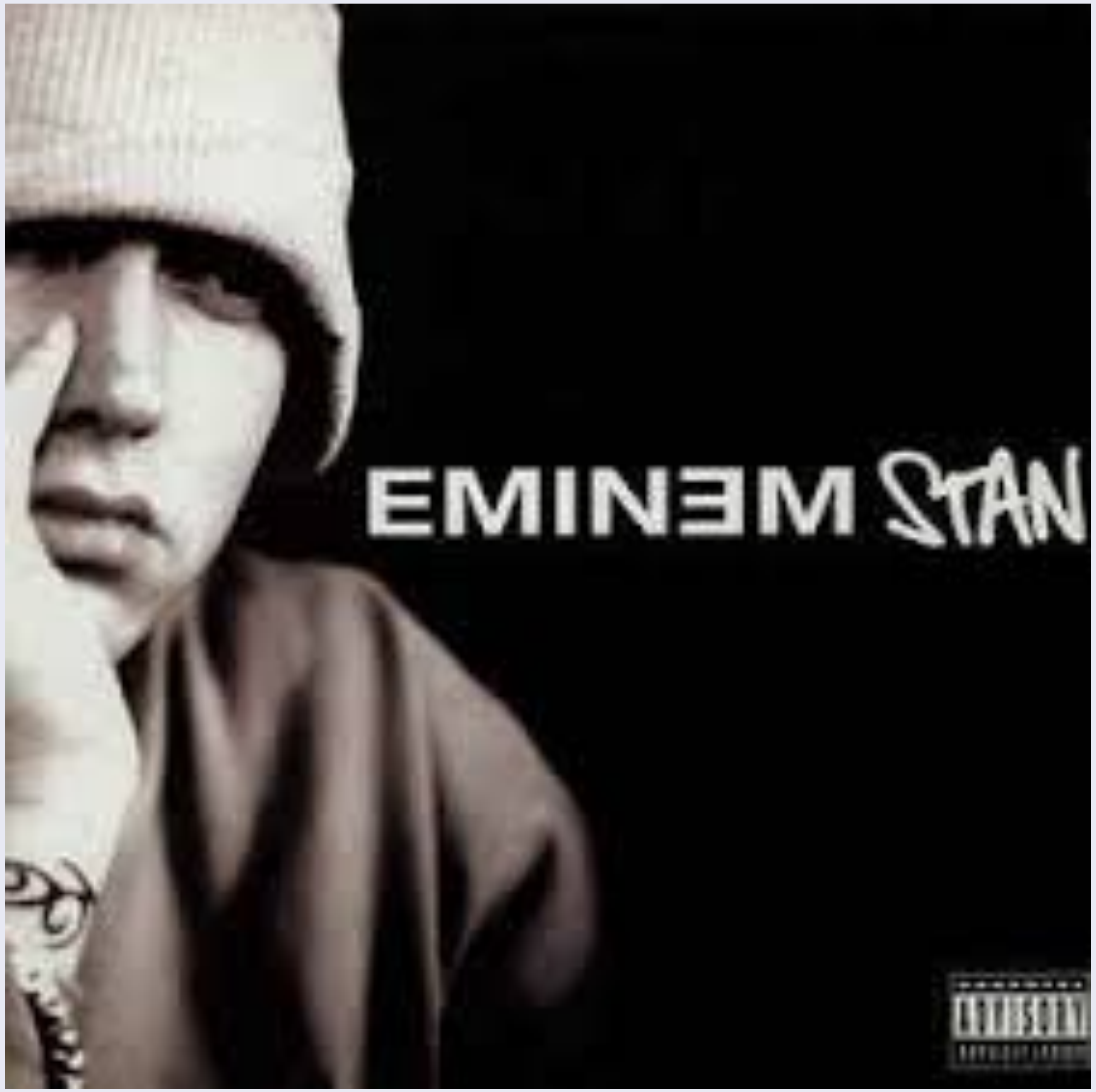
# Dear Stan, I meant to write you sooner but I just been busy

The language is just the tip of the iceberg

PROBPROG. Oct 5, 2018.

Daniel Lee
Stan Development Team / Generable


EMINƎM
STAN
PARENTAL ADVISORY EXPLICIT CONTENT

## Who's heard of Stan?

## Who's written a Stan program?

Today Stan is…

## Today Stan is…
focused on applied research

- **Biological sciences:** clinical drug trials, entomology, ophthalmology, neurology, genomics, agriculture, botany, fisheries, cancer biology, epidemiology, population ecology, neurology

- **Physical sciences:** astrophysics, molecular biology, oceanography, climatology

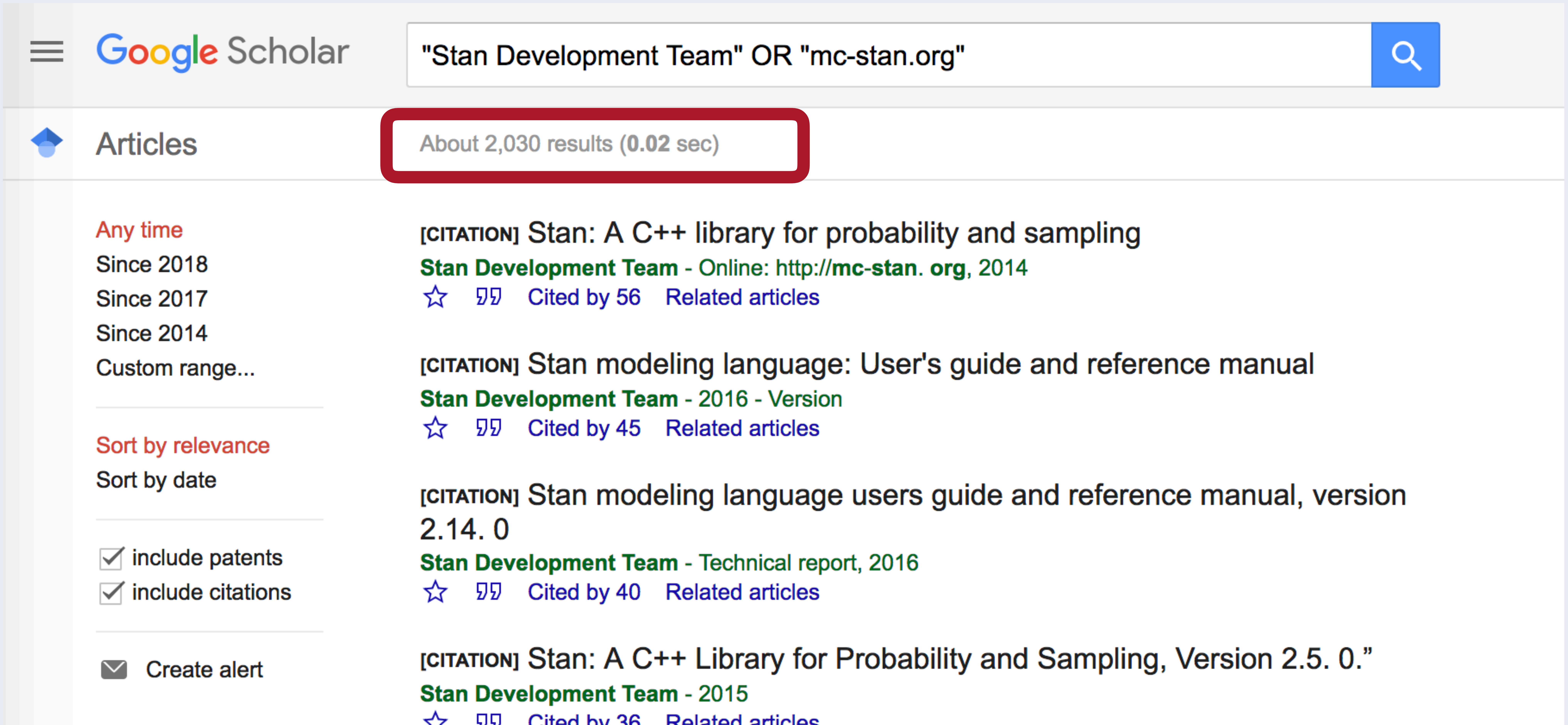
- **Social sciences:** population dynamics, psycholinguistics, social networks, political science

- **Other:** materials engineering, finance, actuarial, sports, public health, recommender systems, educational testing

- **Generable:** pharma. mechanistic models of drugs used in clinical trials

# Today Stan is...

- Cited in ~2000 articles



Google Scholar

"Stan Development Team" OR "mc-stan.org"

Articles

About 2,030 results (**0.02** sec)

Any time
Since 2018
Since 2017
Since 2014
Custom range...

Sort by relevance
Sort by date

✓ include patents
✓ include citations

Create alert

[CITATION] Stan: A C++ library for probability and sampling
**Stan Development Team** - Online: http://**mc-stan**. **org**, 2014
Cited by 56 Related articles

[CITATION] Stan modeling language: User's guide and reference manual
**Stan Development Team** - 2016 - Version
Cited by 45 Related articles

[CITATION] Stan modeling language users guide and reference manual, version 2.14. 0
**Stan Development Team** - Technical report, 2016
Cited by 40 Related articles

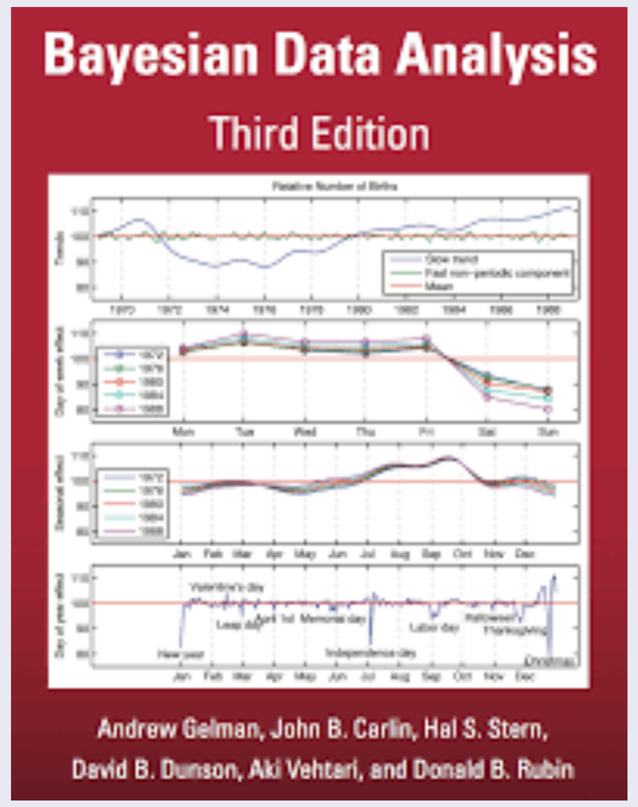
[CITATION] Stan: A C++ Library for Probability and Sampling, Version 2.5. 0."
**Stan Development Team** - 2015
Cited by 36 Related articles

# Today Stan is...

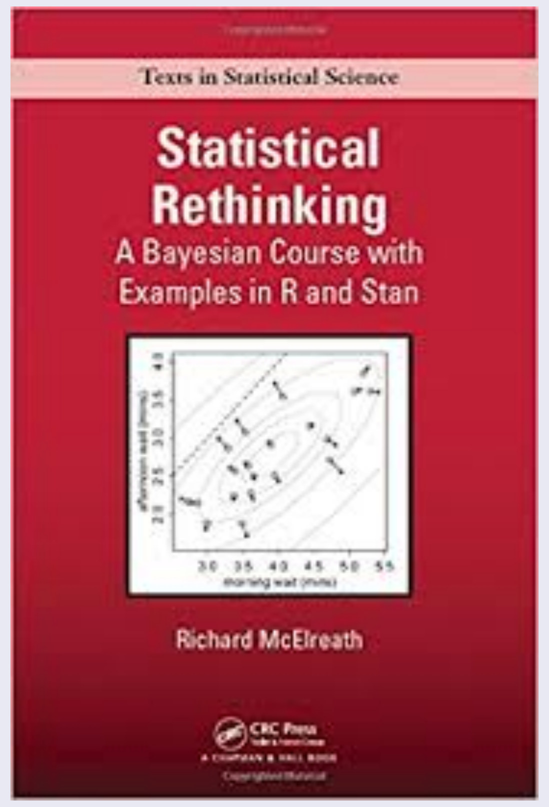
- 6 years old! (+1)
  v1.0:        August 30, 2012.

- 33 versions old
  v2.18.0:   July 13, 2018.
  Latest release includes within-chain parallelization (MPI and threading)
  GPU coming soon (it's already on a branch)

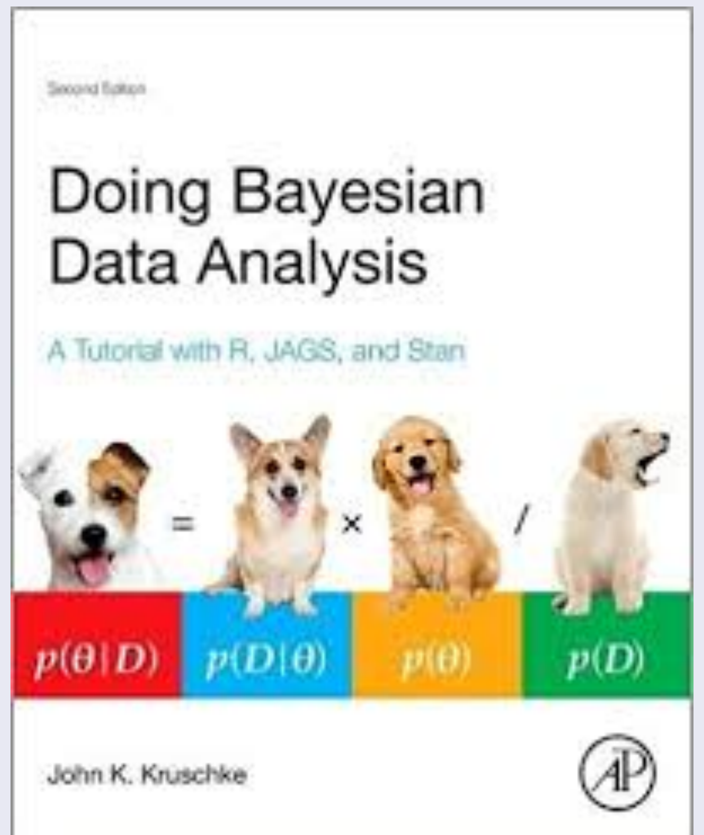
- In textbooks:



Bayesian Data Analysis
Third Edition
Andrew Gelman, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, and Donald B. Rubin



Texts in Statistical Science
Statistical Rethinking
A Bayesian Course with Examples in R and Stan
Richard McElreath



Second Edition
Doing Bayesian Data Analysis
A Tutorial with R, JAGS, and Stan
$p(\theta|D)$ = $p(D|\theta)$ × $p(\theta)$ / $p(D)$
John K. Kruschke

# Today Stan is…

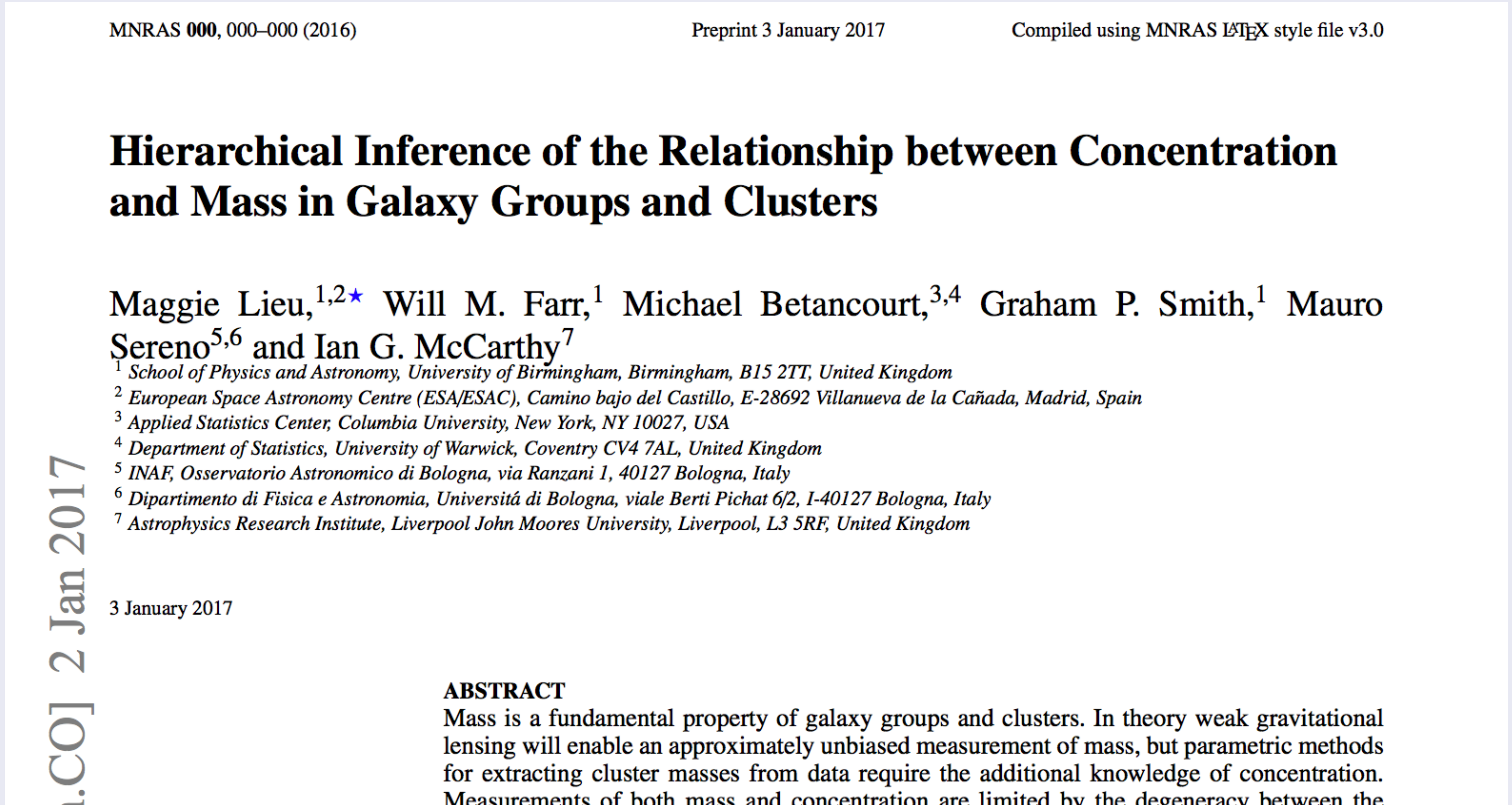
mentioned on TV: Billions S3E9.

## Today Stan is...

- used to implement some of the hardest statistical models

# Today Stan is…

- used to implement some of the hardest statistical models
  - detecting gravitational waves (LIGO)



MNRAS **000**, 000–000 (2016) Preprint 3 January 2017 Compiled using MNRAS LaTeX style file v3.0

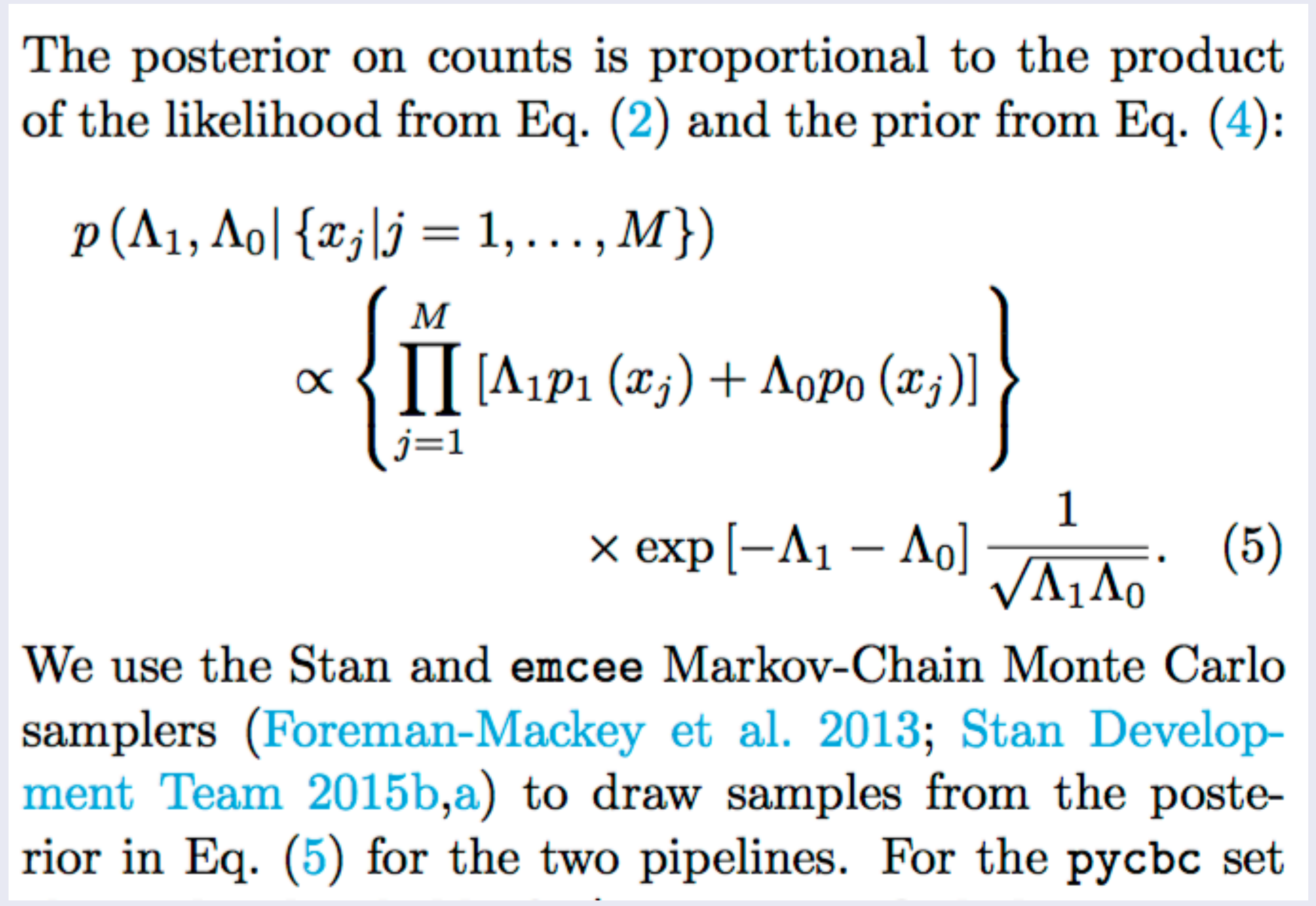
**Hierarchical Inference of the Relationship between Concentration and Mass in Galaxy Groups and Clusters**

Maggie Lieu,[1,2]★ Will M. Farr,[1] Michael Betancourt,[3,4] Graham P. Smith,[1] Mauro Sereno[5,6] and Ian G. McCarthy[7]

[1] *School of Physics and Astronomy, University of Birmingham, Birmingham, B15 2TT, United Kingdom*
[2] *European Space Astronomy Centre (ESA/ESAC), Camino bajo del Castillo, E-28692 Villanueva de la Cañada, Madrid, Spain*
[3] *Applied Statistics Center, Columbia University, New York, NY 10027, USA*
[4] *Department of Statistics, University of Warwick, Coventry CV4 7AL, United Kingdom*
[5] *INAF, Osservatorio Astronomico di Bologna, via Ranzani 1, 40127 Bologna, Italy*
[6] *Dipartimento di Fisica e Astronomia, Universitá di Bologna, viale Berti Pichat 6/2, I-40127 Bologna, Italy*
[7] *Astrophysics Research Institute, Liverpool John Moores University, Liverpool, L3 5RF, United Kingdom*

3 January 2017

**ABSTRACT**
Mass is a fundamental property of galaxy groups and clusters. In theory weak gravitational lensing will enable an approximately unbiased measurement of mass, but parametric methods for extracting cluster masses from data require the additional knowledge of concentration. Measurements of both mass and concentration are limited by the degeneracy between the



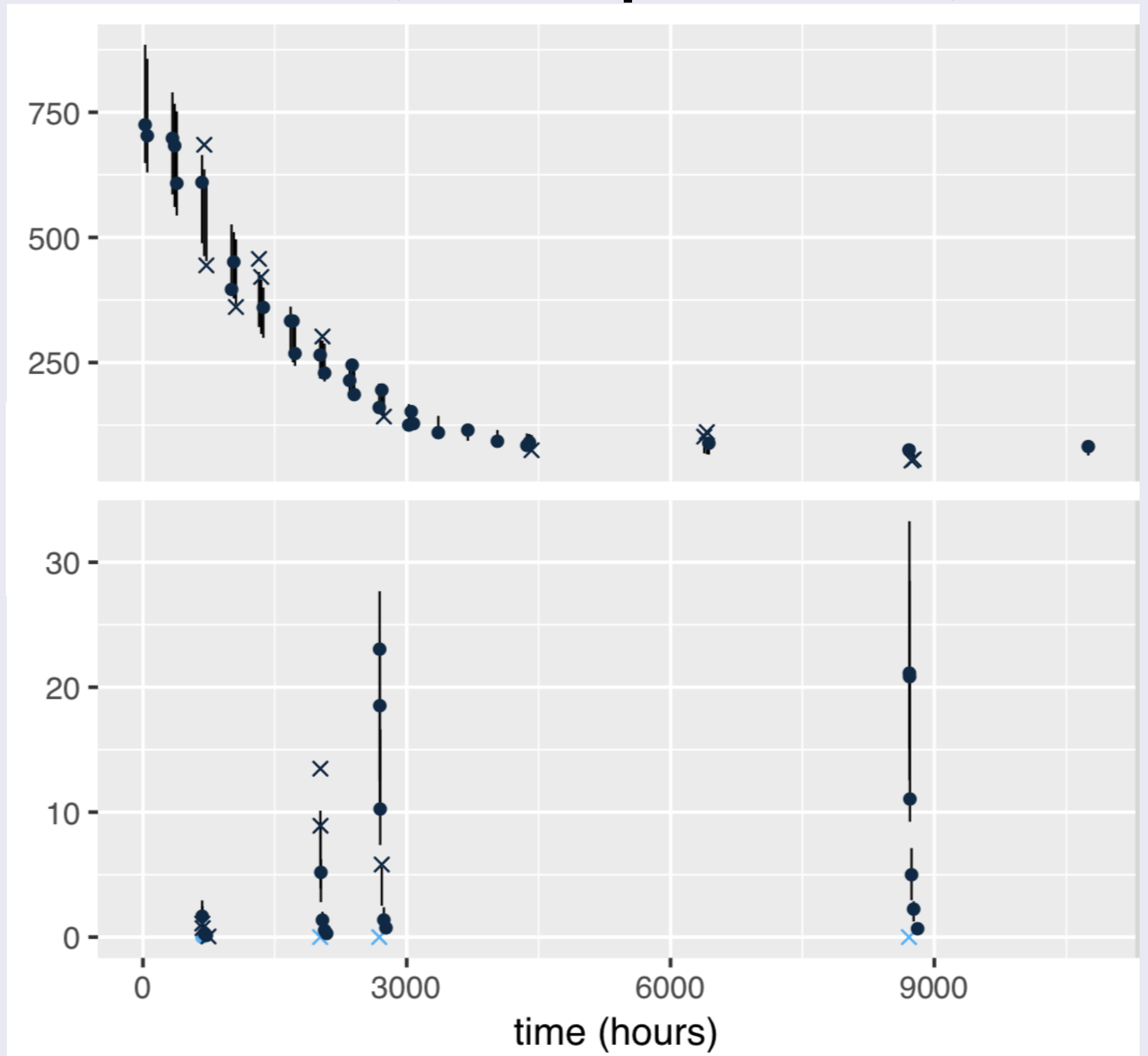
The posterior on counts is proportional to the product of the likelihood from Eq. (2) and the prior from Eq. (4):

$$p\left(\Lambda_1, \Lambda_0 \middle| \left\{x_j | j = 1, \ldots, M\right\}\right) \propto \left\{ \prod_{j=1}^{M} \left[\Lambda_1 p_1\left(x_j\right) + \Lambda_0 p_0\left(x_j\right)\right] \right\} \times \exp\left[-\Lambda_1 - \Lambda_0\right] \frac{1}{\sqrt{\Lambda_1 \Lambda_0}}. \quad (5)$$

We use the Stan and `emcee` Markov-Chain Monte Carlo samplers (Foreman-Mackey et al. 2013; Stan Development Team 2015b,a) to draw samples from the posterior in Eq. (5) for the two pipelines. For the **`pycbc`** set

# Today Stan is…

- used to implement some of the hardest statistical models
    - pharma: mechanistic / semi-mechanistic models
      ODEs, censored data, non-regular observation times, multiple patients, small data (< 30 patients), lots of parameters (~200), full Bayesian inference
      ,



Posterior predictions.
80% credible intervals.

Top: PD. Bottom: PK.

## What is Stan?

## Language for Statistical Models

- **Goal:** specify statistical models

$$\underset{\text{data}}{\underset{\nwarrow}{x}}$$

# Language for Statistical Models

- **Goal:** specify statistical models

$$\underset{\text{parameters}}{\theta}, \underset{\text{data}}{x}$$

## Language for Statistical Models

- **Goal:** specify statistical models

$$\underset{\text{model}}{p}(\underset{\text{parameters}}{\theta}, \underset{\text{data}}{x})$$

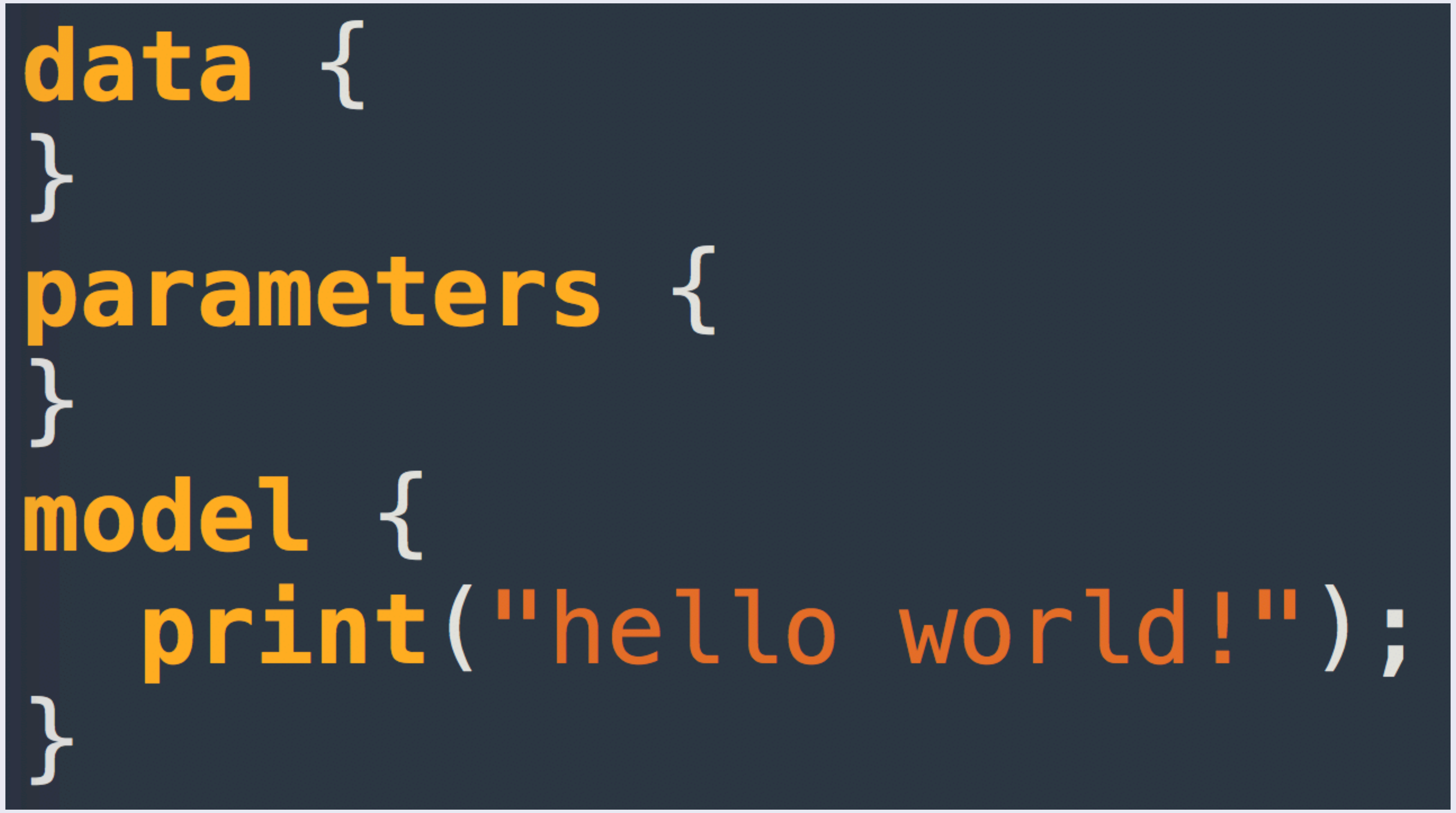
# Language for Statistical Models

- **Goal:** specify statistical models

$$\underset{\text{model}}{p}(\underset{\text{parameters}}{\theta}, \underset{\text{data}}{x})$$

- Stan is a language
    - statically typed, imperative
    - users define programs: data, parameters, **log joint pdf**
- User can specify any *differentiable* joint probability distribution function over data and parameters

## Example: Hello World



```
data {
}
parameters {
}
model {
  print("hello world!");
}
```

# Example: Logistic Regression

```
data {
  int<lower=0> N;
  vector[N] x;
  int<lower=0,upper=1> y[N];
}
parameters {
  real alpha;
  real beta;
}
model {
  y ~ bernoulli_logit(alpha + beta * x);
}
```

## Users define the statistical model

$$p(\theta, x)$$

## Inference algorithms use $p(\theta, x)$

- Bayesian inference; Markov Chain Monte Carlo (MCMC)
- Approximate Bayesian inference
- Optimization

# Inference algorithms use $p(\theta, x)$

- Bayesian inference; Markov Chain Monte Carlo (MCMC)
  - $p(\theta \,|\, x)$ approximated with $\{\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(N)}\}$
- Approximate Bayesian inference
  - ex: $\hat{p}(\theta \,|\, x) \approx q(\hat{\phi})$ where $\hat{\phi} = \underset{\phi}{\operatorname{argmin}} \; D_{\mathrm{KL}}\left(q(\theta \,|\, \phi) \,||\, p(\theta, x)\right)$
- Optimization
  - $\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \; p(\theta, x)$ (only holds when there's a single optima)

## Interfaces

- CmdStan, RStan, PyStan
- C++ API
- C++ automatic differentiation library

- RStanArm, brms, prophet, …

## Stan: mc-stan.org

- Language
- Inference algorithms
- Interfaces

- Open-source github.com/stan-dev
  core: BSD
  interfaces: GPL or BSD

## What does Stan do well?

# Language is decoupled from inference

- Stan language defines the statistical model
  Inference is independent of the statistical model

# Language is decoupled from inference

- Stan language defines the statistical model
  Inference is independent of the statistical model

- **This is by design.**

# Language is decoupled from inference

- Stan language defines the statistical model
  Inference is independent of the statistical model

- **This is by design.**

- Users focus on writing statistical models: $p(\theta, x)$

# Language is decoupled from inference

- Stan language defines the statistical model
  Inference is independent of the statistical model

- **This is by design.**

- Users focus on writing statistical models: $$p(\theta, x)$$

- Users choose what inference they want:

Bayesian
$$p(\theta \,|\, x)$$

approximate Bayesian
$$\hat{p}(\theta \,|\, x) \approx q(\hat{\phi})$$

optimization
$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \; p(\theta, x)$$

# Inference algorithms have access to

- Log probability distribution function in multiple flavors

1. Log probability function
2. Log probability function up to an additive constant
3. Log probability function with Jacobian adjustment
4. Log probability function with Jacobian adjustment up to an additive constant

# Inference algorithms have access to

- Log probability distribution function in multiple flavors

1. Log probability function
2. Log probability function up to an additive constant
3. Log probability function with Jacobian adjustment
4. **Log probability function with Jacobian adjustment up to an additive constant**

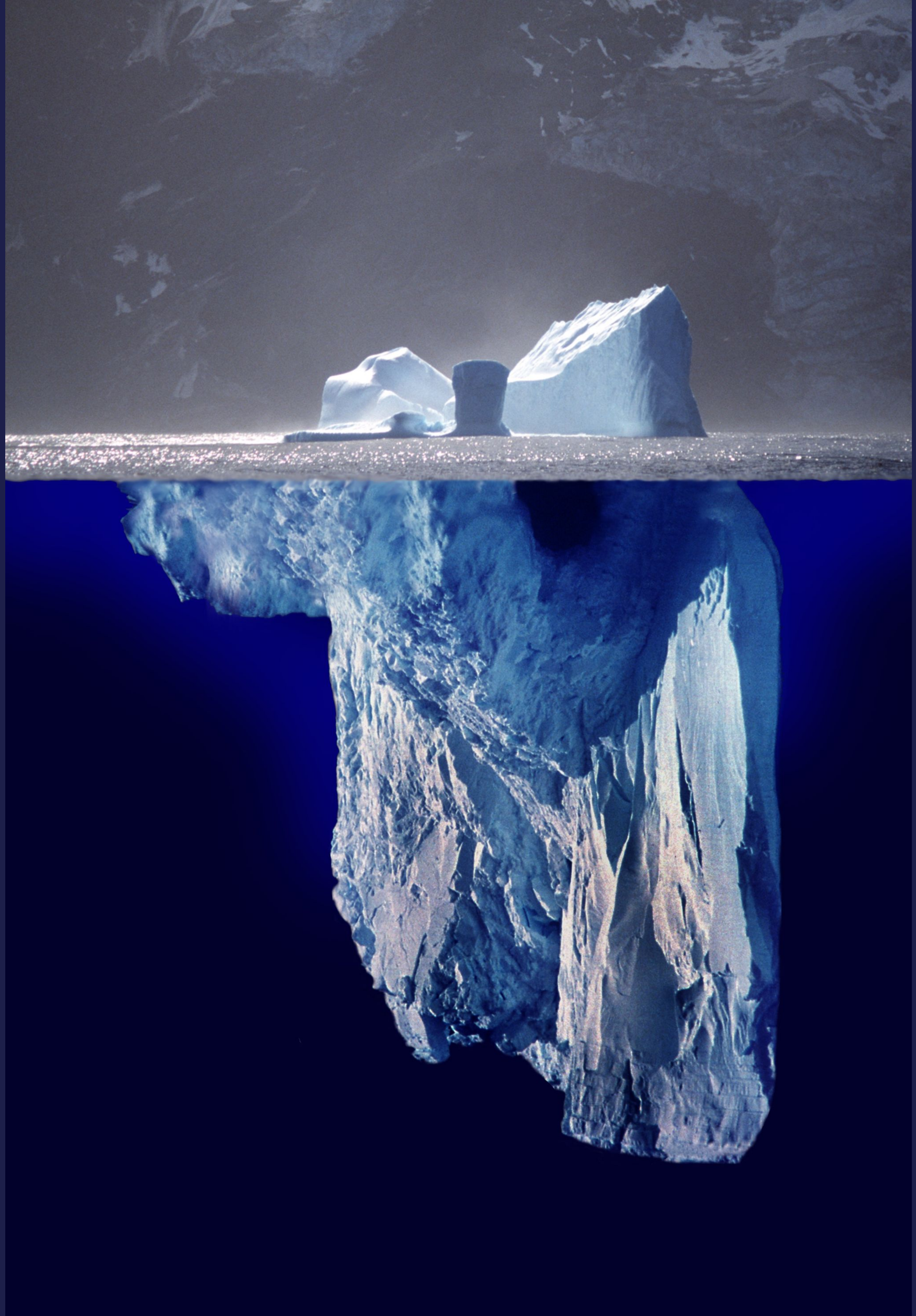
# Inference algorithms have access to

- Log probability distribution function in multiple flavors
    - gradients with respect to the parameters of the model

# Inference algorithms have access to

- Log probability distribution function in multiple flavors
    - gradients with respect to the parameters of the model

    - for **any model** written in the Stan language,
      inference algorithms have access to the log probability functions
      and the gradients with respect to the parameters



## The language is just the tip of the iceberg

# What you might not realize

- The language is just the tip of the iceberg
  Our users interact with the language.

## What you might not realize

- The language is just the tip of the iceberg
  Our users interact with the language.

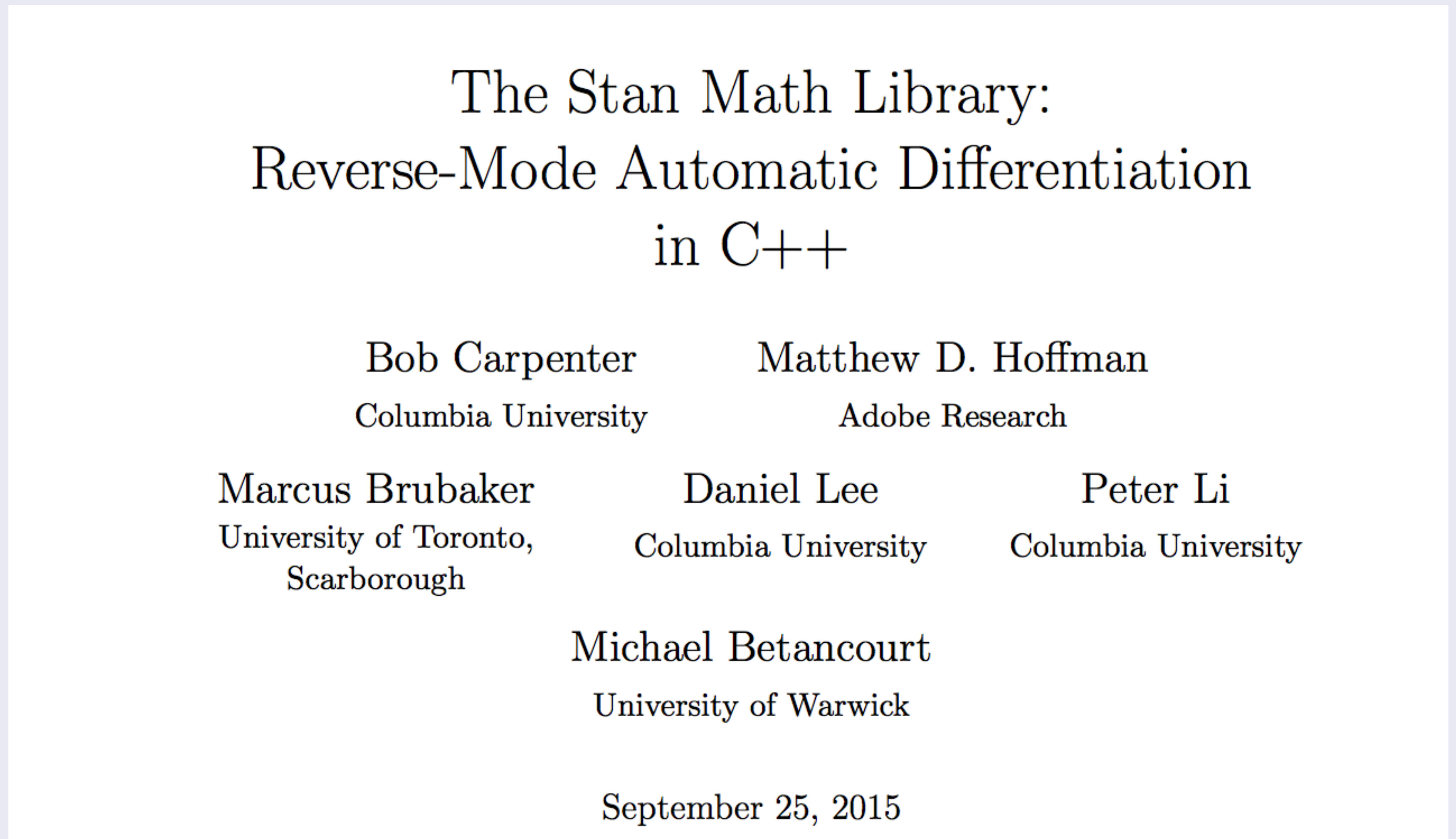
- Stan language is translated to C++

- Most of the work is done in C++
  Full C++ API for inference.
  C++ autodiff library

https://github.com/stan-dev/stan

https://github.com/stan-dev/math

https://arxiv.org/abs/1509.07164



The Stan Math Library:
Reverse-Mode Automatic Differentiation
in C++

Bob Carpenter, Columbia University
Matthew D. Hoffman, Adobe Research
Marcus Brubaker, University of Toronto, Scarborough
Daniel Lee, Columbia University
Peter Li, Columbia University
Michael Betancourt, University of Warwick

September 25, 2015

## Still awake?

I've got questions for you!

## Q1:
## Can Stan be parallelized?

## Yes.

Stan has MPI and threading available now.

GPU is in the works.

Users have hacked implementations in C++.

## Q2:
## Can Stan deal with discrete parameters?

## No, but…

**In the C++ you could.**

**It's straightforward to marginalize.**

**It's easy to generate discrete data.**

## Q3:
## Does Stan produce a graphical model?

# No.

## Graphical models are a subset of what's expressible in Stan.

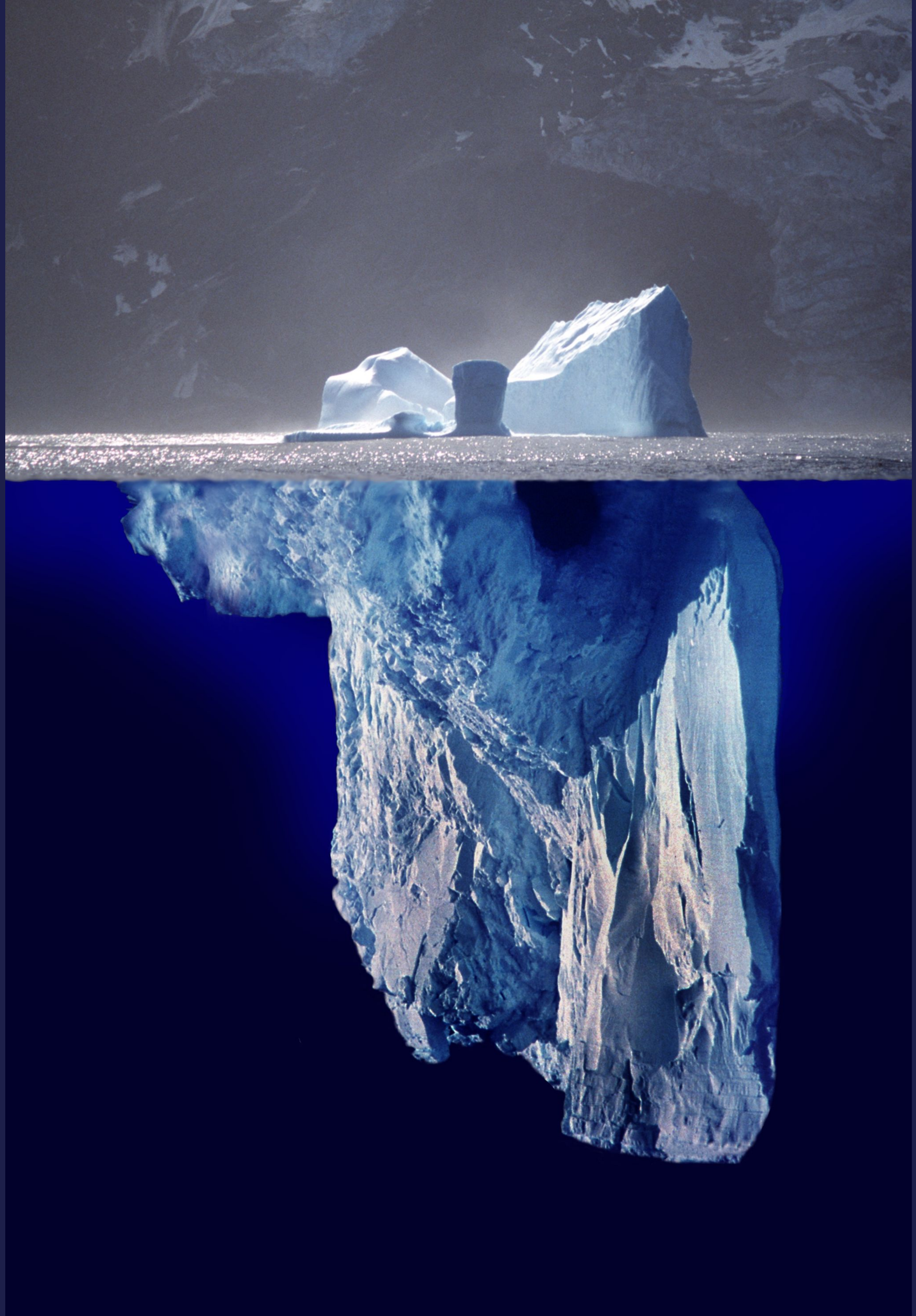
## The Stan language is Turing complete.

**(imperative, conditional branching, arbitrary amount of memory)**

## Q4:
## Can Stan do online learning?
**(using mini-batch, read from DB, etc)**

## Yes.

### By using the C++ API.



## The language is just the tip of the iceberg

# Final thoughts

- Stan was original designed as a C++-only library (late 2010).
- Algorithm development? Stan is for you:
  - Work with the C++ APIs
  - Lots of users
  - Implementations are tested on lots of real-world problems
- "Dear Stan, I mean to write you sooner but I just been busy"

  None of this is new.
  It's been available for 6+ years. Come join us!
  github.com/stan-dev

## Thank you!

daniel@generable.com @djsyclik mc-stan.org