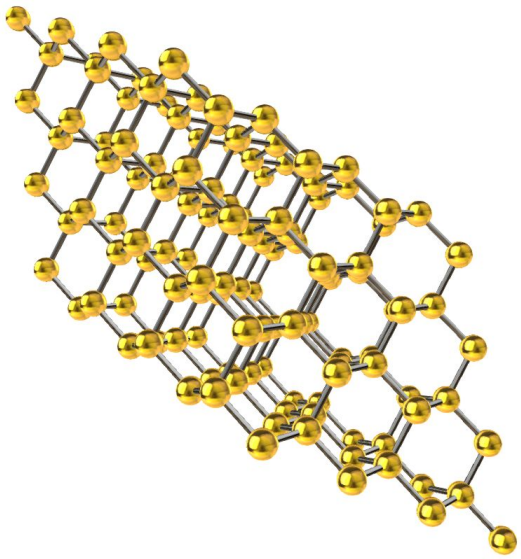


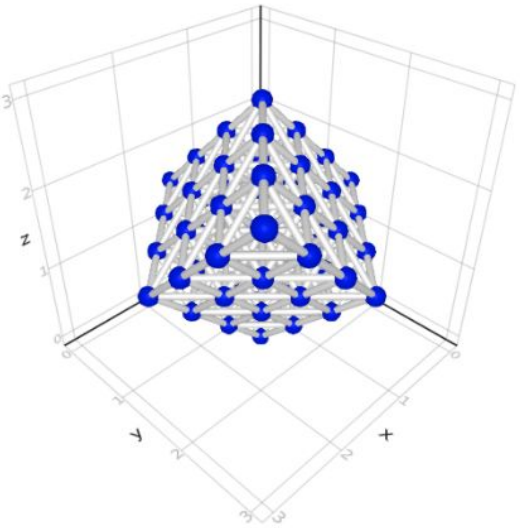
[9] https://simondanisch.github.io/WGLDemos/soil/

[10] https://github.com/kool7d/BioMakie.jl

Universität zu Köln [11] [12]

Interaktives plotten von Kristallgittern:





Von den Meilensteinen wurde hauptsächlich der Python-Wrapper ausgelassen, da die Zeit knapp geworden ist. Aber im Prinzip ist die Julia Python Interaktion ausgereift genug, dass jeder ohne großen Aufwand die entstandene Software von Python aus benutzen kann.

Zielgruppe, Nutzen und mögliche Weiterentwicklungen

Wie schon in den Ergebnissen erwähnt, benutzen viele Gruppen schon Makie und ReactiveRuntime für ihre Veröffentlichungen. Es gibt auch mehrere interessante Projekte, die gerade noch in der Umsetzungsphase sind und erst bald veröffentlicht werden.

Man kann also sagen, dass die Makie Community sehr aktiv ist. Es gibt mehrere Open Source Mitarbeiter die wöchentlich Verbesserungen beitragen und ich habe auch viele Pläne für die Weiterentwicklung.

Eines der großen Ziele, dem Makie schon sehr nah ist, ist es reibungslos zwischen Browser, Desktop und PDF Publikationen zu wechseln und in diesen sehr unterschiedlichen Medien, die gleichen APIs und Benutzer Erlebnisse zu Verfügung zu stellen. Es soll also möglich sein, interaktive Visualisierungen genauso gut als Desktop Applikationen zu veröffentlichen, wie auf einer Webseite, oder eben auch direkt in einer klassischen wissenschaftlichen Veröffentlichung in einem Journal (PDF). Es gibt noch immer kleine visuelle Unterschiede in Makie zwischen Desktop

[11] https://github.com/ffreyer/LatPhysPlottingMakie.jl
[12] https://github.com/janattig/LatticePhysics.jl

(OpenGL[13] [14]), Web (JS + WebGL[15] [16]), PDF/SVG (Cairo[17] [18]), die ich in Zukunft eliminieren möchte.

Die nächste Baustelle ist das effiziente speichern und übertragen von den Berechnungen. Die kombinatorische Explosion ist groß, und somit fallen schnell große Datenmengen an. Es gibt zwei sehr vielversprechende Ansätze, die ich ausprobieren möchte:

- Kompilation von Julia nach Javascript, so, dass viele Berechnungen auch im Browser ausgeführt werden können und nicht gespeichert werden müssen.
- Streamen und komprimieren der Zustände. Im Moment werden alle benötigten Daten auf einmal zum Browser geschickt, was die Wartezeit in die Höhe treibt. Das gute ist jedoch, dass Streaming perfekt funktioniert, da die Zustände schon unabhängig gespeichert werden und somit leicht in einzelnen Paketen an den Browser geschickt werden können.

Außerdem möchte ich noch weiter an Pluto.jl arbeiten und einer performanten Makie Integration. Die aktuelle Integration basiert auf der Ursprünglichen Jupyter Integration, welche noch nicht ganz ausgereift ist, im Vergleich zu dem was möglich wäre.

Kurze Darstellung der Arbeiten, die zu keiner Lösung geführt haben

Am Anfang der Planung bin ich noch davon ausgegangen spezifische Technologien wie CBOR[19] oder ObservablesHQ[20] für die implementation der Kommunikation zwischen Julia und Javascript und für das Reaktive verhalten zu benutzen. Dies sind jedoch Implementations Details, die nichts wirklich in der groben Planung zu suchen hatten. Es kam dann relativ schnell raus, das es simplere alternativen gibt um Julia und Javascript zu kommunizieren zu lassen. Pluto.jl hat außerdem alle wünsche erfüllt, die ich mir von ObservablesQH versprochen hatte.

Kurze Angabe von Präsentationsmöglichkeiten für mögliche Nutzer

- Twitter
- https://github.com/JuliaPlots/Makie.jl
- https://github.com/SimonDanisch/JSServe.jl
- Julia Slack #makie channel

[13] https://www.opengl.org/
[14] https://github.com/JuliaPlots/GLMakie.jl
[15] https://github.com/JuliaPlots/WGLMakie.jl
[16] https://de.wikipedia.org/wiki/WebGL
[17] https://github.com/JuliaPlots/CairoMakie.jl
[18] https://de.wikipedia.org/wiki/Cairo_%28Grafikbibliothek%29
[19] https://cbor.io/
[20] https://observablehq.com/

Kurze Erläuterung zur Einhaltung der Arbeits- und Kostenplanung

Ich habe ein Projekt mit einer Firma die Makie & JSServe[21] benutzt angefangen, welches mir gegen Ende des PrototypeFunds ein wenig die Zeit geraubt hat die letzten Details zu implementieren, wie zum Beispiel die Python Integration. Deswegen habe ich auch nicht den letzten Zeitraum voll abgerechnet. Die guten Neuigkeiten sind, dass ich weiterhin bei der Firma bleibe und Makie & ReactiveRuntime weiter Open Source entwickeln werde!

Kurze Darstellung von etwaigen Ergebnissen bei anderen Stellen

Pluto.jl kam aus dem nichts und hat eine unerwartet gute reaktive Alternative zu Jupyter geboten. Dies hat den Fokus von reaktiven Notebooks auf Geschwindigkeitsverbesserung, Stabilität, und Publikationen im Web gelenkt.

[21] https://github.com/SimonDanisch/JSServe.jl