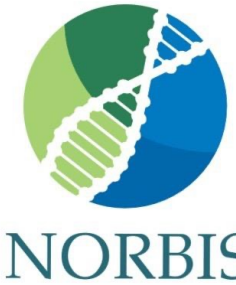


Differentiable programming for flexible modelling with small data

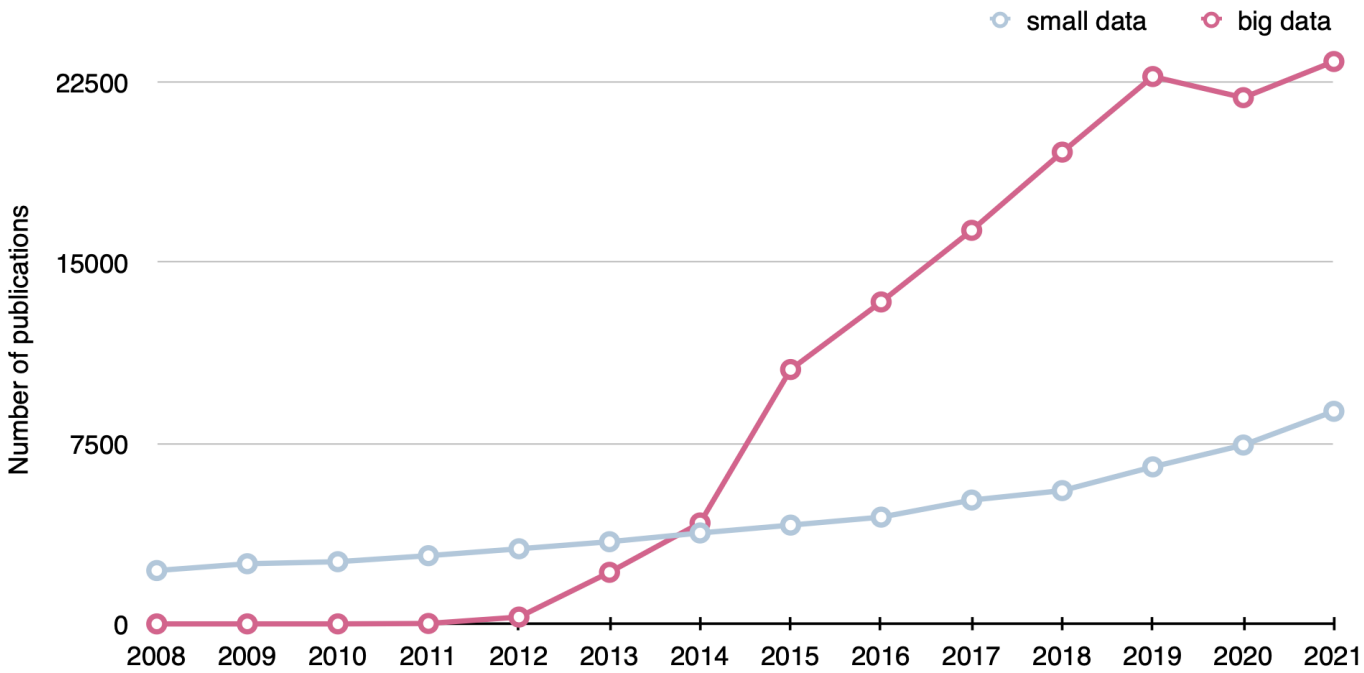
Maren Hackenberg

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

8th Annual NORBIS Conference, Rosendal

Small data vs. big data

Web of Science search



Small data vs. big data

nature medicine

Explore content ⌄ About the journal ⌄ Publish with us ⌄

nature › nature medicine › articles › article

Article | Published: 13 August 2018

Clinically applicable deep learning for diagnosis and referral in retinal disease

Jeffrey De Fauw, Joseph R. Ledsam, Bernardino Romera-Paredes, Stanislav Nikolov, Nenad Tomasev, Sam Blackwell, Harry Askham, Xavier Glorot, Brendan O'Donoghue, Daniel Visentin, George van den Driessche, Balaji Lakshminarayanan, Clemens Meyer, Faith Mackinder, Simon Bouton, Kareem Ayoub, Reena Chopra, Dominic King, Alan Karthikesalingam, Cían O. Hughes, Rosalind Raine, Julian Hughes, Dawn A. Sim, Catherine Egan, ... Olaf Ronneberger

The volume and complexity of diagnostic imaging is increasing at a pace faster than the availability of human expertise to interpret it. Artificial intelligence has shown great promise in classifying two-dimensional photographs of some common diseases and typically relies on databases of millions of annotated images. Until now, the challenge of reaching the performance of expert clinicians in a real-world clinical pathway with three-dimensional diagnostic scans has remained unsolved. Here, we apply a novel deep learning architecture to a clinically heterogeneous set of three-dimensional optical coherence tomography scans from patients referred to a major eye hospital. We demonstrate performance in making a referral recommendation that reaches or exceeds that of experts on a range of sight-threatening retinal diseases after training on only 14,884 scans. Moreover, we demonstrate that the tissue segmentations produced by our architecture act as a device-independent representation; referral accuracy is maintained when using tissue segmentations from a different type of device. Our work removes previous barriers to wider clinical use without prohibitive training data requirements across multiple pathologies in a real-world setting.

ELSEVIER

Statistics & Probability Letters

Volume 136, May 2018, Pages 142-145

When small data beats big data

Julian J. Faraway, Nicole H. Augustin

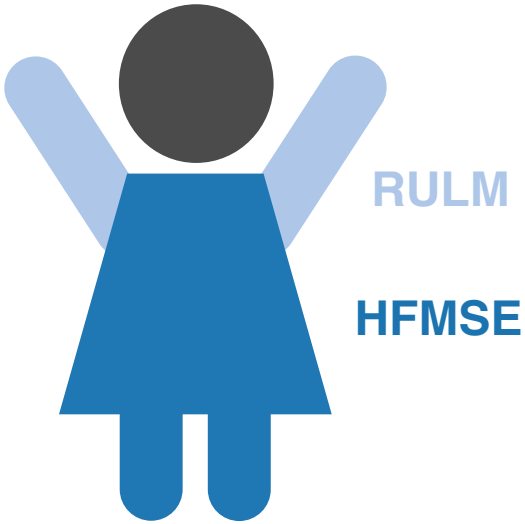
Small data is sometimes preferable to big data. A high quality small sample can produce superior inferences to a low quality large sample. Data has acquisition, computation and privacy costs which require costs to be balanced against benefits. Statistical inference works well on small data but not so well on large data. Sometimes aggregation into small datasets is better than large individual-level data. Small data is a better starting point for teaching of Statistics.

An exemplary small data challenge: Learn disease trajectories of patients with spinal muscular atrophy

SMARTCARE

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

RULM
HFMSE

Different motor function tests

UNIVERSITÄTS KLINIKUM FREIBURG

How could we tackle such a modelling problem?

- Small data challenges require dedicated methods
- „differentiable programming“ represents an approach to modelling that is promising for small data
- We’ll get deeper into that!
  - Step 1: How it looks like – demo (Colab notebook Part I)
  - Step 2: How it really works – blogpost + discussion
  - Step 3: How to use it for our problem – hands-on (Colab notebook Part II)

Differentiable programming: demo in Google Colab

- Time to start your own notebook (as first install + precompile takes ~15 minutes)!
- Please check out the instructions in the workshop Github repository: https://github.com/maren-ha/NORBIS_workshop_differentiable_programming

a) using Google Colab (recommended)

- ***Prerequisites:*** You need a Google account (and a small bit of free space on your Google Drive).
- ***Pros:*** Requires no local installation of Julia, no OS-dependent intricacies when installing packages etc.
- ***Cons:*** You have to re-install Julia and all the required packages again whenever your Colab runtime crashes or restarts, which will take some time (~ 5-10 minutes).
- ***Instructions:***
  - get a copy of the notebook to your Google Drive: Open the notebook... Open in Colab
  - ... and create a copy in your Google Drive

> ⚡ **Very important** ⚡ Please make a copy of the notebook **straight away** or upload the `Colab_Practicals.ipynb` file from the repository to your Google Drive, so that you have a copy in your own Google Drive! Otherwise you will not be able to save it and any changes you do in the notebook will be lost after you close the tab.

- After you have copied the notebook to your drive:
  - run the first cell to install the Julia kernel
  - re-load the page (by clicking on the ↩ icon in the address line of the browser or pressing Ctrl + R)
  - upload the `Project.toml` file from this repo to your Google Drive (Click on "Files" in the left menu, then on the upload icon, then select the file)
  - proceed by running the next cells, which check the installation and install the package dependencies

Differentiable programming: Blogpost

Reading + discussion

What Is Differentiable Programming?

> *With four parameters I can fit an elephant, and with five I can make him wiggle his trunk.*
> *– John Von Neumann*

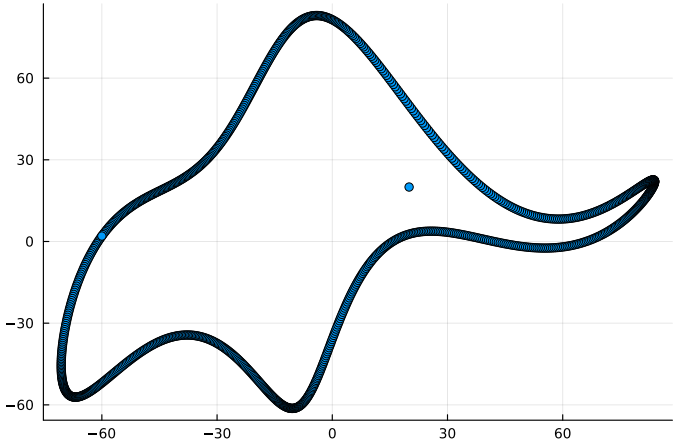
The idea of "differentiable programming" is coming up a lot in the machine learning world. To many, it's not clear if this term reflects a real shift in how researchers think about machine learning, or is just (another) rebranding of "deep learning". This post clarifies what new things differentiable programming (or ∂P) brings to the machine learning table.

Most importantly, differentiable programming is actually a shift *opposite* from the direction taken by deep learning; from increasingly heavily parameterised models to simpler ones that take more advantage of problem structure.

… btw you *can* actually fit an elephant with 4 parameters …

```julia
# elephant parameters
p1, p2, p3, p4 = (50 - 30im, 18 +  8im, 12 - 10im, -14 - 60im)
p5 = 40 + 20im # eyepiece

function fourier(t,C)
    f = zeros(size(t))
    A, B = real(C), imag(C)
    for k in 1:length(C)
        f .+= A[k].*cos.(k .*t) + B[k].*sin.(k .*t)
    end
    return f
end

function elephant(t, p1, p2, p3, p4, p5)
    npar = 6
    Cx = zeros(Complex, npar)
    Cy = zeros(Complex, npar)

    Cx[1] = real(p1)*1im
    Cx[2] = real(p2)*1im
    Cx[3] = real(p3)
    Cx[5] = real(p4)

    Cy[1] = imag(p4) + imag(p1)*1im
    Cy[2] = imag(p2)*1im
    Cy[3] = imag(p3)*1im

    x = push!(fourier(t,Cx), -imag(p5))
    y = push!(fourier(t,Cy), imag(p5))

    return x,y
end

x, y = elephant(collect(range(0, 2pi, length=1000)), p1, p2, p3, p4, p5)
plot(y, -x, seriestype = :scatter, legend=false) | Plot{Plots.GRBackend() n=1}
```

… produces…



AAPT American Journal of Physics

HOME BROWSE INFO FOR AUTHORS COLLECTIONS AAPT Books

Home > American Journal of Physics > Volume 78, Issue 6 > 10.1119/1.3254017

Full • Submitted: 20 August 2008 • Accepted: 05 October 2009 • Published Online: 12 May 2010

Drawing an elephant with four complex parameters

American Journal of Physics **78**, 648 (2010); https://doi.org/10.1119/1.3254017

Jürgen Mayer
• Max Planck Institute of Molecular Cell Biology and Genetics, Pfotenhauerstr. 108, 01307 Dresden, Germany
Khaled Khairy
• European Molecular Biology Laboratory, Meyerhofstraße. 1, 69117 Heidelberg, Germany
Jonathon Howard
• Max Planck Institute of Molecular Cell Biology and Genetics, Pfotenhauerstr. 108, 01307 Dresden, Germany

UNIVERSITÄTS KLINIKUM FREIBURG

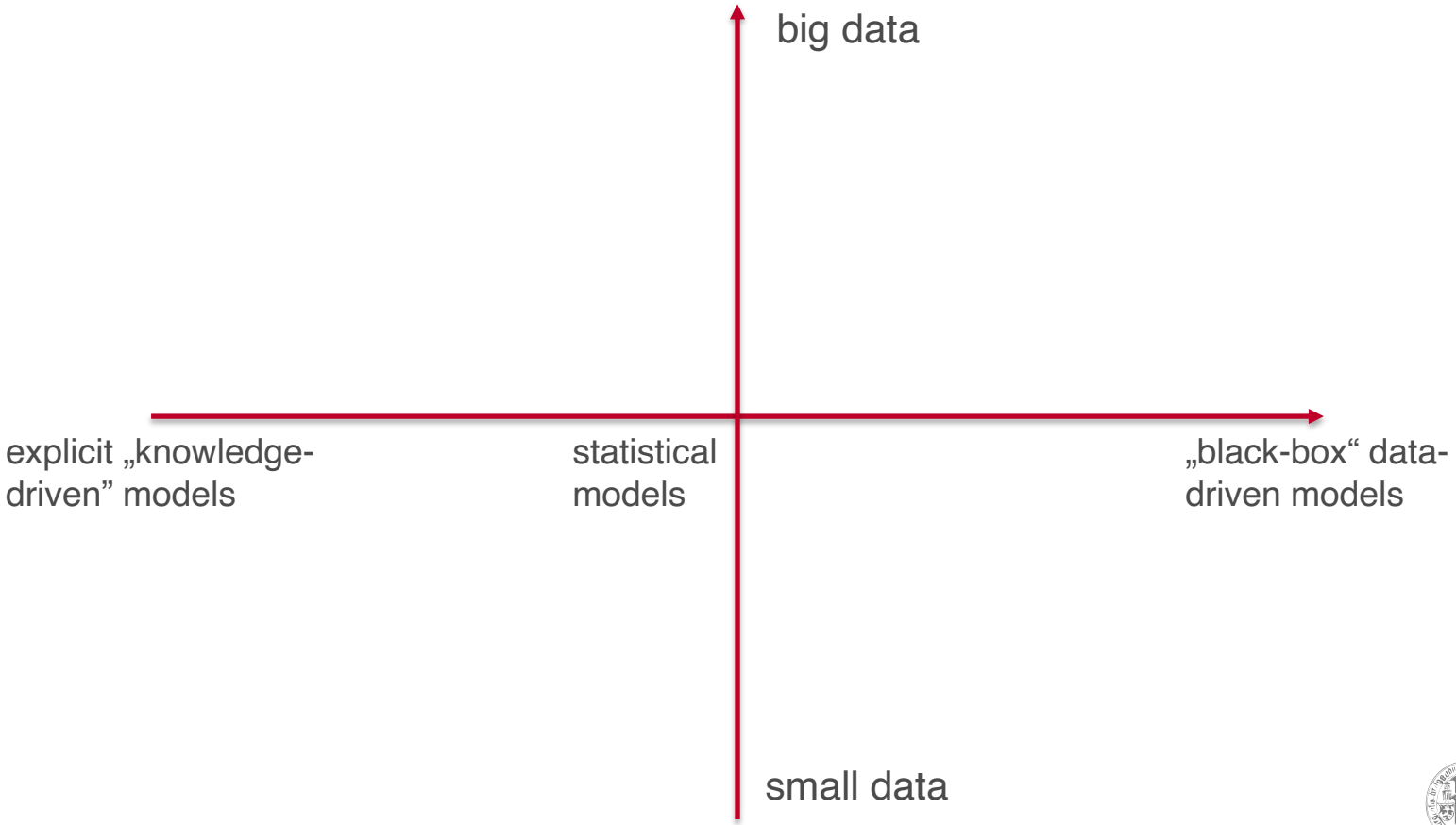
Reading + group discussion (45 minutes)

Please discuss in groups and collect answers on the digital white board!

- Place a dot on the flipchart coordinate system: I mostly work with … big data – small data; explicit knowledge-driven models (regression, physical models, dynamical systems…) – black-box / data-driven models (random forests, neural networks,…)

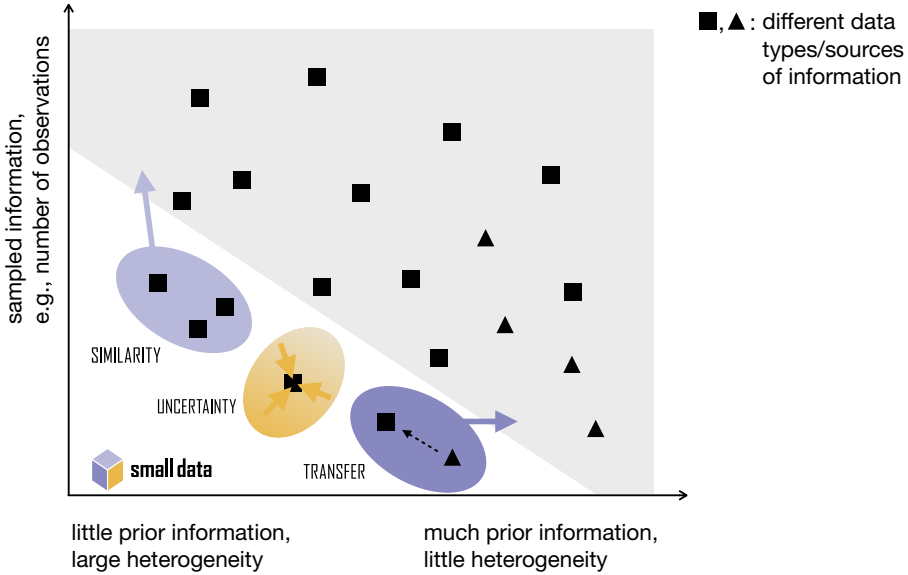
1. Can you think of examples for “big” and “small” datasets in your discipline or research sub-field?
2. How would you define “small data” as opposed to big data, what characteristics would you take into account?
3. What did you find interesting/cool/surprising about the blogpost?
4. Name a statement you didn’t fully understand or didn’t agree with!
5. How does differentiable programming work in practice, what are the core ingredients?
6. Can you think of an example for a task or an application where you would use something like deep learning, and of a task where you prefer explicit domain modeling?

I mostly work with…



Small data

- small data is not only about the number of observations but also the level of hereogeneity, amount of structure, …
- → relative definition
- The specific challenges require a set of dedicated methods



Differentiable programming for flexible model building

Machine Learning

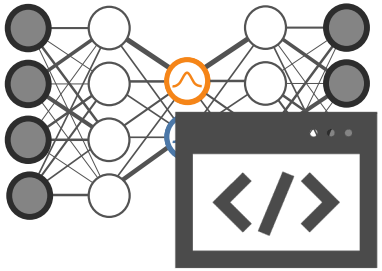
Scientific Computing

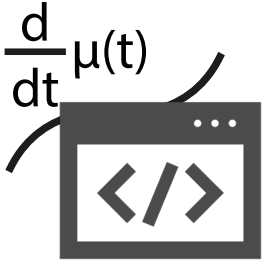
TensorFlow

PyTorch

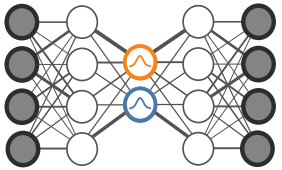
Zygote

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

Bringing it together: combining explicit dynamic modeling and deep learning for a small data problem

Baseline characterisation

- age
- SMA subtype
- …

Different motor function tests over time

- RULM
- HFMSE
- …

Latent health status

Explicit model

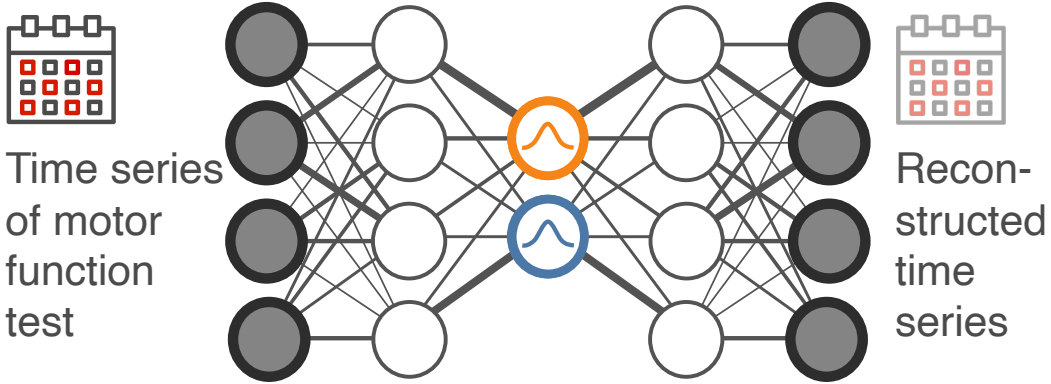
Subgroup-specific local models

Hetero-geneity

Irregular time points

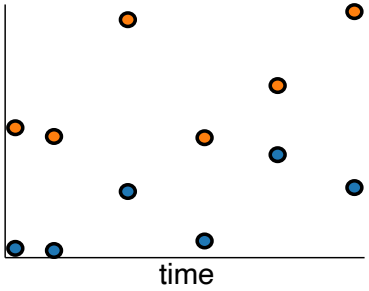
RULM
HFMSE
Different motor function tests

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

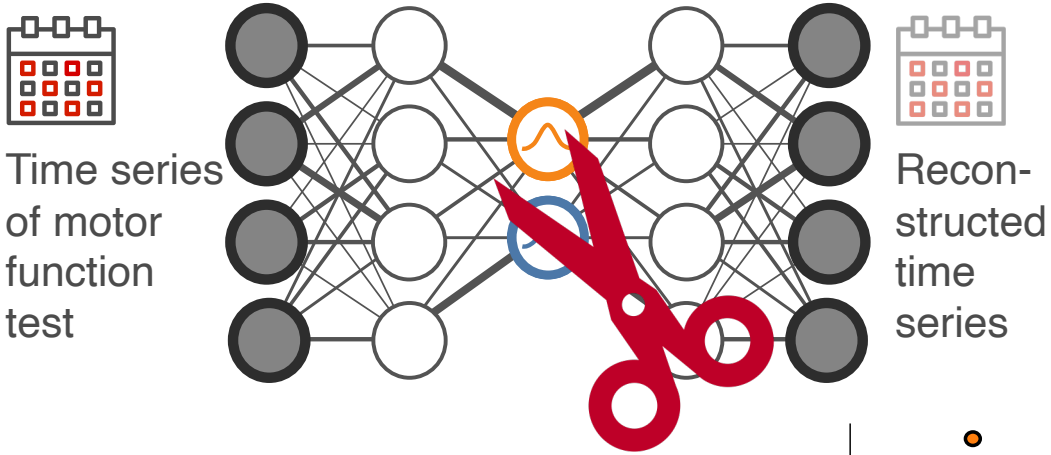


Time series of motor function test

Recon-structed time series

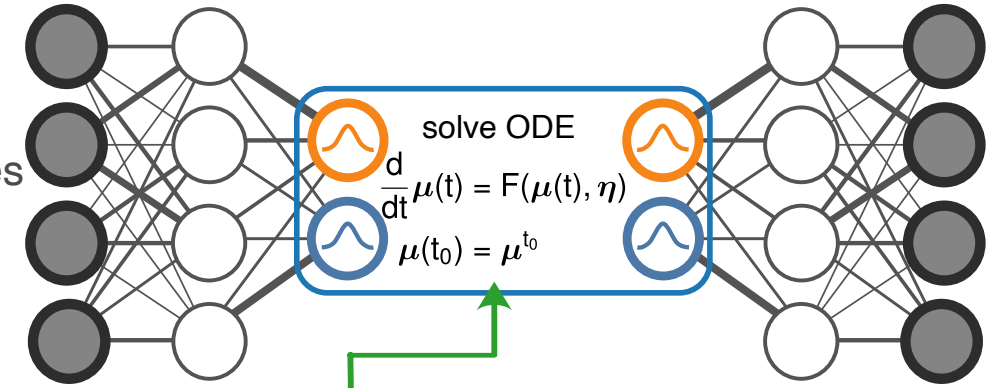


Describe individual SMA trajectories as ODEs in the latent space of a deep learning model



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

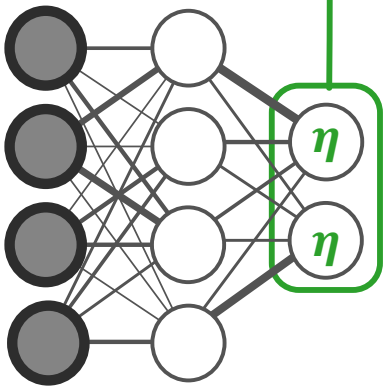
Time series of motor function test

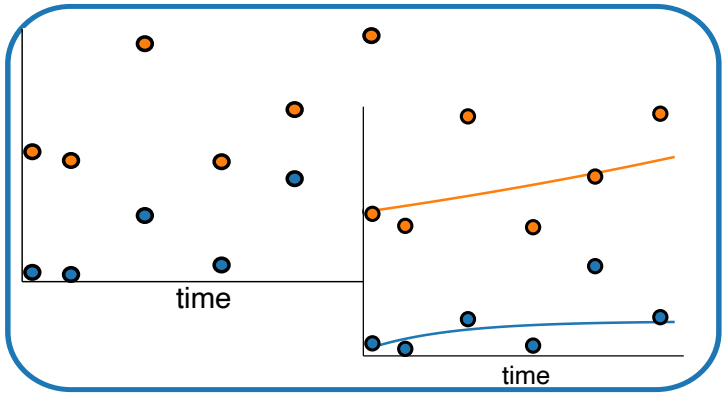


Recon-structed time series

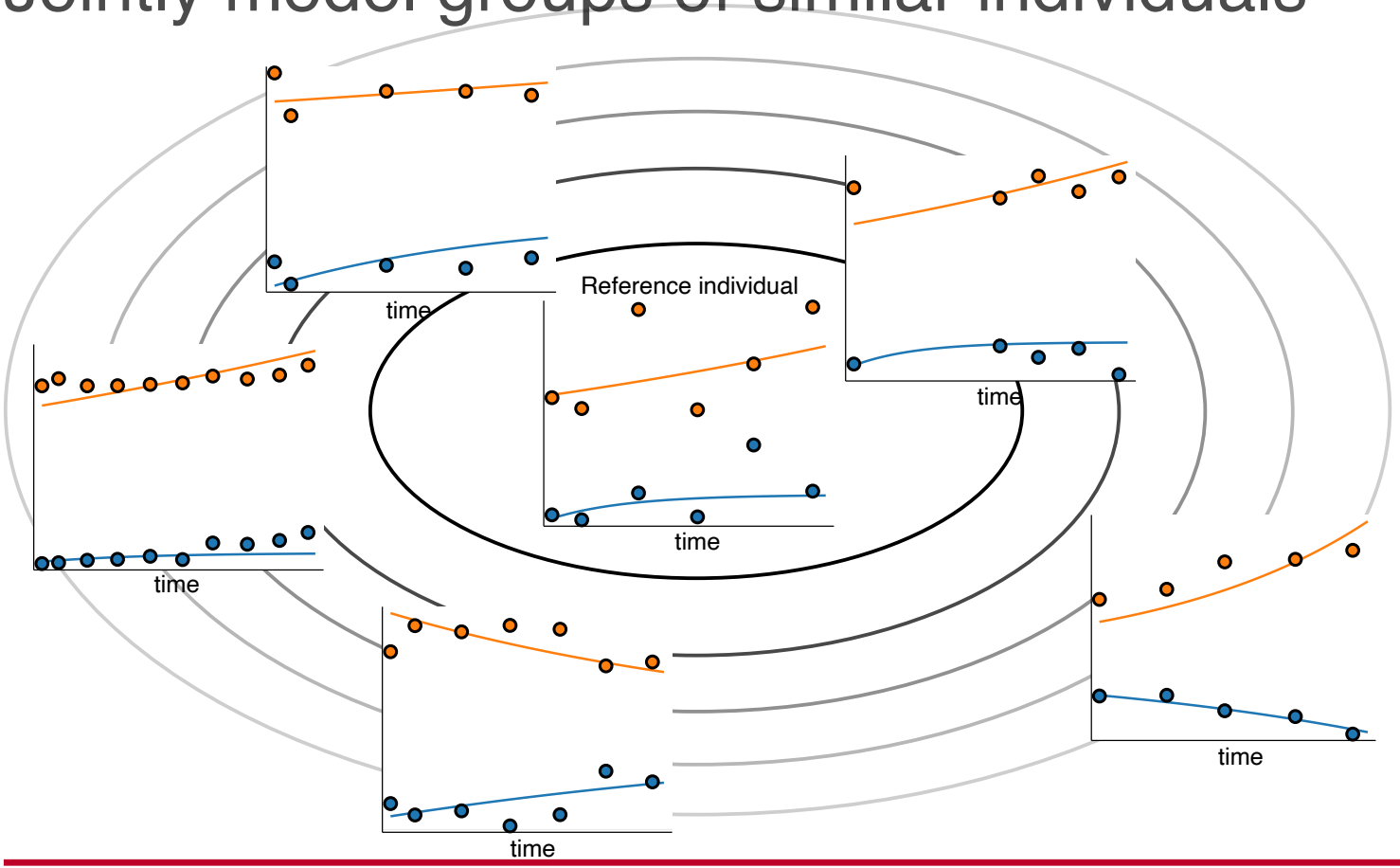
```
m = init_ODEVAE()
ps = getparams(m)
opt = ADAM(η)
trainingdata = zip(xs, xs_baseline, tvals)

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
```

Baseline variables

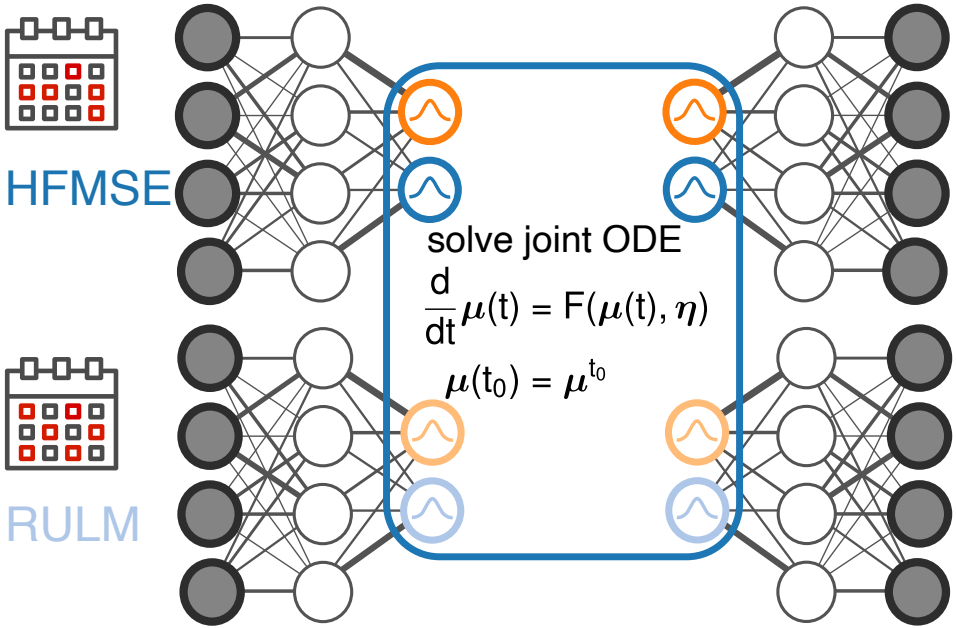


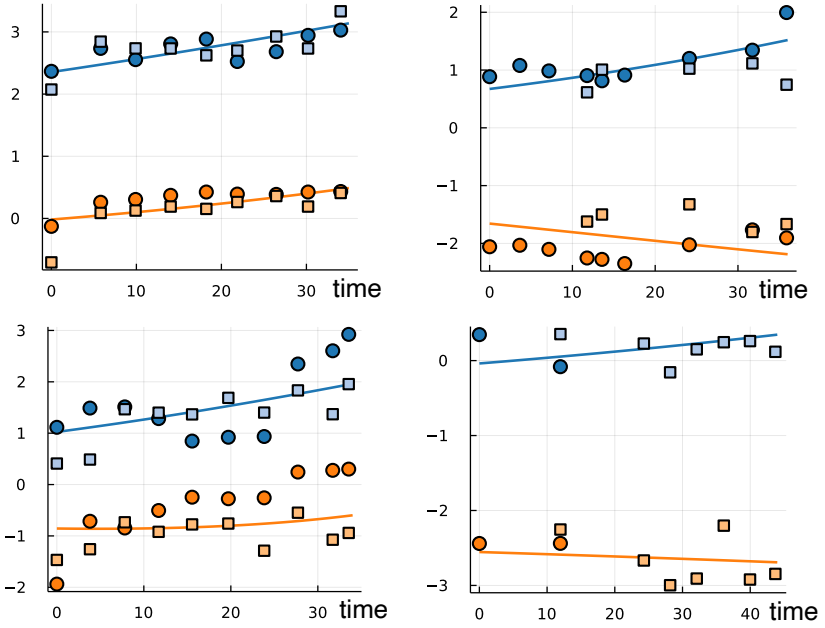


Jointly model groups of similar individuals



Integrating different motor function tests





HFMSE test measurements

RULM test measurements

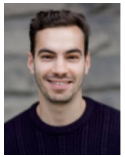
Thanks to…

Special thanks to:

Manuela Zucknick

Harald Binder

and the AG Machine Learning



maren@imbi.uni-freiburg.de

Colleagues at the IMBI and from the SMArtCARE registry:

Eva Brombacher

Michelle Pfaffenlehner

Astrid Pechmann

Janbernd Kirschner

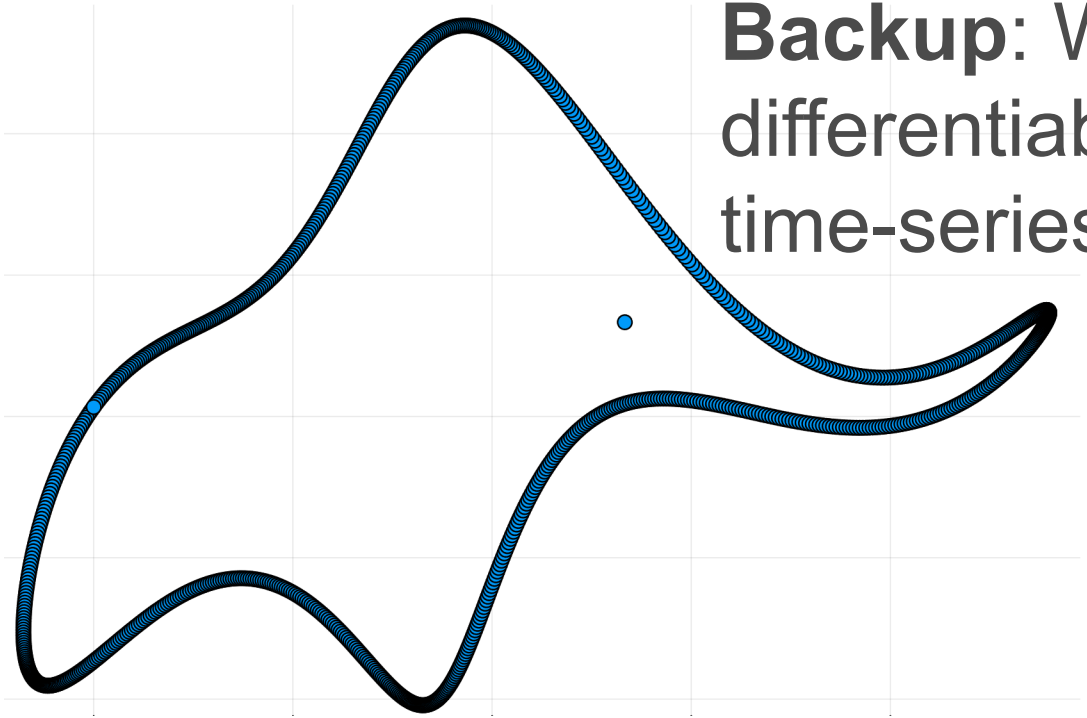
If you want to know more:

https://doi.org/10.1080/00031305.2021.2002189

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

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

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

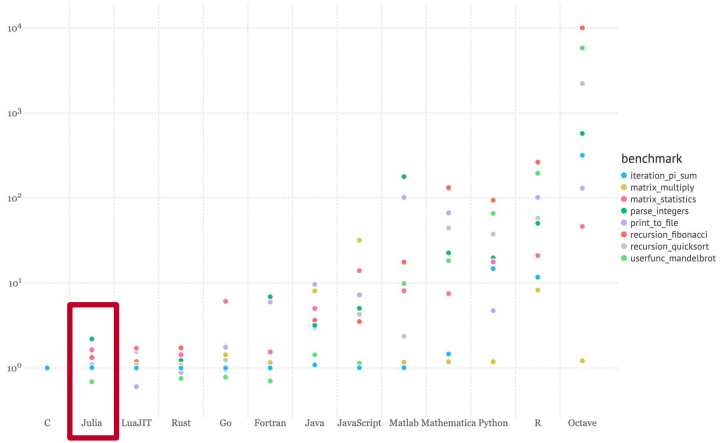


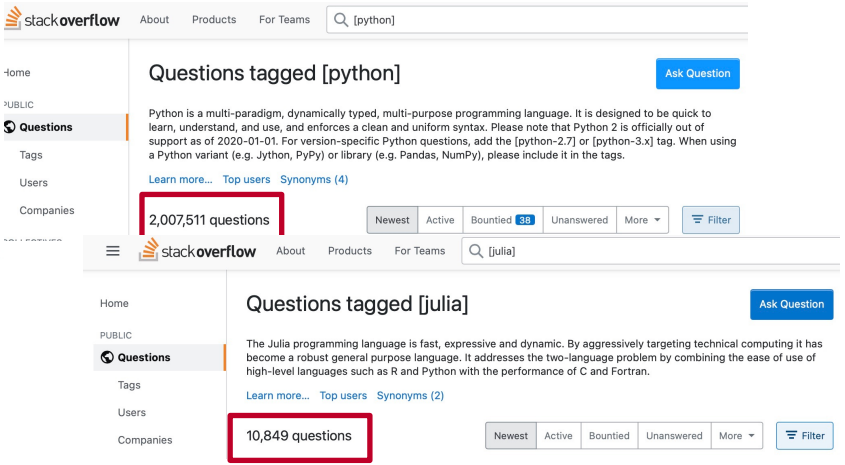
Backup: Why Julia? + differentiable programming for time-series single-cell RNA-seq and statistical modelling

…so why Julia?

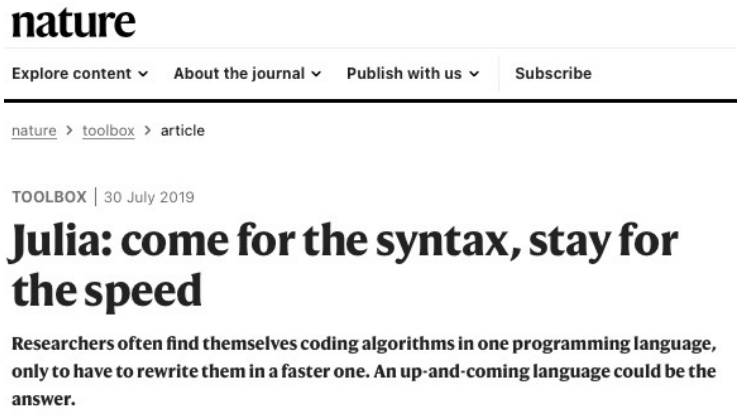
- Fast yet easy to learn with a syntax similar to R and Python
- Everything is pure Julia
- Easy interfaces to other languages

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

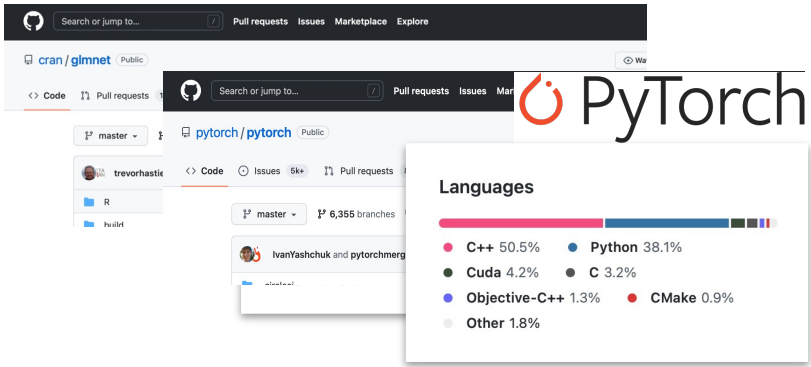


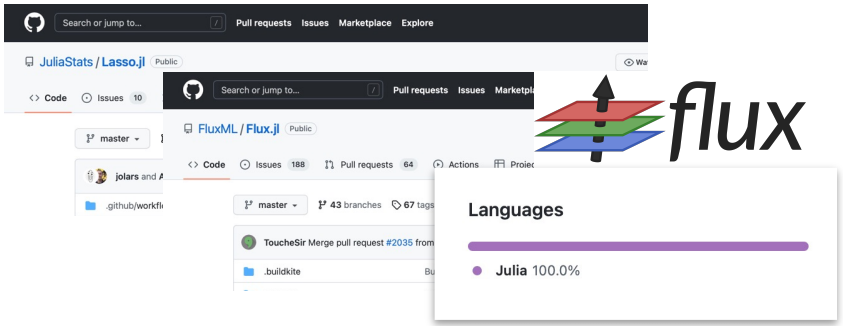


Julia doesn’t constrain your thinking

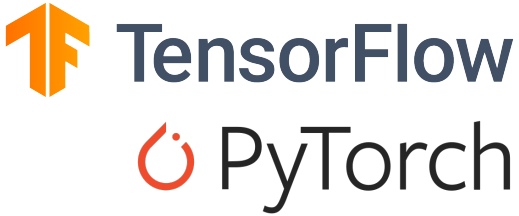


- Complex algorithms can be implemented in the language itself
- Modification of existing libraries is easy, enabling faster prototyping and methods development
- No “legacy characteristics”





Engineering trade-offs in automatic differentation tools



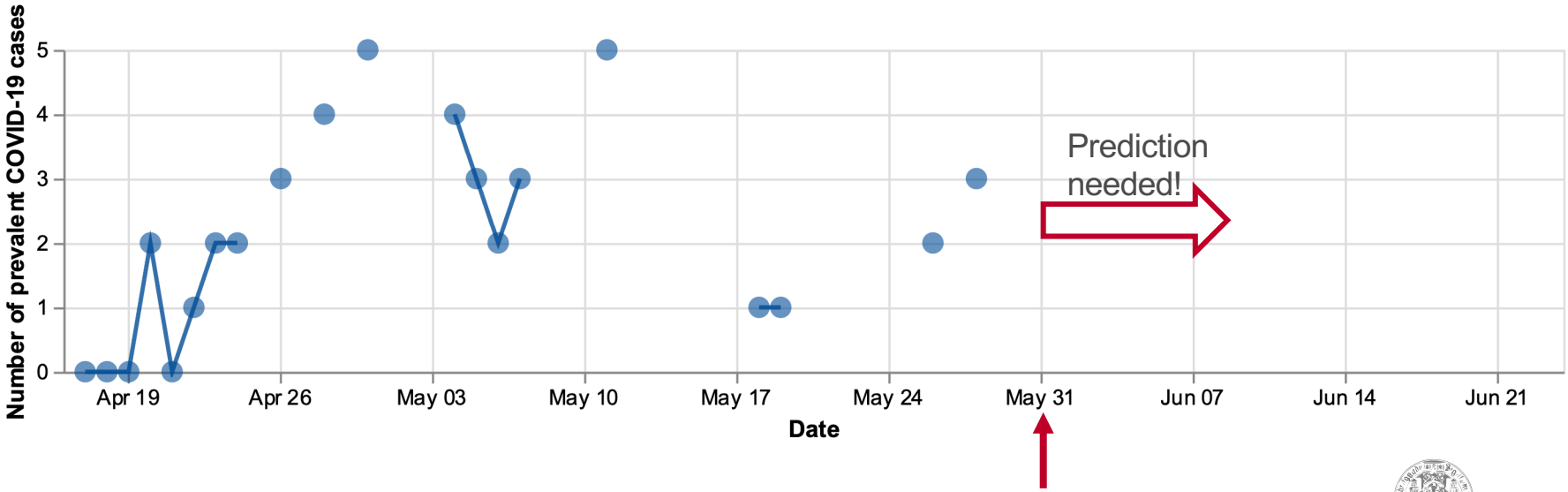
- Graph-building system in static sub-language: efficient but less flexible
- Tape-based method: unrolls all control flow
- Overhead partly compensated by highly efficient kernels for machine learning applications

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

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

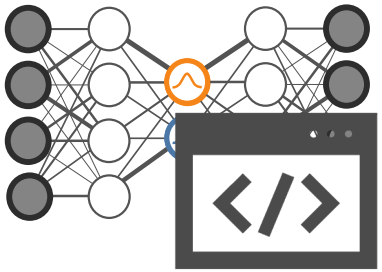
Differentiable programming for flexible model building

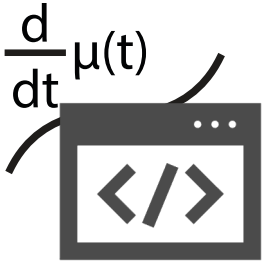
Machine Learning

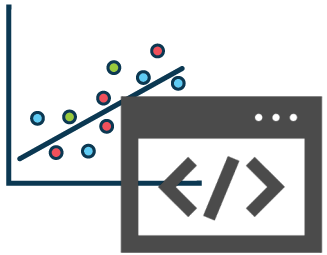
Scientific Computing

Statistical Modelling

write as a computer program



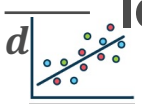




$\frac{d}{d}$  **loss**

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

optimise via automatic differentiation

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

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


```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

$$\widehat{\mathrm{d}y}_T^{\mathrm{zero}} := 0$$

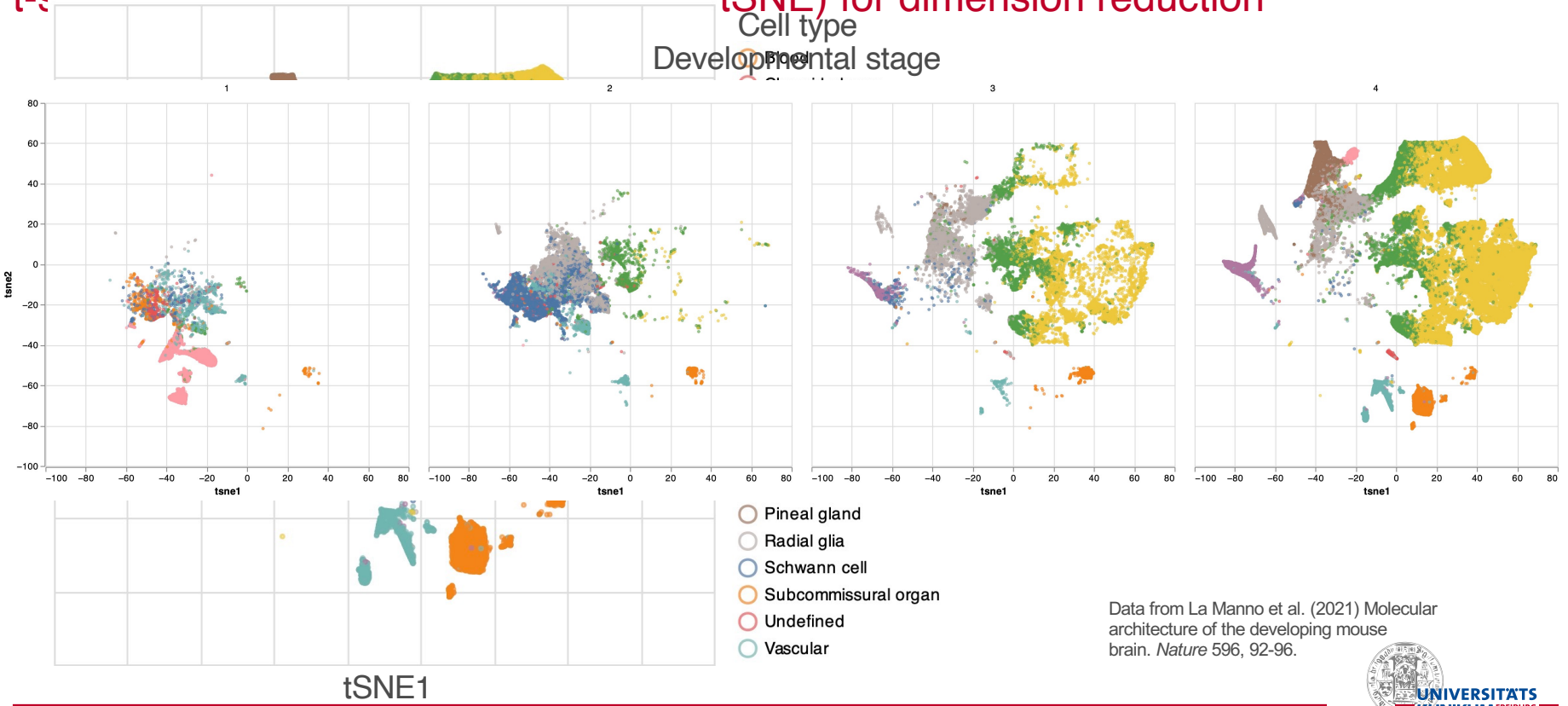
$$\widehat{\mathrm{d}y}_T^{\mathrm{modmean}} := \begin{cases} 0, & \text{if } (y_{T-1} - y_{T-2}) = 0, \\ \widehat{\mathrm{d}y}_T^{\mathrm{mean}}, & \text{else.} \end{cases}$$

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

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

How do you typically look at single-cell RNA-seq data?

t-stochastic neighborhood embedding (tSNE) for dimension reduction



Data from La Manno et al. (2021) Molecular architecture of the developing mouse brain. *Nature* 596, 92-96.

Dimension reduction can be misleading

Data: ex utero mouse embryo E8.5

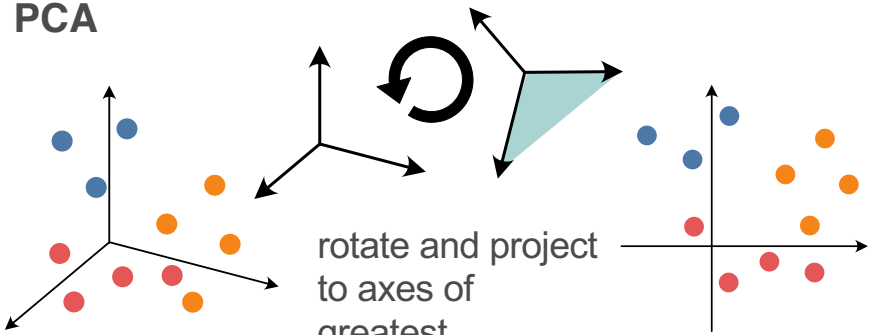
Cell Types

- Neural progenitor cells
- Radial glia
- Erythroid lineage
- Osteoblasts
- Hepatocytes
- Epithelial cells
- Chondrocytes and osteoblasts
- Jaw & tooth progenitors
- Primitive erythroid lineage
- White blood cells
- Cholinergic neurons
- Schwann cell precursors
- Myocytes
- Notochord cells
- Limb mesenchyme

Figures taken from Chari et al. (2021) The specious art of single-cell genomics. *bioRxiv preprint,* doi*:* 10.1101/2021.08.25.457696.

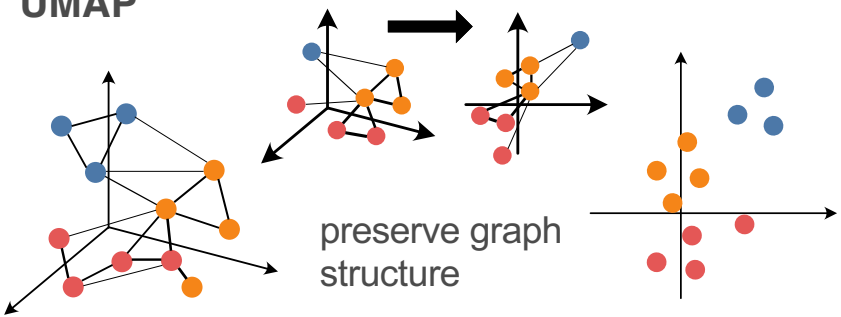
Comparing popular dimension reduction methods

PCA



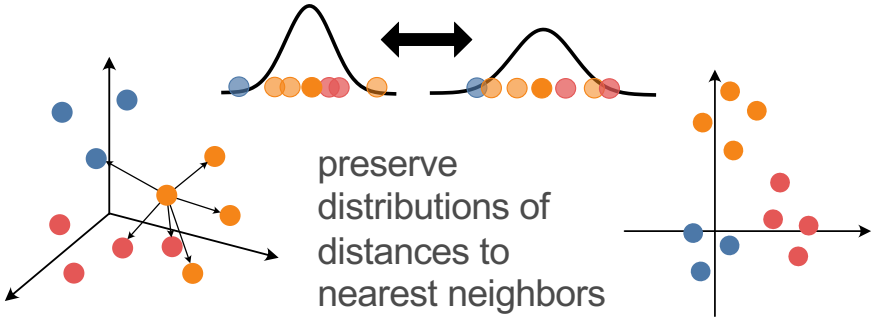
rotate and project to axes of greatest variation

UMAP



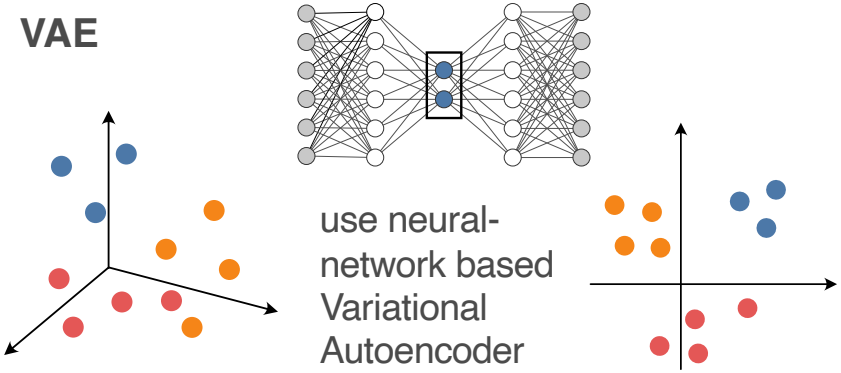
preserve graph structure

tSNE



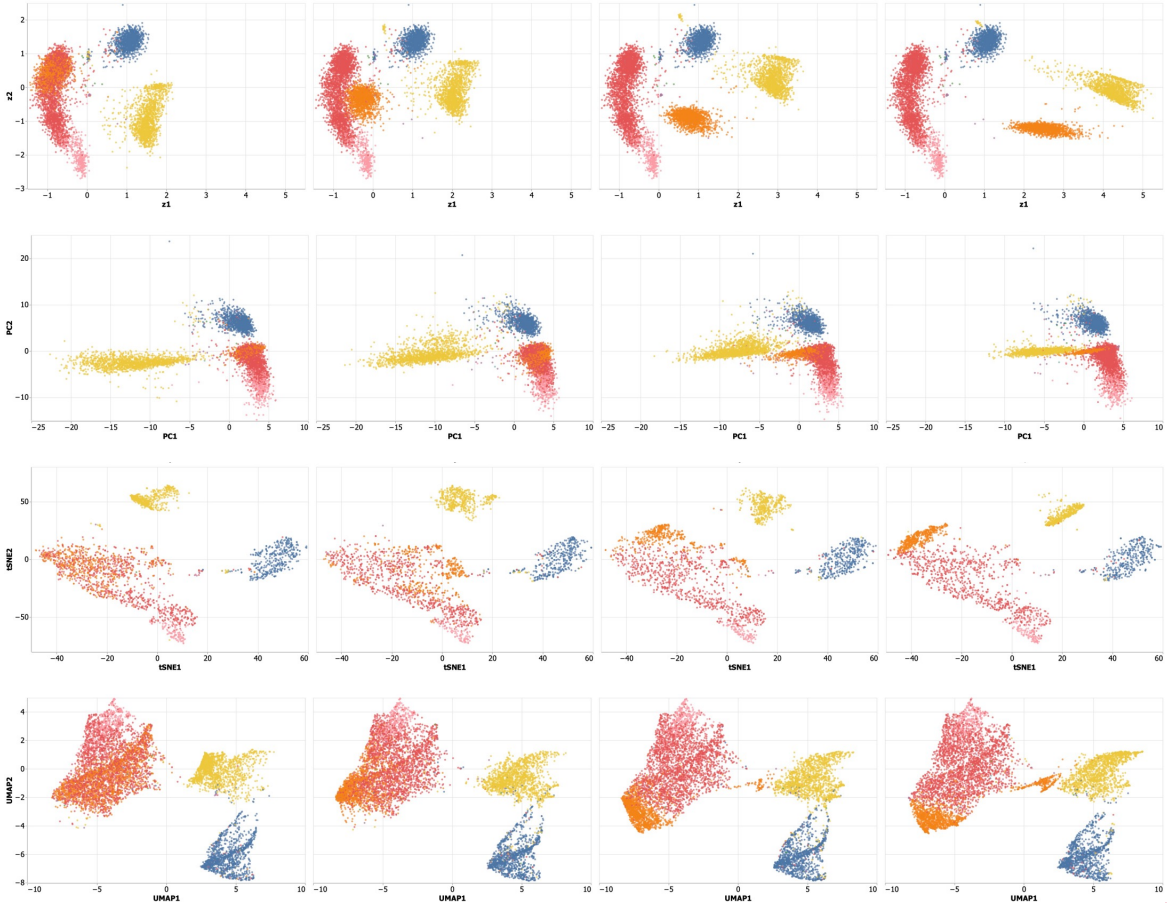
preserve distributions of distances to nearest neighbors

VAE

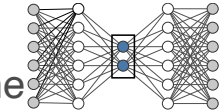


use neural-network based Variational Autoencoder

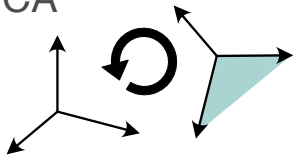
Results: Cluster spreading + rotation



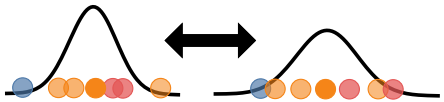
Orignal trans-formation in the VAE latent space



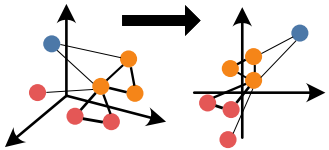
PCA



tSNE



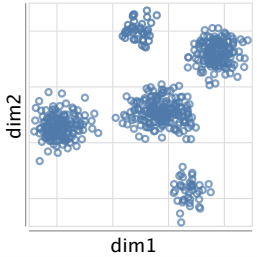
UMAP



Why deep learning?

Neural networks can easily be combined with other explicit models

$x \in \mathbb{R}^p$ → tSNE algorithm → $z \in \mathbb{R}^2$

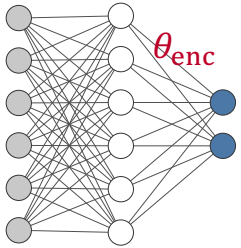


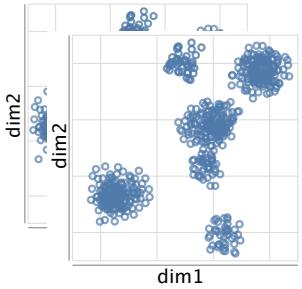
no explicit function f_{tSNE}

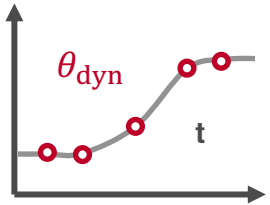
$x \in \mathbb{R}^p$ → VAE encoder f_{enc} → $z \in \mathbb{R}^2$ → dynamic model f_{dyn} → $z^{t_1}, z^{t_2}, \ldots \in \mathbb{R}^2$

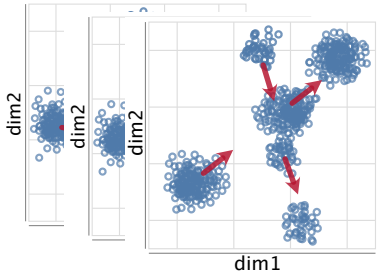
Define $\text{loss}(x; \theta_{\text{enc}}, \theta_{\text{dyn}})$ and minimize w.r.t. $\theta_{\text{enc}}, \theta_{\text{dyn}}$

$\frac{\partial}{\partial \theta_{\text{enc}}} \text{loss}$ ← $\frac{\partial}{\partial \theta_{\text{dyn}}} \text{loss}$ ← $\frac{\partial}{\partial \theta} \text{loss}(x; \theta_{\text{enc}}, \theta_{\text{dyn}})$

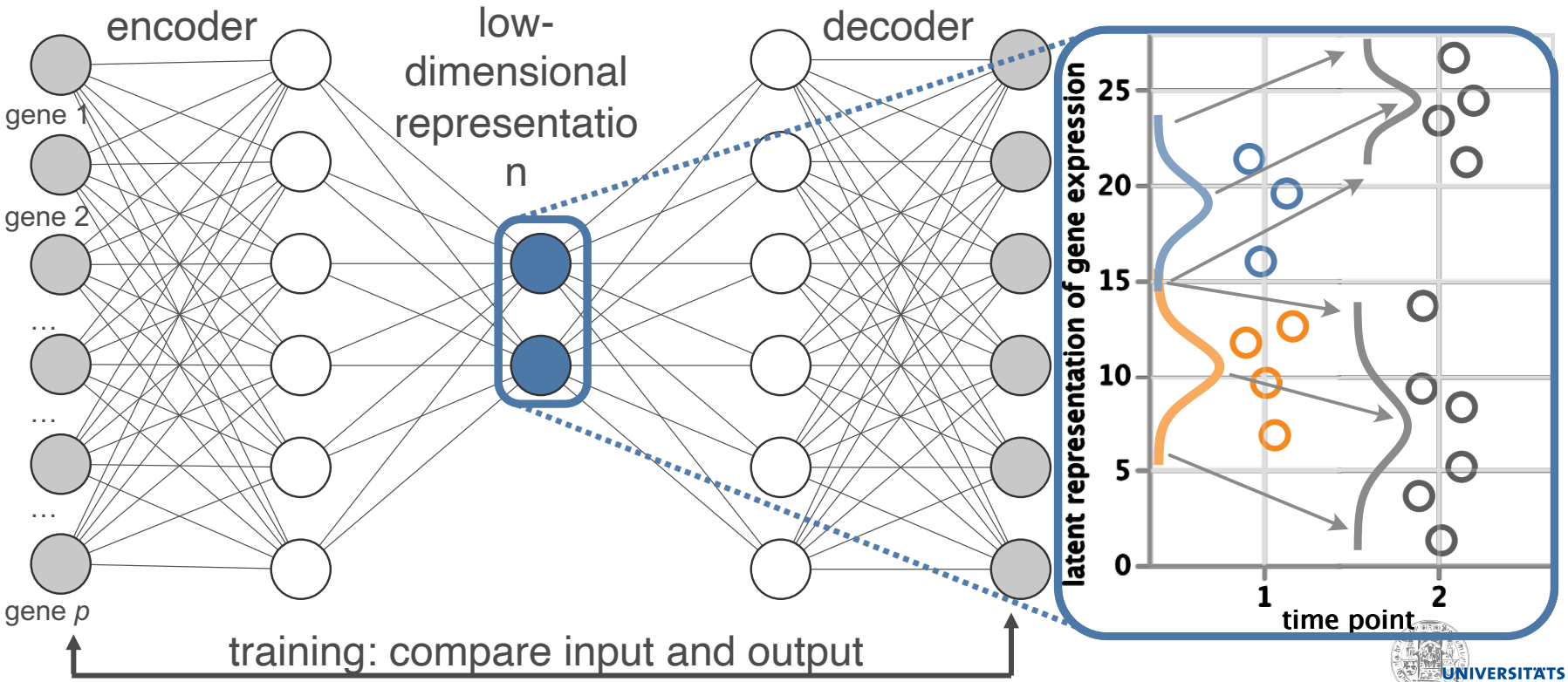






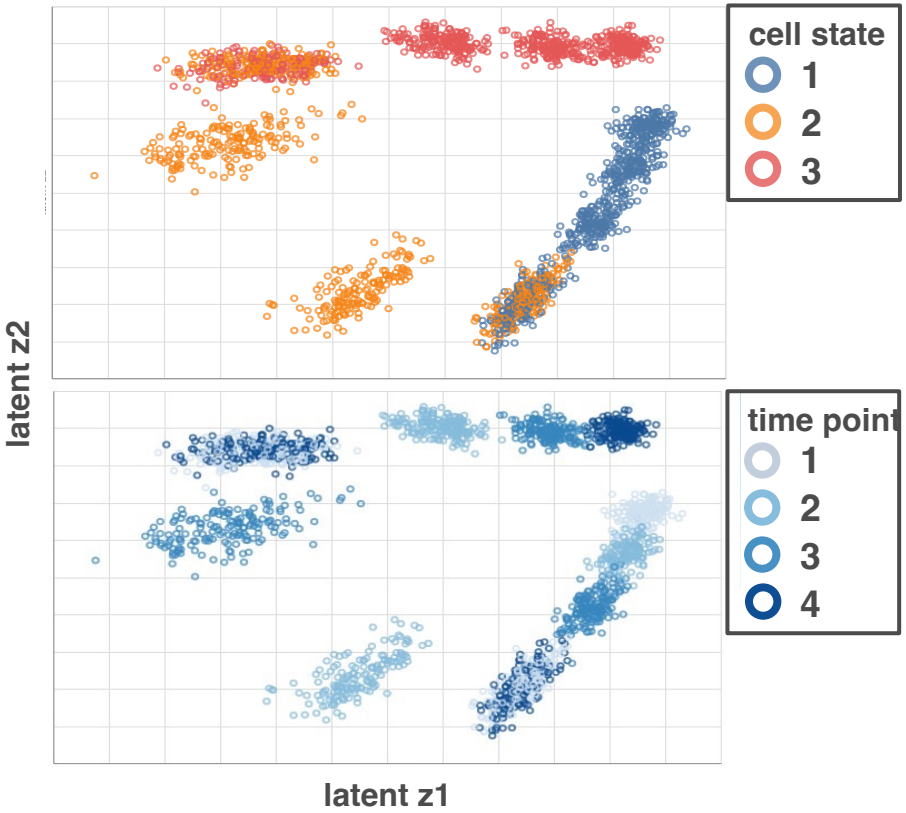


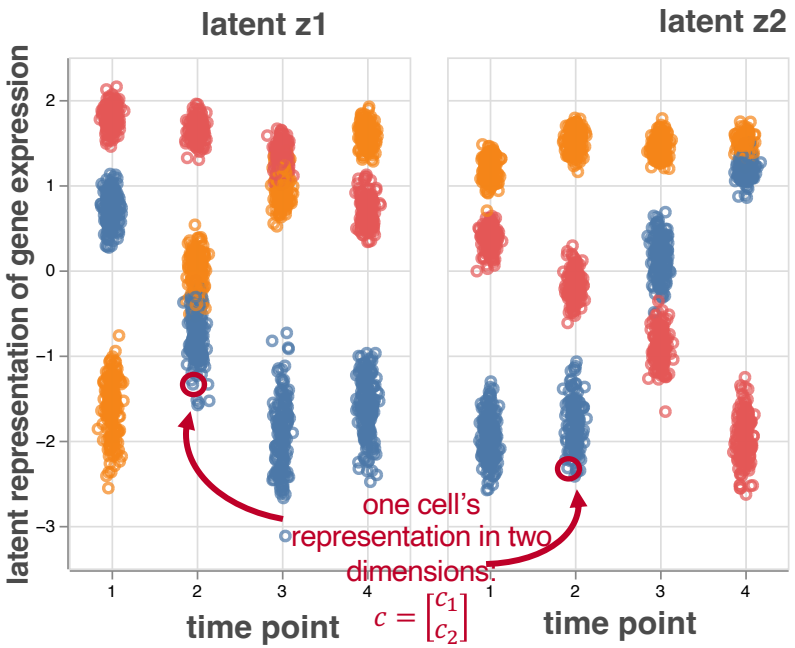
Combining VAEs with dynamic modeling for time-series single-cell RNA-sequencing



Simulating time-series scRNA-seq count data

Count data simulated with the `splatter` R package

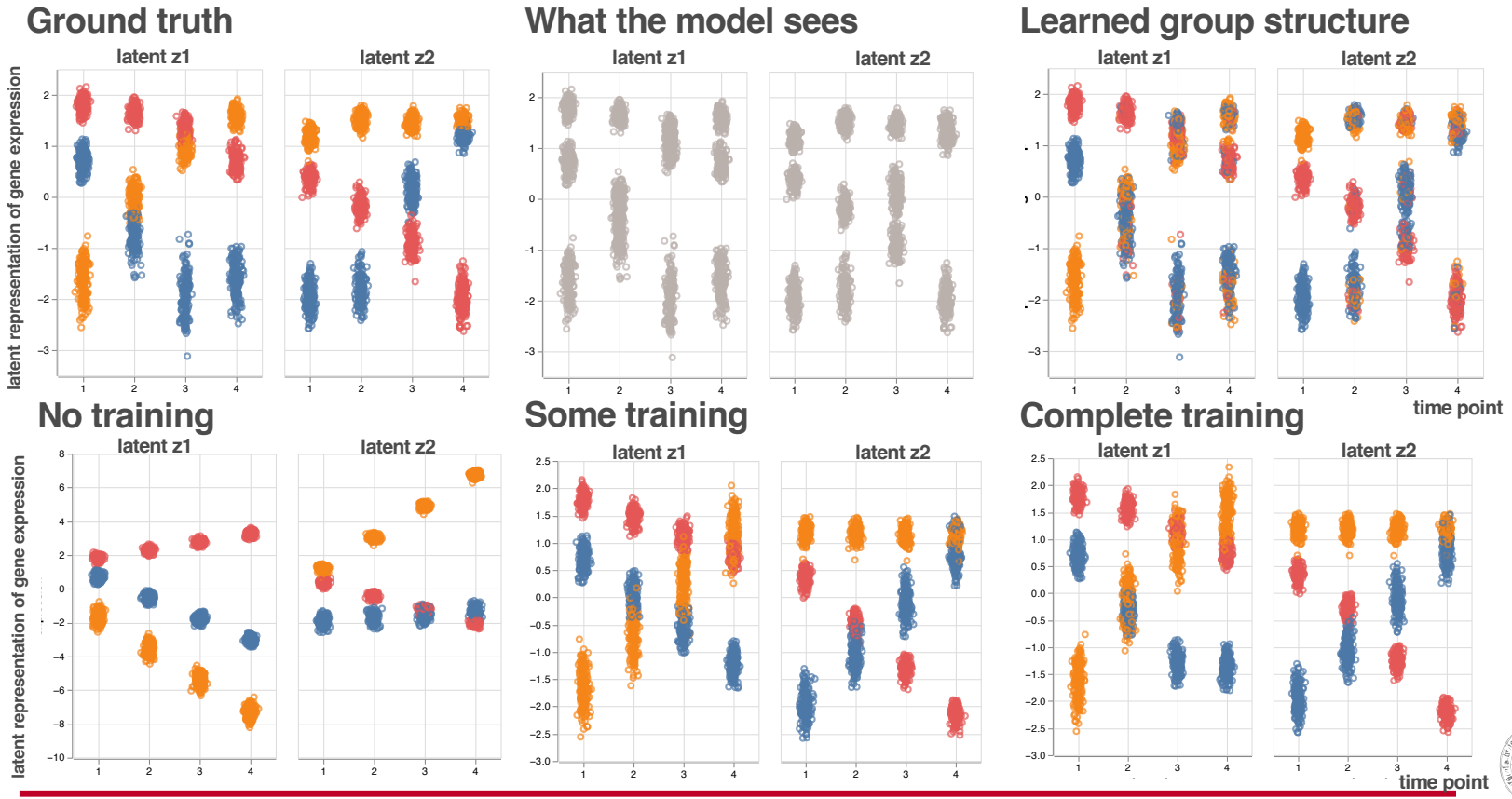




Each cell (circle) appears twice: dimension 1 of its low-dimensional representation in the left panel, dimension 2 in the right panel

Simulation study results

The dynamic model simultaneously infers underlying trajectories and cell states



VAE representation of original data

Model predictions

scVI

Search docs

Getting started

- Overview
- Installation
- Contents

Data processing

The scVAE model

The scLDVAE model

Model training

Model evaluation

Getting started

Edit on GitHub

scVI.jl

A Julia package for fitting VAEs to single-cell data using count distributions. Based on the Python implementation in the `scvi-tools` package.

Overview

The scVI model was first proposed in Lopez R, Regier J, Cole MB *et al.* Deep generative modeling for single-cell transcriptomics. *Nat Methods* **15**, 1053-1058 (2018).

More on the much more extensive Python package ecosystem `scvi-tools` can be found on the website and in the corresponding paper Gayoso A, Lopez R, Xing G. *et al.* A Python library for probabilistic analysis of single-cell omics data. *Nat Biotechnol* **40**, 163–166 (2022).

https://github.com/maren-ha/scVI.jl

UNIVERSITÄTS KLINIKUM FREIBURG