Die Programmiersprache Julia

und ihre AI Bibliotheken

CGI

julia

Falko Spiller, 01/Apr/2021

- IT Diensleister, IT Beratung
- Ca 75.000 Mitarbeiter, in Deutschland ca 2.300
- In Hamburg ca 330

Falko

- Physik studiert
- mathelastiger IT Berater
- Polyglott Programmer

Was ist mein Ziel? Was nicht?

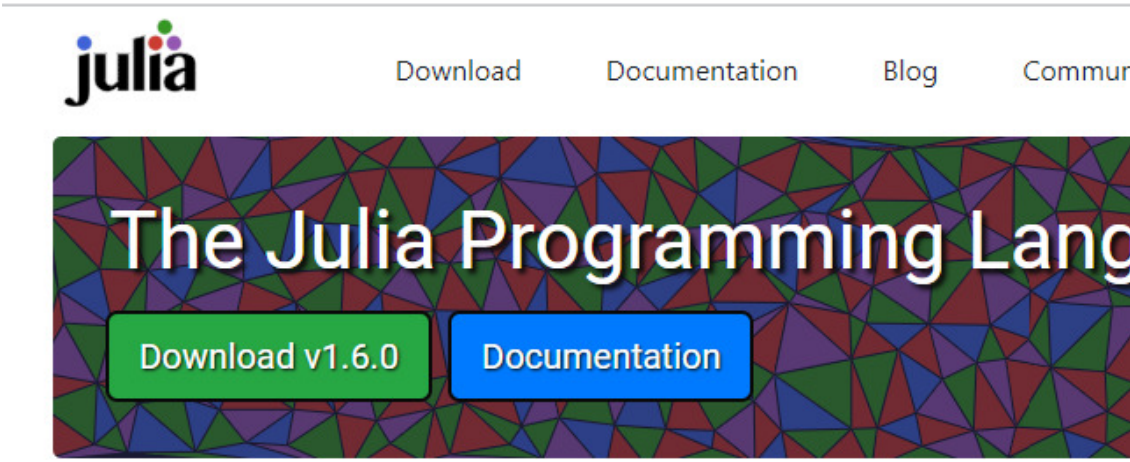
+ Über eine Programmiersprache reden, die ich mag.
+ Testen, ob ich sie für weitere Talks hier benutzen kann,
  („Probabilistic Programming mit Turing.jl“)
+ Bei Interesse Julia-Meetup, o.ä.

- Ich will niemandem seine Lieblingssprache abspenstig machen.
- Oder irgendeine Sprache für „die beste“ erklären.

julia

Buzzwords: Julia ist

- Modern & Interactiv.
- Wie Python, nur in schnell.
- Wie R nur modern.



Mehr Buzzwords

- Hohe Performance, besonders auch bei Numeric
- Dynamically typed (Optionale Type-Annotationen)
- Multi paradigm
- Just in Time Compilation (LLVM)
- Mathy Syntax möglich
- Multiple dispatch
- Makros (Code-Transformation)
- Gut Dokumentiert
- Lebendige Nutzergemeinschaft

julia

Geschichte

Begonnen in 2012, stabil (0.4.6) 2016

1.0 im August 2018

Backed by a group from MIT

Aktuell 1.6.0 (25. März 2021)

(Ich bin seit spätestens 0.6.2 dabei.)

julia

Das „Two language problem“

zB Deep learning mit Tensorflow

Python?

C++!

Julias erklärtes Ziel ist Code-Ausführung, die mit C mithalten kann.

julia

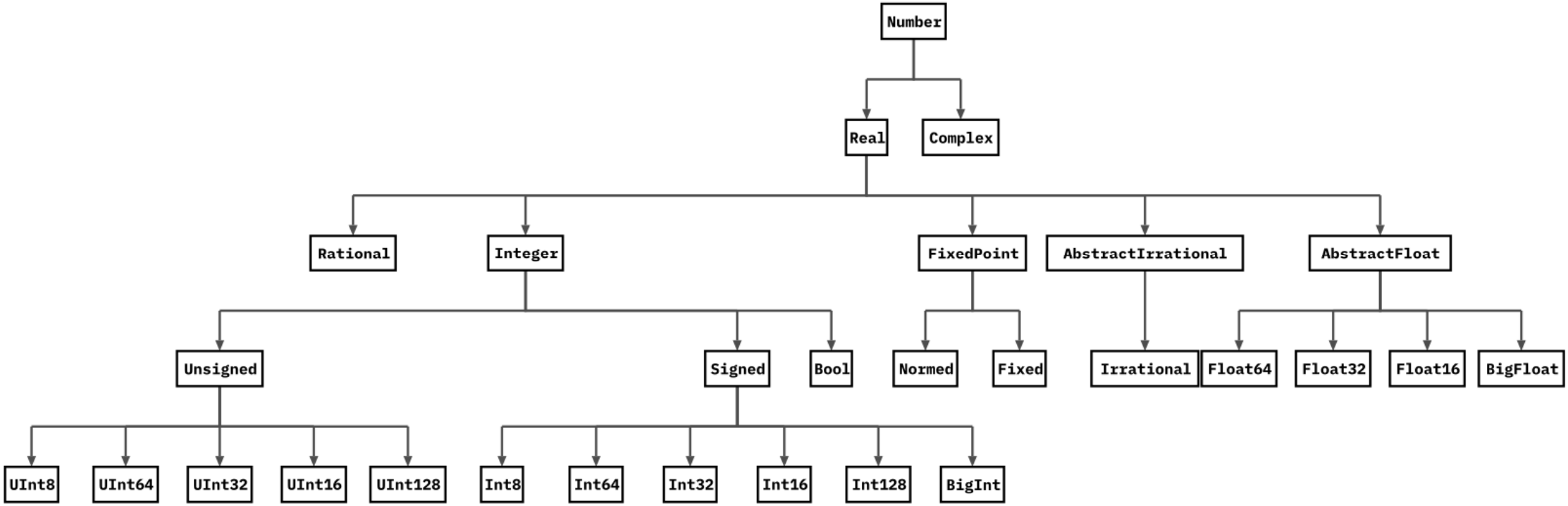
Code?

```
p(x) = 2x^2 + π
f(x, y) = 1 + 2p(x)y

function fibo(n)
    if n < 2
        1
    else
        fibo(n-1) + fibo(n-2)
    end
end
```

julia

Numbers



julia

zB. Rationale Zahlen

```
julia> third = big(1)//3
1//3
julia> fifth = big(1//5)
1//5
julia> x = third / fifth
5//3
julia> x^42
227373675443232059478759765625//109418989131512359209
```

Komplexe Zahlen

```
julia> y = x + im*third
5//3 + 1//3*im

julia> y^20
-97710243226624//3486784401 - 33962164019200//1162261467*im
```

Numerik, mathy syntax, & Vektorisierung

```
julia> sin(π/3)
0.8660254037844386

julia> sin.(π./[1, 2, 3, 4, 5])
5-element Array{Float64,1}:
 1.2246467991473532e-16
 1.0
 0.8660254037844386
 0.7071067811865475
 0.5877852522924731
```

julia

Matrizen

```
A = rand(Int16,4,12)
```

Transposition mit '

```
A'
```

Matrix-Multiplikation? **A*A**, oder punktweise: **A .* A**

Funktionaler Stil

Jeder Ausdruck hat einen Wert.

```
[n*(n+1) for n in 1:5]
```

```
sin.(1:20)
```

```
map(x -> x^2, 1:20)
sum(filter(even, 1:100))
```

Type Annotationen sind möglich (meist unnötig)

```
p(x :: Float64) :: Float64 = 2x^2 + π
f(x :: Float64, y :: BigInt) = 1 + 2p(x)y

function fibo(n :: Int64) :: BigInt
    if n < 2
        big(1)
    else
        fibo(n-1) + fibo(n-2)
    end
end
```

julia

DLL Aufrufe

Time in milli seconds (Windows)

```
ccall((:clock, "msvcrt"), Int32, ())
```

julia

Der REPL, interaktive Shell

- Package-Mode mit] (Who needs maven when you got a REPL?)
- Hilfe-System mit ? , sowie
  - **`apropos(„lowercase“)`**
  - **`methods(lowercase)`**, **`methodswith(type)`**
- Tab-Completion. Auch für Argumente, TeX-Symbols mit \
- Shell-Zugriff mit `;`
- Um Ausgabe zu unterdrücken, Ausdruck mit `;` `abschließen`
- **`@time`**
- Zugriff auf letztes Ergebnis: **`ans`**
- Befehls-Historie, inkl. Suche (bash like) mit ^r

julia

Cons?

Neu.

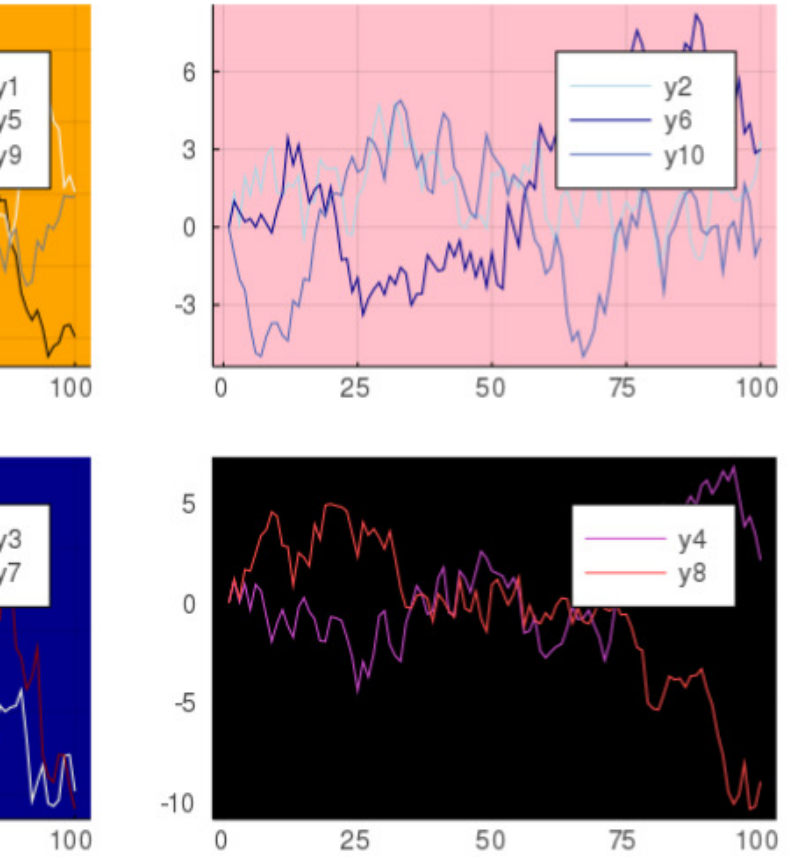
Keine große Firma dahinter, also keine Betreuung [aber Julia Computing, Inc.]
(Es gibt coole, freundliche, und schnell reagierende Anwender.)

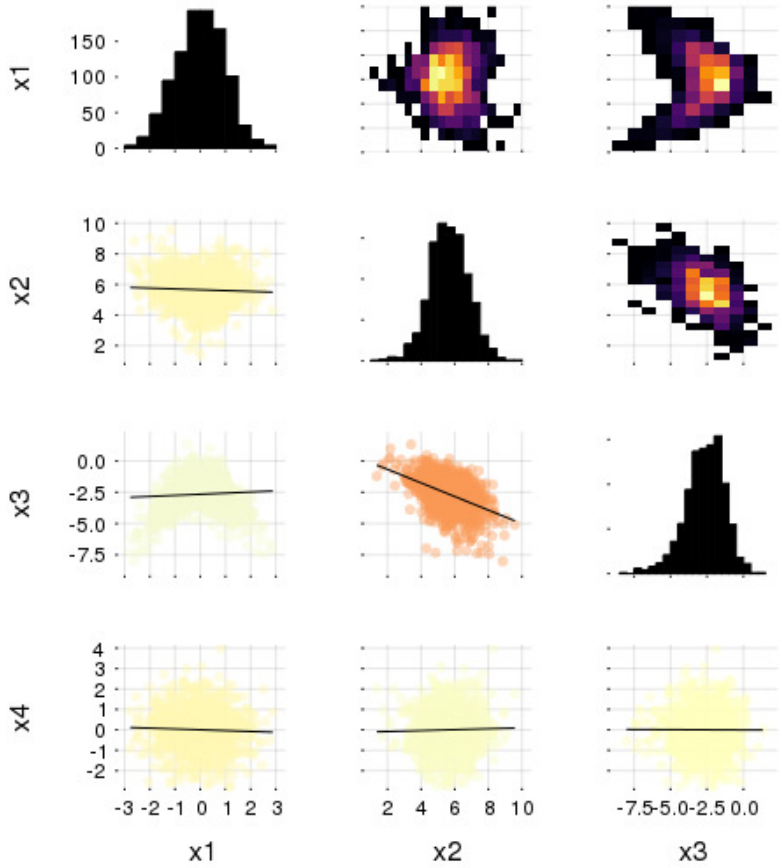
Fast moving
(Dokumentation ist u.U. veraltet,
Pakete inkompatibel. Letzteres ist mit den letzten Releases besser geworden)

TTFP: Time To First Plot (Just-in-time muss erst laufen)
Große Verbesserung letzte Woche! (Version 1.6.0)

julia

Grafik (Plots & StatsPlots)





Beide Graphiken aus der Dokumentation

Where to get it?

Julia itself
https://julialang.org/

IDE:
Atom with Juno http://junolab.org/
Julia-mode for Emacs
Pluto.jl

Communication
https://discourse.julialang.org/

Bibliotheken, interessant für AI

- **Notebooks: Pluto.jl**
- **Deep learning library Flux.jl**
- ***Probabilistic programming: Turing.jl***
- Gaussian Processes: Stheno.jl
- Data exploration: Queryverse.jl (mit DataFrame.jl, CSV.jl, und Plots/StatsPlots)
- Automatic Differentiation: mehrere, zB. Zygote.jl

Where to find much more:
https://juliaobserver.com/

julia

Pluto.jl, reactive Notebooks

Pluto.jl

```
a = 13
```
```
a = 13
```

```
b = 27
```
```
b = 4a - 25
```

```
c = 1
```
```
c = b % 13
```

```
Enter cell code...
```

- Ähnlich Jupyter, aber Abhängigkeiten werden verwaltet (wie Excel!)
- Man muss Ordnung halten.
- Man verliert nicht so leicht den Überblick

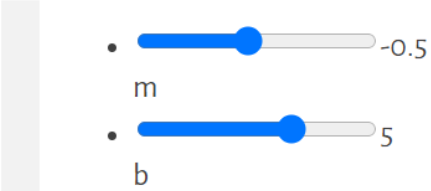
Notebooks sind reines Julia (+ Kommentare), also auch ohne Pluto lauffähig!

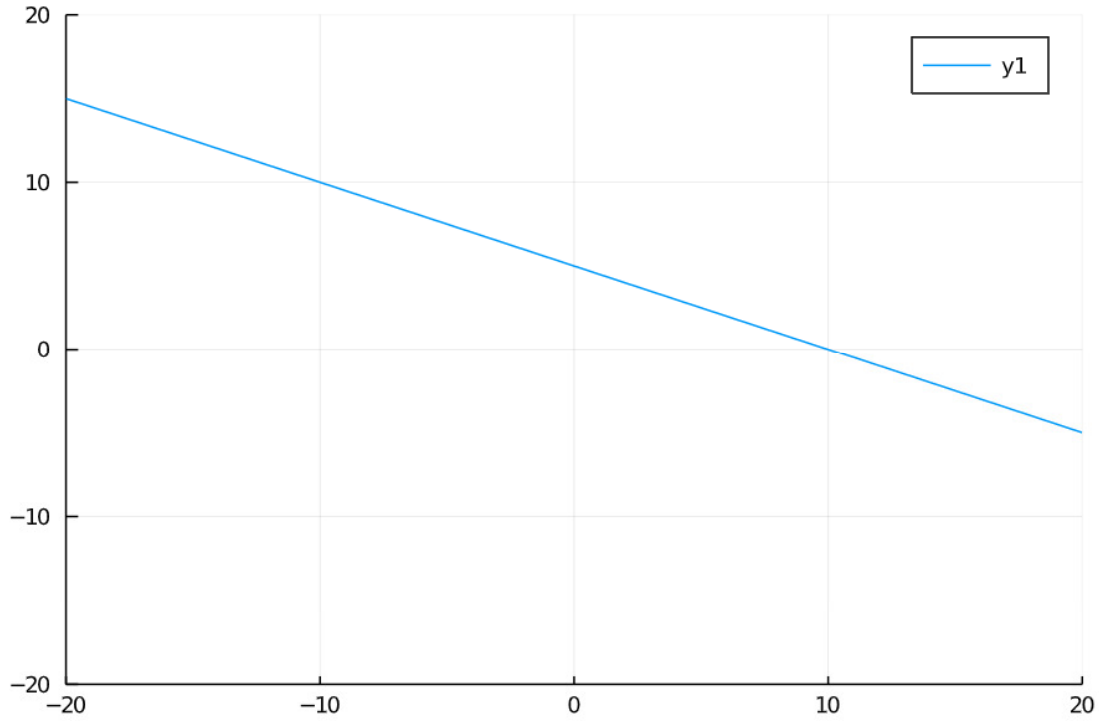
julia

Pluto.jl

- Interactive
- Markdown / LaTeX in Text-Zellen

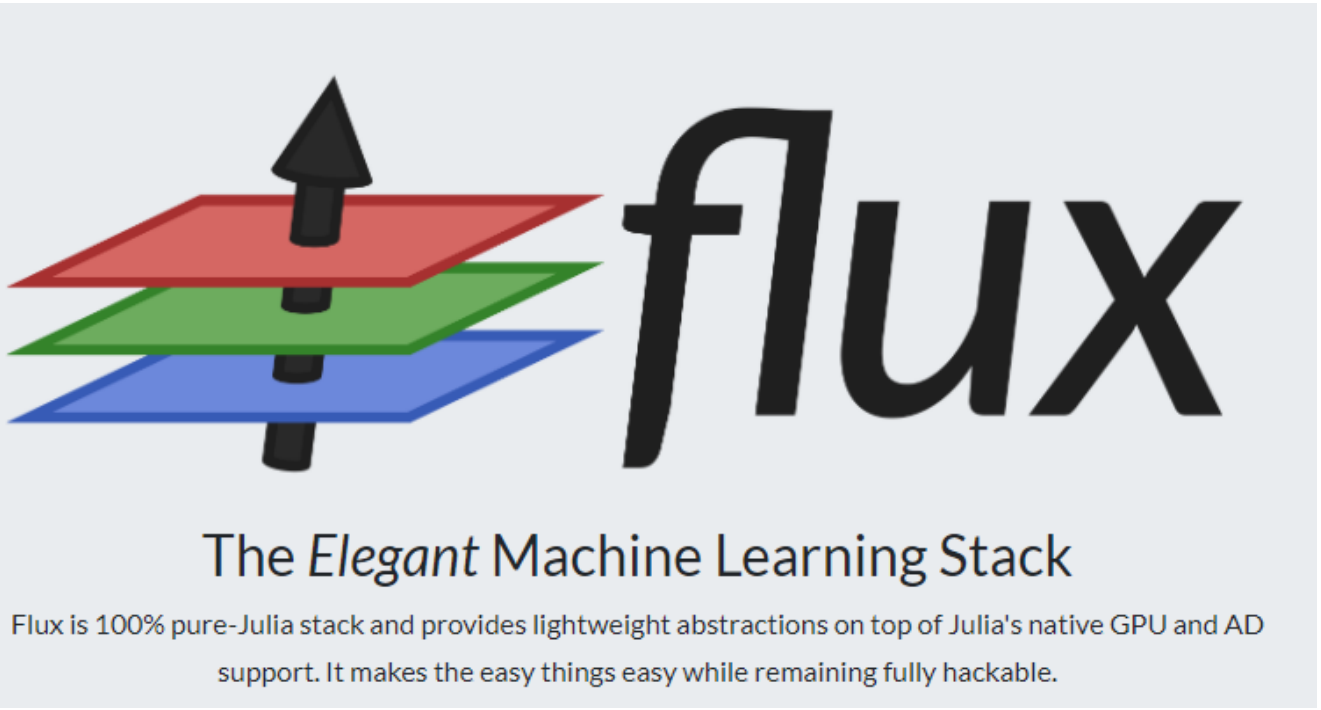
Ausgezeichnet für die explorative Phase!






```
begin
    xs = -20:0.2:20
    plot(xs, map(x -> m*x+b, xs),
        xlims=(-20,20),
        ylims=(-20,20),
    )
end
```

Deep learning mit Flux.jl - https://fluxml.ai/




```
Chain(Conv((5, 5), px[end]=>6, relu),
      MaxPool((2, 2)),
      Conv((5, 5), 6=>16, relu),
      MaxPool((2, 2)),
      flatten,
      Dense(prod(out_size), 120, relu),
      Dense(120, 84, relu),
      Dense(84, nclasses)
)
```

Ein CNN für MNIST

julia

Flux.jl

- **Alles ist Julia** (keine darunterliegende schnelle C++ Schicht)
- **Automatic Differentiation** für (fast) beliebigen Julia-Code
  Activation-Functions!
- Controll-Statements, Loops, Custom Types
- CUDA-Anbindung

Julia ist überraschend „composable“!

Turing.jl – Probabilistic Programming *(Plan für Mai)*

- Stochastische Modelle, Bayesian Inference
- Generative Modelle, Markov Chain Monte Carlo
- Explainable AI
- Möglichkeit, Expertenwissen ins Modell aufzunehmen
- Möglichkeit, Vorschriften ins Modell aufzunehmen
- Kausalität als Eigenschaft der Modelle

julia

Danke

Interesse an Meetups? Oder vergleichbarem Format bei ARIC?

Interesse an Wissensaustausch?

falko.spiller@cgi.com

CGI

julia