julia – A pragmatic approach to scientific computing

Maren Hackenberg

Institute of Medical Biometry and Statistics, Faculty of Medicine and Medical Center – University of Freiburg, Germany

Mini symposium: Modern software tools for modern statistics

RStudio Blog

Home About RStudio.com SUBSCRIBE

CATEGORIES

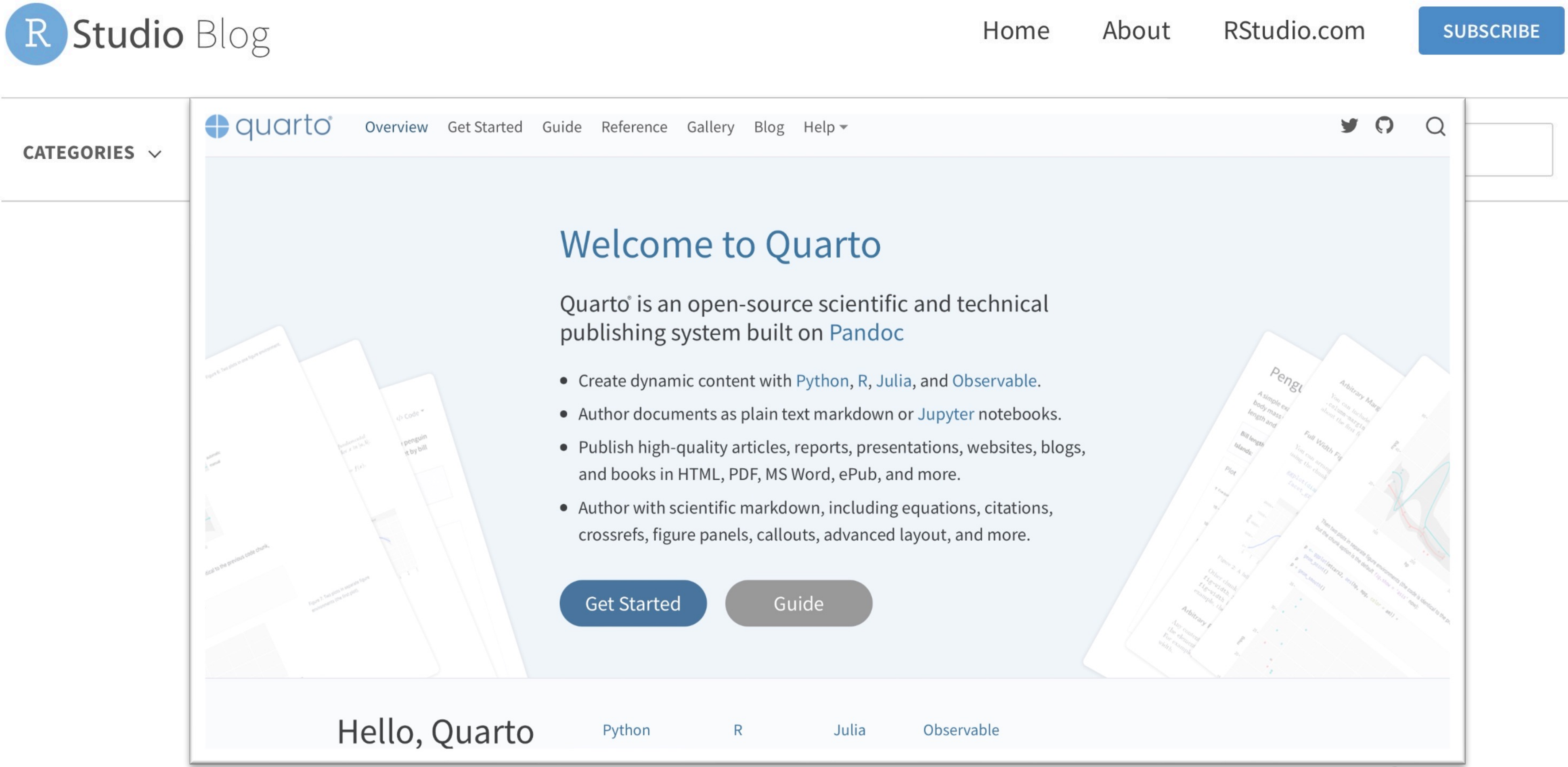
Search the blog

Jul 28 5 min

COMPANY NEWS AND EVENTS
OPEN SOURCE

Announcing Quarto, a new scientific and technical publishing system

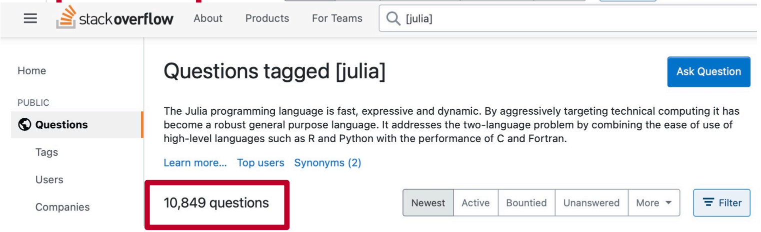
J.J. ALLAIRE


RStudio Blog
Home
About
RStudio.com
SUBSCRIBE
CATEGORIES
quarto
Overview
Get Started
Guide
Reference
Gallery
Blog
Help
Welcome to Quarto
Quarto® is an open-source scientific and technical publishing system built on Pandoc
• Create dynamic content with Python, R, Julia, and Observable.
• Author documents as plain text markdown or Jupyter notebooks.
• Publish high-quality articles, reports, presentations, websites, blogs, and books in HTML, PDF, MS Word, ePub, and more.
• Author with scientific markdown, including equations, citations, crossrefs, figure panels, callouts, advanced layout, and more.
Get Started
Guide
Hello, Quarto
Python
R
Julia
Observable

…so why Julia?

- Fast yet easy to learn with a syntax similar to R and Python
- Everything is pure Julia
- Easy interfaces to other languages

- Young language with a smaller user community
- Less training material
- Less mature package ecosystem



Julia doesn’t constrain your thinking

nature

Explore content ⌄ About the journal ⌄ Publish with us ⌄ Subscribe

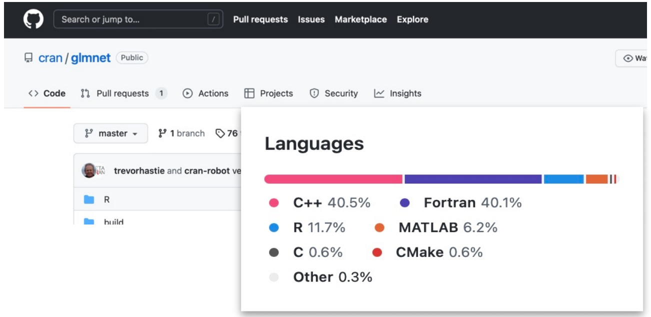
nature > toolbox > article

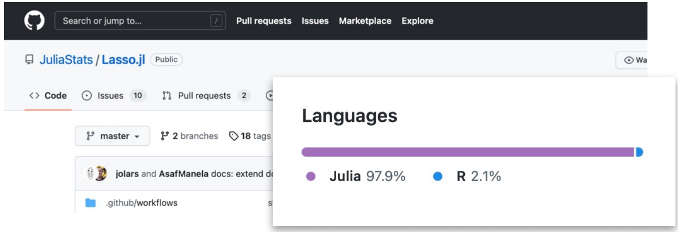
TOOLBOX | 30 July 2019

Julia: come for the syntax, stay for the speed

Researchers often find themselves coding algorithms in one programming language, only to have to rewrite them in a faster one. An up-and-coming language could be the answer.

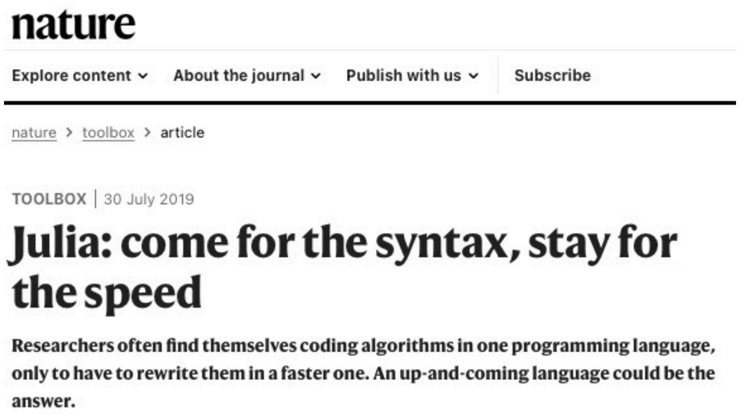
- Complex algorithms can be implemented in the language itself





Julia doesn’t constrain your thinking

- Complex algorithms can be implemented in the language itself



nature

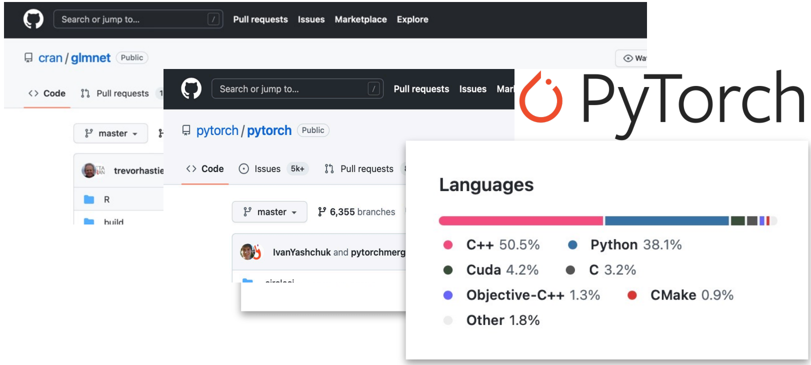
Explore content ⌄ About the journal ⌄ Publish with us ⌄ Subscribe

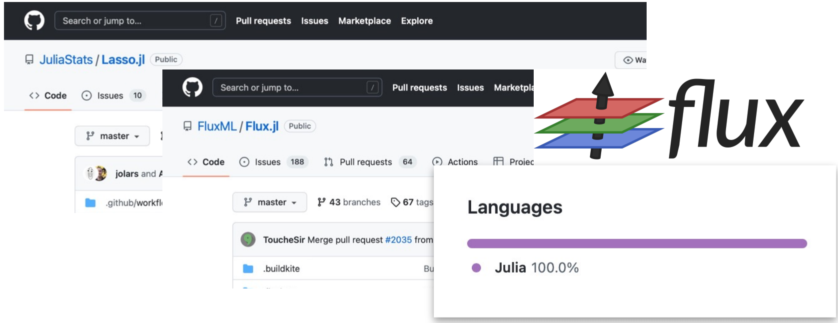
nature > toolbox > article

TOOLBOX | 30 July 2019

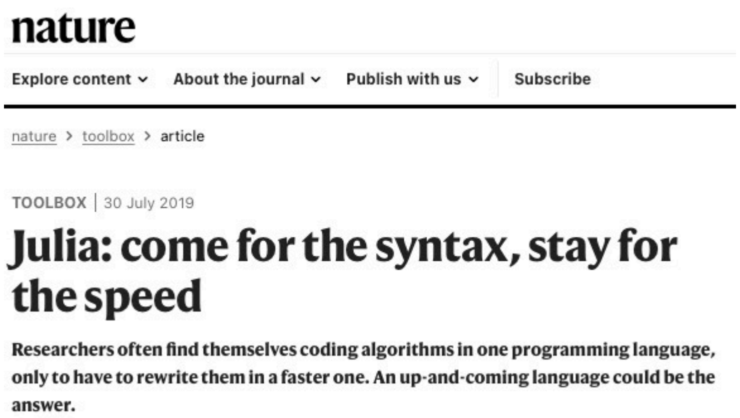
Julia: come for the syntax, stay for the speed

Researchers often find themselves coding algorithms in one programming language, only to have to rewrite them in a faster one. An up-and-coming language could be the answer.





Julia doesn’t constrain your thinking



nature

Explore content ⌄ About the journal ⌄ Publish with us ⌄ Subscribe

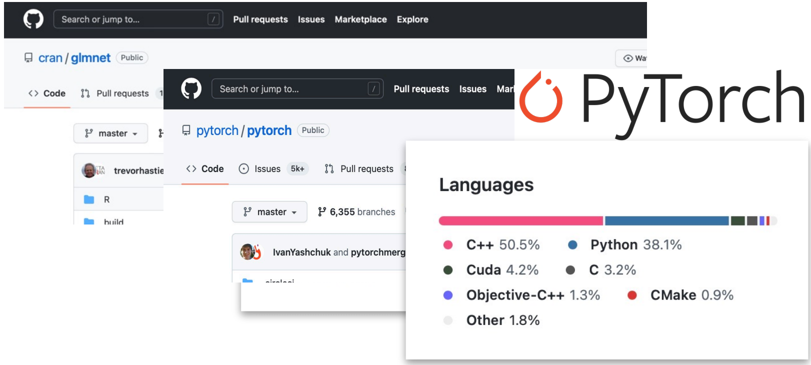
nature > toolbox > article

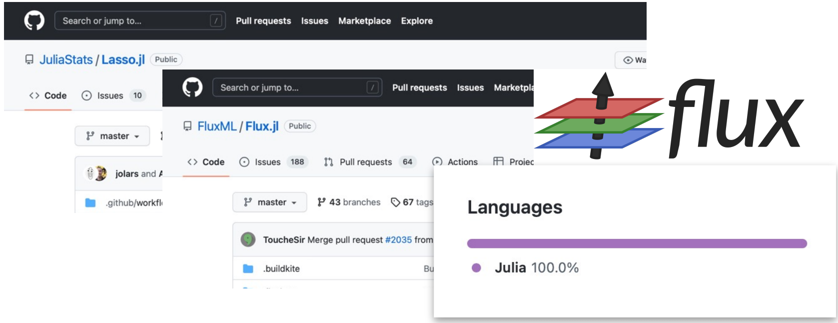
TOOLBOX | 30 July 2019

Julia: come for the syntax, stay for the speed

Researchers often find themselves coding algorithms in one programming language, only to have to rewrite them in a faster one. An up-and-coming language could be the answer.

- Complex algorithms can be implemented in the language itself
- Modification of existing libraries is easy, enabling faster prototyping and methods development
- No “legacy characteristics”

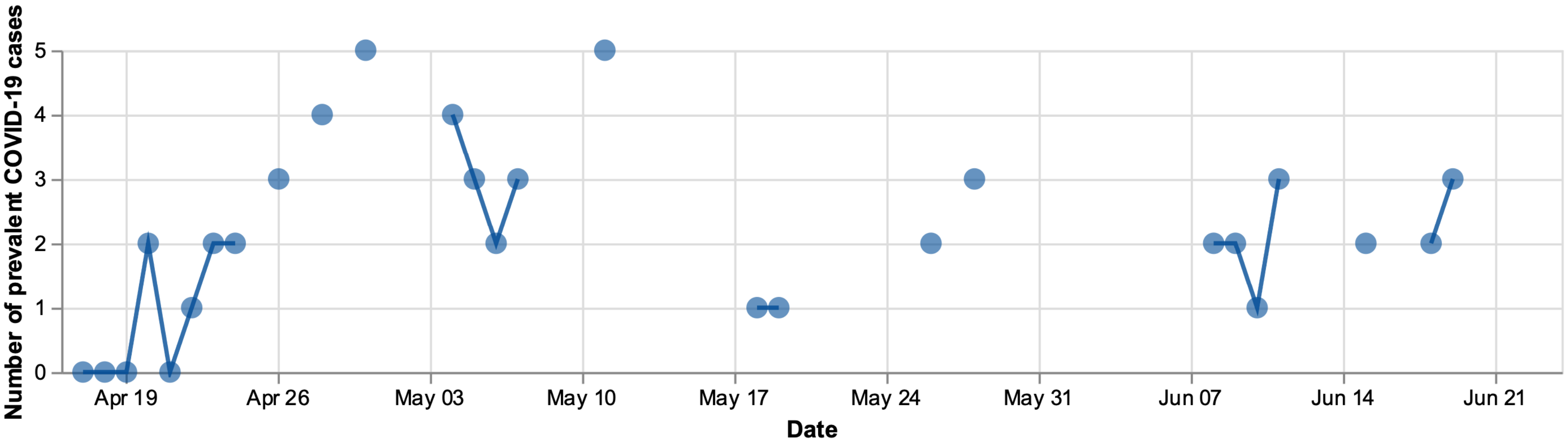




Modelling challenge I: Predict COVID-19 ICU demand

Missing data in the "Intensivregister"

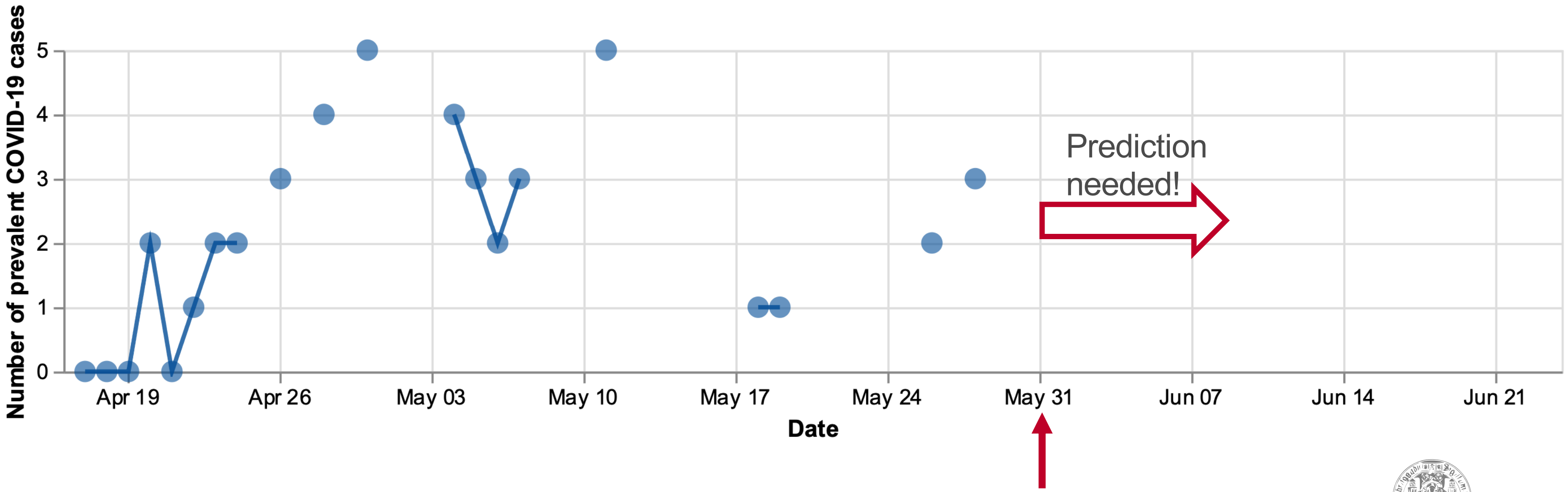
- 63% of all hospitals have missing daily reports
- 7% of hospitals with more than 25% missing daily reports



Modelling challenge I: Predict COVID-19 ICU demand

Missing data in the "Intensivregister"

- 63% of all hospitals have missing daily reports
- 7% of hospitals with more than 25% missing daily reports



A model to predict increments of daily cases

For each hospital:

- Linear model of the increments

current ICU cases

$$\widehat{\mathrm{d}y}_{t+1} = (y_{t+1} - y_t) = \beta_1 + \beta_2 \cdot y_t + \beta_3 \cdot z_t, \quad t = 2, \ldots, T.$$

newly infected COVID-19 cases in the county

- Accounting for missing daily reports

$$\widehat{\mathrm{d}y}_{t+1} = \beta_1 + \beta_2 \cdot \widetilde{y}_t + \beta_3 \cdot z_t, \text{ where } \widetilde{y}_t = \begin{cases} y_t, & \text{if } y_t \text{ was observed,} \\ \widetilde{y}_{t-1} + \widehat{\mathrm{d}y}_t, & \text{else.} \end{cases}$$

- A non-linear optimisation problem

$$\begin{aligned} \widehat{\mathrm{d}y}_{t+1} &= \beta_1 + \beta_2 \cdot \widetilde{y}_t + \beta_3 \cdot z_t \\ &= \beta_1 + \beta_2 \cdot (\widetilde{y}_{t-1} + \widehat{\mathrm{d}y}_t) + \beta_3 \cdot z_t \\ &= \beta_1 + \beta_2 \cdot (\widetilde{y}_{t-1} + (\beta_1 + \beta_2 \cdot \widetilde{y}_{t-1} + \beta_3 \cdot z_{t-1})) + \beta_3 \cdot z_t \end{aligned}$$

Defining a loss function and optimising with Zygote

```
function loss(y, z, reported, β)
    sqerror = 0.0 # squared error
    firstseen = false # set to true after skipping potential missings
    lastcur = 0.0 # prevalent cases from previous time point
    contribno = 0.0 # number of non-missing observations
    for t=1:length(y)
        # skip missings at the start until first reported value
        if !firstseen
            if reported[t] == true # first non-missing observation encountered:
                firstseen = true
                last_y = y[t]
            else
                continue
            end
        else # make a prediction for current increment:
            pred_dy = β[1] + β[2] * last_y + β[3] * z[t-1]
            if reported[t] == true # if current time point is observed:
                dy = y[t] - last_y
                sqerror += (dy - pred_dy)^2
                contribno += 1.0
                last_y = y[t]
            else # if value at current time point is missing:
                last_y += pred_dy
            end
        end
    end
    return sqerror/contribno # return MSE over all observed time points
end
```

→ If you can program it, you can estimate the model parameters

*as long as the function is differentiable

Differentiable programming for flexible model building

Machine Learning

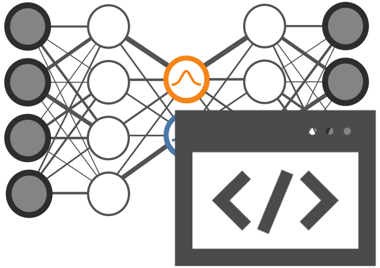
Scientific Computing

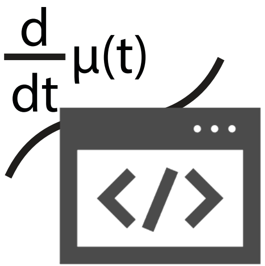
TensorFlow

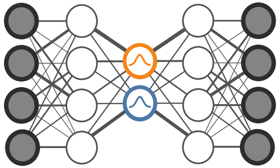
PyTorch

Zygote

write as a
computer program





loss( **,** $\frac{d}{dt}\mu(t)$ **)**

$\frac{d}{d}$  **loss**

$\frac{d}{d\,\frac{d}{dt}\mu(t)}$ **loss**

optimise via automatic
differentiation

Engineering trade-offs in automatic differentation tools

TensorFlow PyTorch JAX

- Graph-building system in static sub-language: efficient but less flexible
- Tape-based method: unrolls all control flow
- Overhead partly compensated by highly efficient kernels for machine learning applications

Zygote

- Source-to-source transformation
- Supports full dynamism
  - E.g., straightforward differentiation through ODE solvers
  - Supports user-defined structs and recursion
- Keeps control flow intact

Defining a loss function and optimising with Zygote

```julia
function loss(y, z, reported, β)
    sqerror = 0.0 # squared error
    firstseen = false # set to true after skipping potential missings
    lastcur = 0.0 # prevalent cases from previous time point
    contribno = 0.0 # number of non-missing observations
    for t=1:length(y)
        # skip missings at the start until first reported value
        if !firstseen
            if reported[t] == true # first non-missing observation encountered:
                firstseen = true
                last_y = y[t]
            else
                continue
            end
        else # make a prediction for current increment:
            pred_dy = β[1] + β[2] * last_y + β[3] * z[t-1]
            if reported[t] == true # if current time point is observed:
                dy = y[t] - last_y
                sqerror += (dy - pred_dy)^2
                contribno += 1.0
                last_y = y[t]
            else # if value at current time point is missing:
                last_y += pred_dy
            end
        end
    end
    return sqerror/contribno # return MSE over all observed time points
end
```

→ If you can program it, you can estimate the model parameters

Zygote

```julia
using Zygote

β = [0.0; 0.0; 0.0]
for steps in 1:nsteps
    ∇loss = gradient(arg -> loss(y, z, r, arg), β)[1]
    β = .-= η .* ∇loss
end
```

*as long as the function is differentiable

Results: Comparing prediction performance

Prediction model	Sum of squared error over all hospitals
Zero model	86.0
Mean model	89.24
Modified mean model	84.34
LOCF + regression	181.07
Increment model	74.75

$$\widehat{\mathrm{d}y_T}^{\text{zero}} := 0$$

$$\widehat{\mathrm{d}y_T}^{\text{modmean}} := \begin{cases} 0, & \text{if } (y_{T-1} - y_{T-2}) = 0, \\ \widehat{\mathrm{d}y_T}^{\text{mean}}, & \text{else.} \end{cases}$$

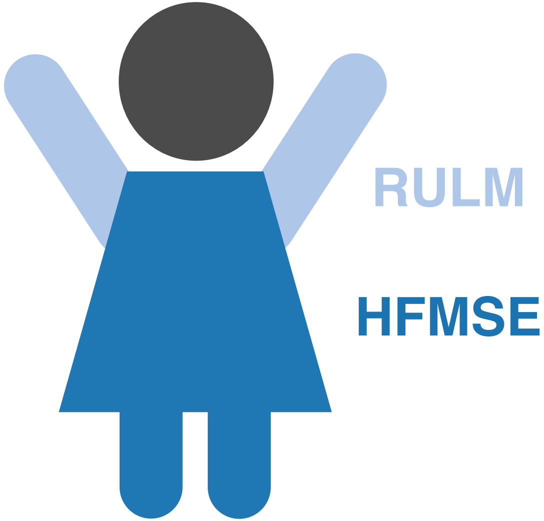
$$\widehat{\mathrm{d}y_T}^{\text{mean}} := \frac{1}{T-2} \sum_{t=2}^{T-1} (y_t - y_{t-1})$$

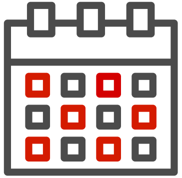
$$\widehat{\mathrm{d}y_T}^{\text{linreg}} := \beta_1 + \beta_2 \cdot \tilde{y}_t + \beta_3 \cdot z_t \text{ where } \tilde{y}_t = \begin{cases} y_t, & \text{if } y_t \text{ was observed,} \\ \tilde{y}_{t-1}, & \text{else.} \end{cases}$$

Modelling challenge II: Learn disease trajectories of patients with spinal muscular atrophy

Baseline characterisation

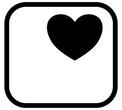
- age
- SMA subtype
- …



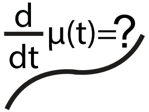


Different motor function tests over time

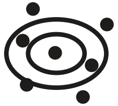
- RULM
- HFMSE
- …



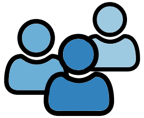
Latent health status



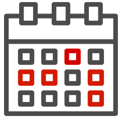
Explicit model



Subgroup-specific local models



Hetero-geneity



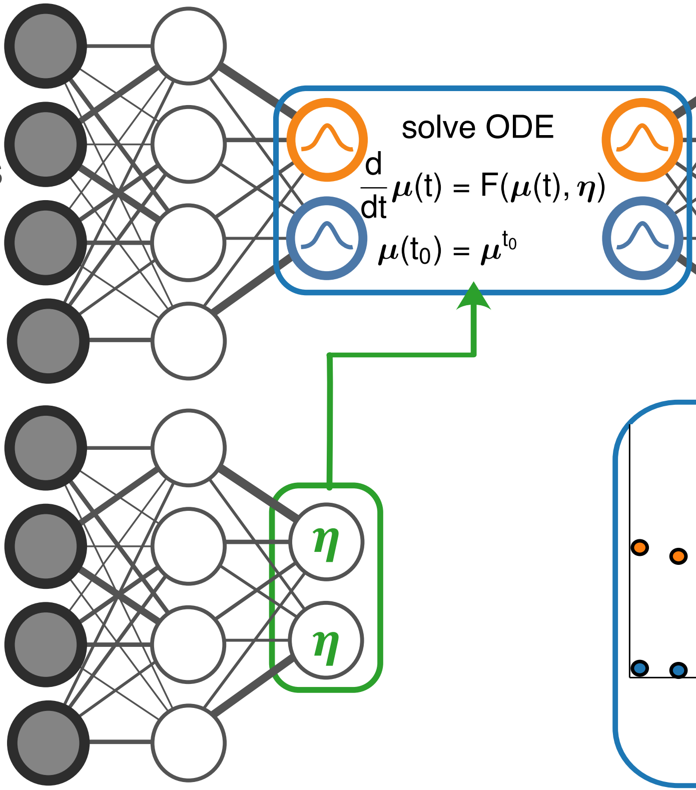
Irregular time points

Different motor function tests

Describe individual SMA trajectories as ODEs in the latent space of a deep learning model

Time series of motor function test

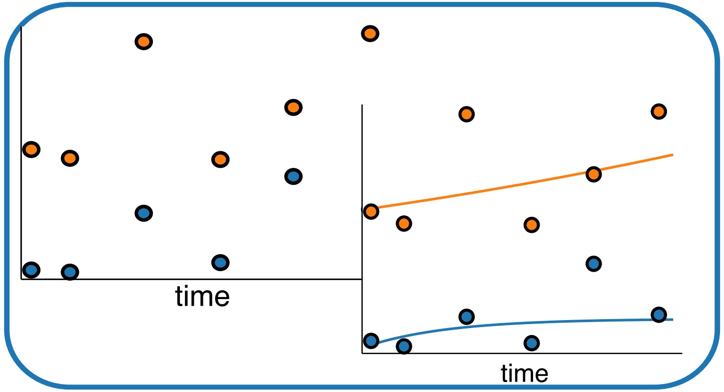
Baseline variables



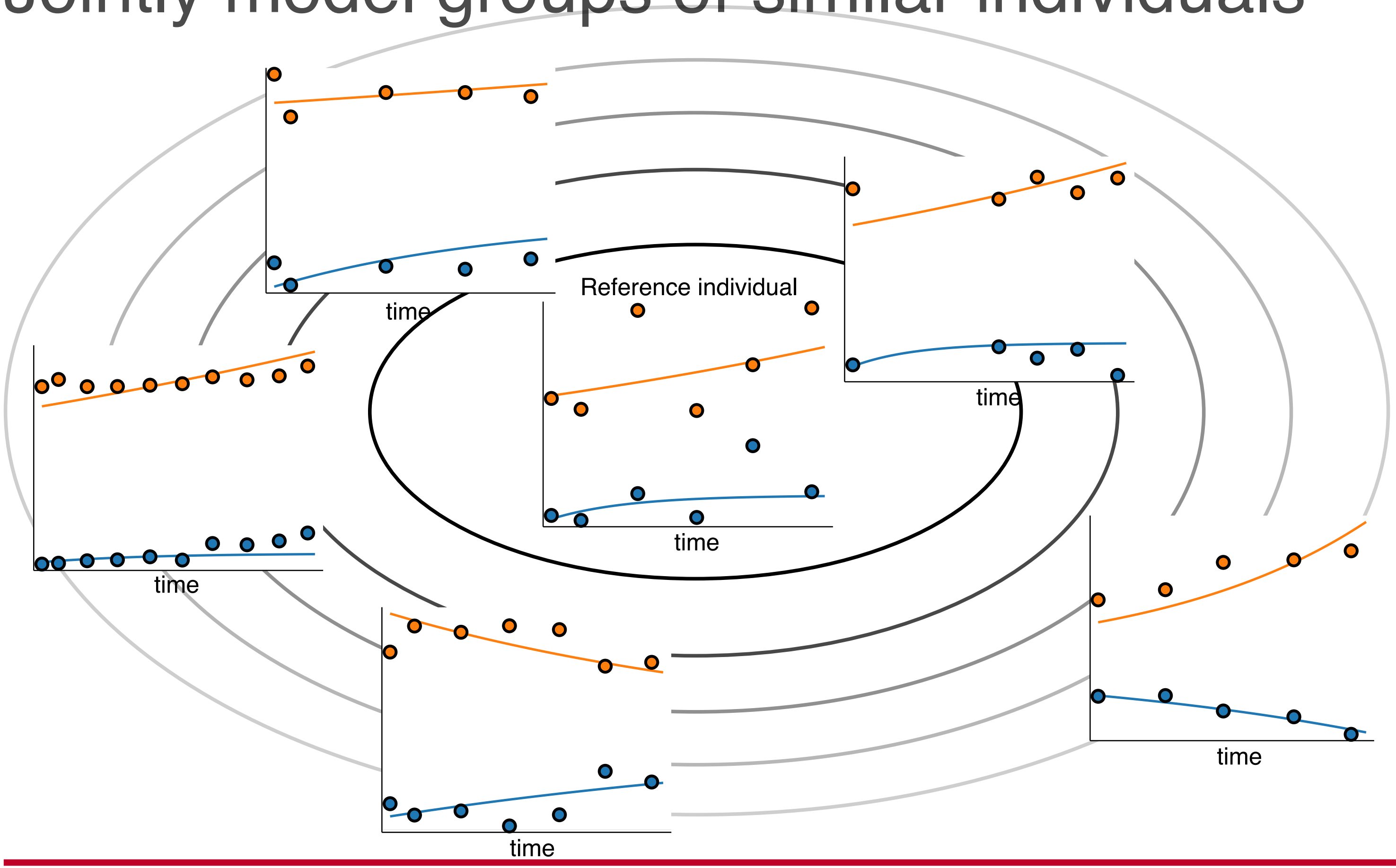
Recon-structed time series

```
m = init_ODEVAE()
ps = getparams(m)
opt = ADAM(η)
trainingdata = zip(xs, xs_baseline, tvals)

for epoch in 1:epochs
    for (X, Y, t) in trainingdata
        grads = gradient(ps) do
            loss(X, Y, t, m, args=args)
        end
        update!(opt, ps, grads)
    end
end
```



Jointly model groups of similar individuals



Julia – a pragmatic approach to scientific computing

Take home messages

- Fresh approach without legacy characteristics
- Fast yet easy to use
- Removes barriers that so far limit scientific ideas, e.g., by providing straightforward access to powerful automatic differentiation libraries
- Key tool for current trends such as differentiable programming:
  - Combining deep learning and dynamic modelling for individual disease trajectories
  - Quick prototyping of a prediction model in a COVID-19 application

➢ Add Julia to your toolbox to expand the universe of modelling approaches available to your research!

Thanks to…

Special thanks to:

Harald Binder

and the AG Machine Learning

Colleagues at the IMBI:

Michelle Pfaffenlehner

Clemens Kreutz

Marlon Grodd

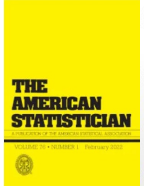
… at the Institute of Mathematics:
Philipp Harms, Thorsten Schmidt

… and from the SMArtCARE registry:
Astrid Pechmann, Janbernd Kirschner

And thanks to you for listening!

If you want to know more:

maren@imbi.uni-freiburg.de



Statistical Practice

Using Differentiable Programming for Flexible Statistical Modeling

Maren Hackenberg, Marlon Grodd, Clemens Kreutz, Martina Fischer, Janina Esins, Linus Grabenhenrich, Christian Karagiannidis & Harald Binder ...show less

Received 07 Dec 2020, Accepted 17 Oct 2021, Accepted author version posted online: 09 Nov 2021, Published online: 21 Dec 2021

Download citation https://doi.org/10.1080/00031305.2021.2002189 Check for updates

RESEARCH ARTICLE Open Access

Deep dynamic modeling with just two time points: Can we still allow for individual trajectories?

Maren Hackenberg, Philipp Harms, Michelle Pfaffenlehner, Astrid Pechmann, Janbernd Kirschner, Thorsten Schmidt, Harald Binder

First published: 06 April 2022 | https://doi.org/10.1002/bimj.202000366

https://doi.org/10.1080/00031305.2021.2002189

https://doi.org/10.1002/bimj.202000366

https://github.com/maren-ha/DifferentiableProgrammingForStatisticalModeling

https://github.com/maren-ha/DeepDynamicModelingWithJust2TimePoints