A Brief History of Stan

Daniel Lee

Stan Developer

bearlee@alum.mit.edu

Outline

1. Today Stan is …
2. The Stan ecosystem
3. How did we get here?
4. Fun bits of history

- Disclosure: this talk is core Stan centric
- Disclosure: this talk is all fluff

Today Stan is ...

Today Stan is …

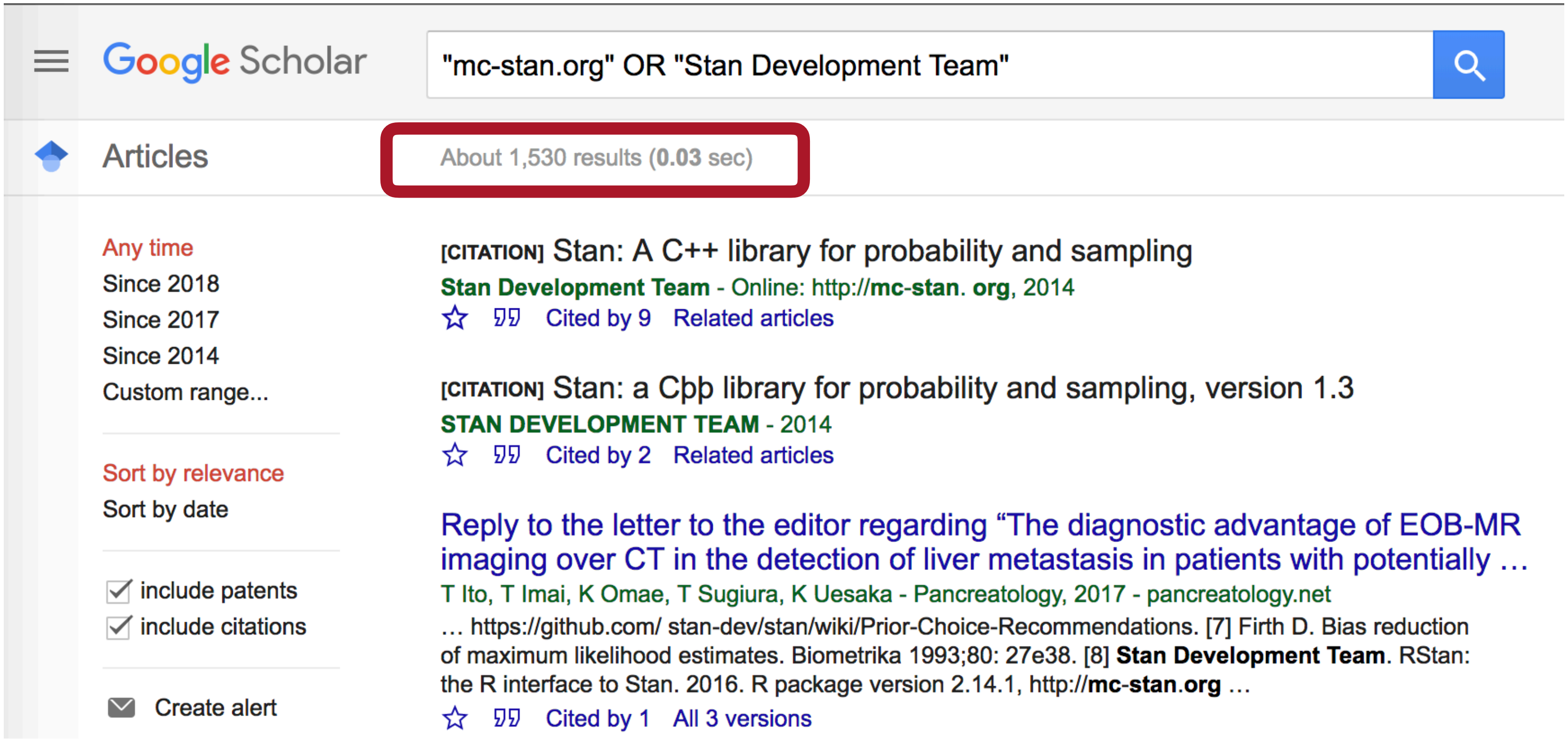
- 1595 days old
  - 4 years, 4 months, 14 days since v1.0.

Today Stan is …

- 32 versions old!
  - First version: v1.0. August 30, 2012.
  - Latest version. v2.17.1. December 11, 2017.

Today Stan is ...

- Cited in ~1500 papers



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

About 1,530 results (**0.03** sec)

[CITATION] Stan: A C++ library for probability and sampling
Stan Development Team - Online: http://**mc-stan. org**, 2014
☆ 99 Cited by 9 Related articles

[CITATION] Stan: a Cþþ library for probability and sampling, version 1.3
STAN DEVELOPMENT TEAM - 2014
☆ 99 Cited by 2 Related articles

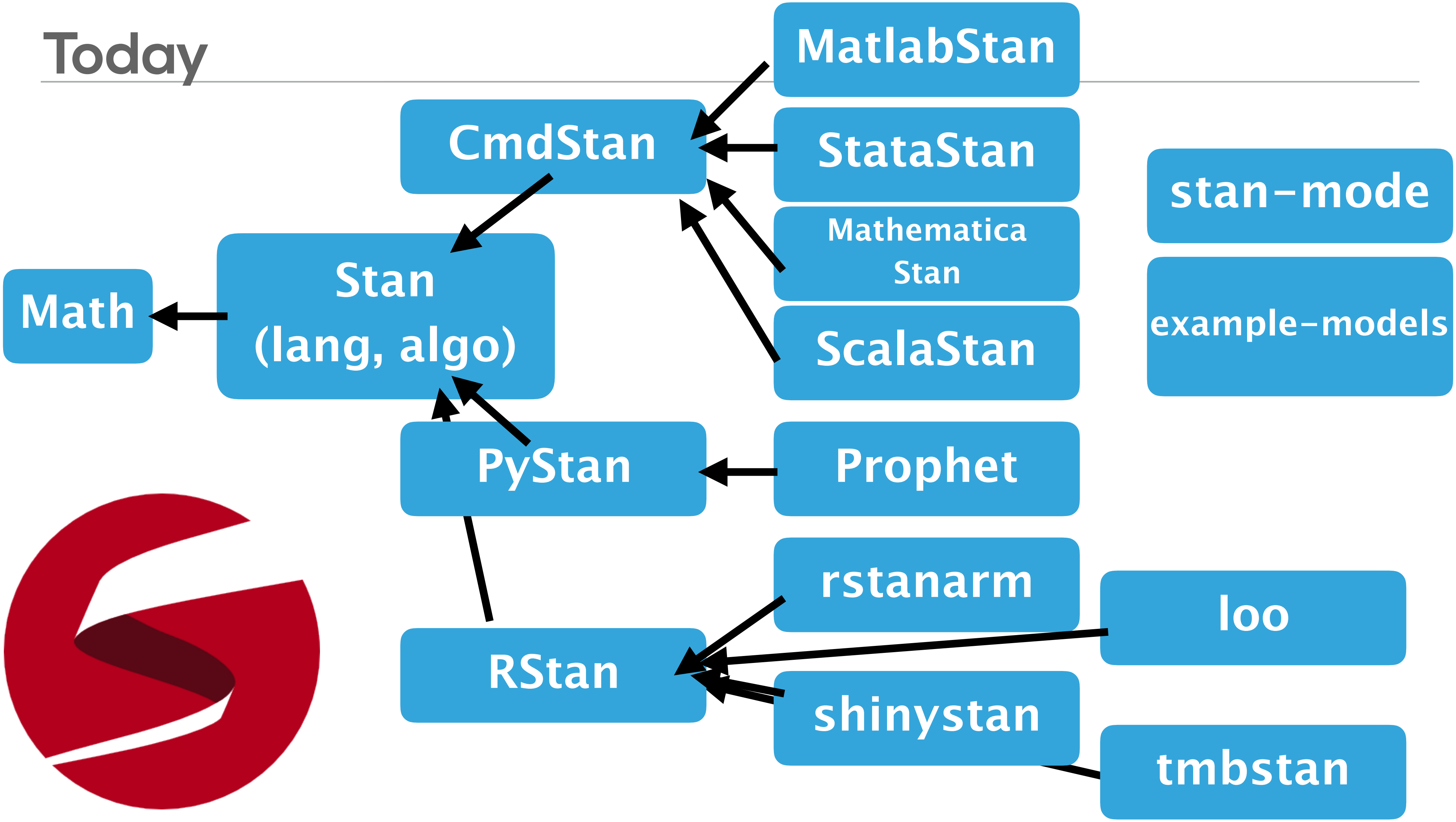
Reply to the letter to the editor regarding “The diagnostic advantage of EOB-MR imaging over CT in the detection of liver metastasis in patients with potentially …
T Ito, T Imai, K Omae, T Sugiura, K Uesaka - Pancreatology, 2017 - pancreatology.net
… https://github.com/ stan-dev/stan/wiki/Prior-Choice-Recommendations. [7] Firth D. Bias reduction of maximum likelihood estimates. Biometrika 1993;80: 27e38. [8] **Stan Development Team**. RStan: the R interface to Stan. 2016. R package version 2.14.1, http://**mc-stan.org** …
☆ 99 Cited by 1 All 3 versions

Today Stan is …

- in textbooks
  - Bayesian Data Analysis, Third Edition
  - Statistical Rethinking
  - Doing Bayesian Analysis
- used in academic research
- used in industry

The Stan ecosystem

Today



MatlabStan

CmdStan

StataStan

Mathematica Stan

stan−mode

Math

Stan (lang, algo)

ScalaStan

example−models

PyStan

Prophet

rstanarm

loo

RStan

shinystan

tmbstan

August 30, 2012. v1.0

Stan

STAN

August 30, 2012. v1.0

Stan
(math, stan, lang, algo,
cmdstan, rstan)

How did we get here?

What was around before Stan?

- Languages for specifying models
  - BUGS, JAGS
  - Infer.net
  - lme4, bglmer
  - C++ or other lower level language
- Hamiltonian Monte Carlo
  - Algorithm described in textbooks
  - hand-coded implementations
- Automatic differentiation / symbolic differentiation
  - Theano
  - CppAD, Sacado, Boost, …

How did we get here?

IMO:

1. Technical progress

   language + auto-diff + transforms + NUTS

2. Focus on solving a problem
3. Community
   - User community
   - Developer community

Technical progress (v1.0)

- Stan Language
  - Turing complete
  - Everything in the language could be autodiffed
- Auto diff package
  - Flexible; fast
- Algorithms
  - HMC is hard to tune: NUTS fixed that
- Transformations

Focus on solving a problem

- We *ate our own dog food*
  - http://andrewgelman.com/2012/08/30/a-stan-is-born/
  - http://andrewgelman.com/2012/08/30/stan-is-fast/

- We didn't release until
  - Stan was faster than JAGS
  - There was an R interface

Community

- stan-users (archived)
  https://groups.google.com/forum/?fromgroups#!forum/stan-users
- stan-dev (archived)
  https://groups.google.com/forum/?fromgroups#!forum/stan-dev
- discourse
  http://discourse.mc-stan.org

Community: developers

- We value:
  - Openness
    - See stan-dev (archived) and discourse.
  - Being helpful

Team (order of joining team)

- Andrew Gelman
- Bob Carpenter
- Daniel Lee
- Ben Goodrich
- Michael Betancourt
- Marcus Brubaker
- Jiqiang Guo
- Allen Riddell
- Marco Inacio
- Jeffrey Arnold
- Mitzi Morris
- Rob Trangucci
- Rob Goedman
- Brian Lau
- Jonah Sol Gabry
- Robert L. Grant
- Krzysztof Sakrejda
- Aki Vehtari
- Rayleigh Lei
- Sebastian Weber
- Charles Margossian
- Vincent Picaud
- Imad Ali
- Sean Talts
- Ben Bales
- Ari Hartikainen
- Matthijs Vákár
- Andrew Johnson
- Dan Simpson

First commits! (to Stan / Math)

- Bob Carpenter
  Apr 19, 2011
- Matt Hoffman
  May 5, 2011
- Daniel Lee
  Sept 19, 2011
- Ben Goodrich
  Jan 12, 2012
- Jiqiang Guo
  Feb 23, 2012
- Marcus Brubaker
  Mar 10, 2012
- Michael Betancourt
  Dec 23, 2012
- Peter Li
  Jan 23, 2013
- Marco Inacio
  Apr 27, 2013
- Jeffrey Arnold
  May 31, 2013
- Allen Riddell
  Apr 7, 2014
- Mitzi Morris
  Mar 19, 2014
- Rob Trangucci
  Oct 1, 2014
- Sebastian Weber
  Sep 27, 2015
- Krzysztof Sakrejda
  Mar 8, 2015
- Jonah Sol Gabry
  May 27, 2015
- Aki Vehtari
  Nov 4, 2016
- Rayleigh Lei
  Sep 2, 2016
- Charles Margossian
  Aug 18, 2016
- Sean Talts
  Dec 27, 2016
- Ben Bales
  Apr 14, 2017
- Matthijs Vákár
  Aug 23, 2017
- Andrew Johnson
  Sep 24, 2017

Team: here at StanCon

- Bob Carpenter
- Daniel Lee
- Ben Goodrich
- Mitzi Morris
- Jonah Sol Gabry
- Aki Vehtari
- Charles Margossian
- Imad Ali
- Sean Talts
- Ben Bales

What have we done since v1.0

- Language expansion
  - functions, user-defined functions
  - ordinary differential equation solver
- Speed
  - Vectorization
- Algorithms
  - Optimization, ADVI
  - New NUTS
- Usability
  - RStan, rstanarm, loo, …

- Communities
  - User
  - Developer

Fun bits of history

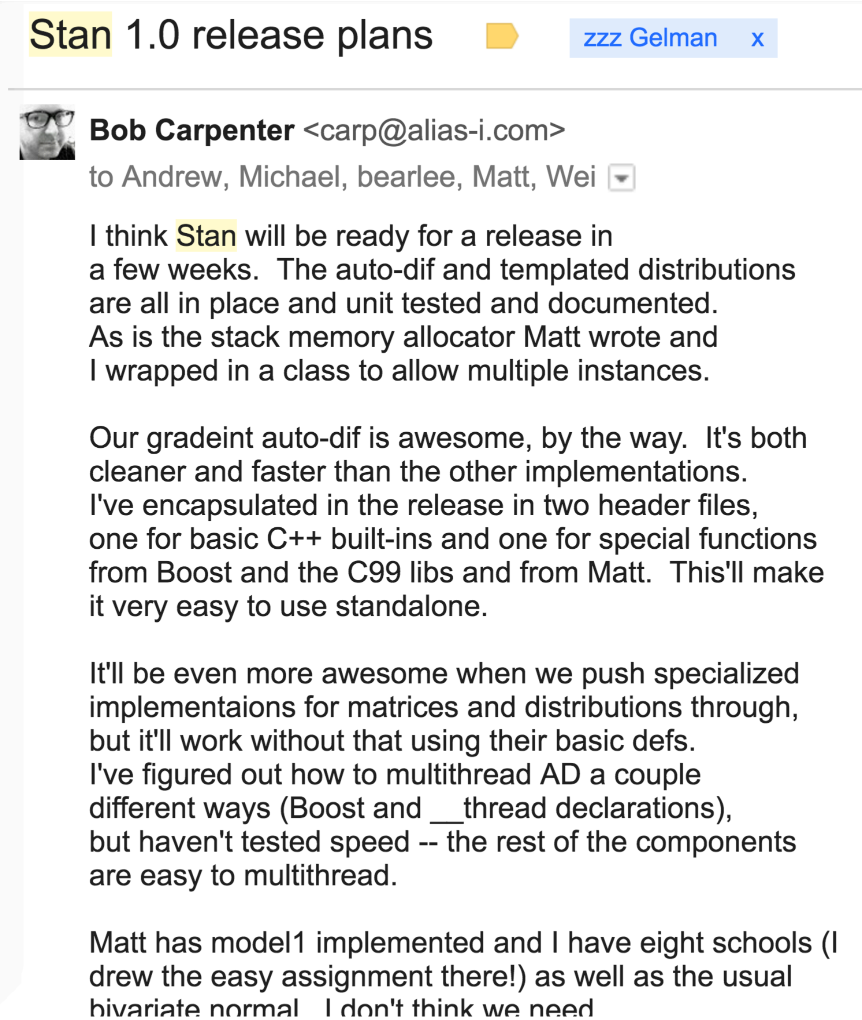
v1.0: August 30, 2012.

- We thought we were ready in early 2011.
- First commit! (in version control)

```
r183 | carp | 2011-04-19 15:50:52 -0400 (Tue, 19 Apr 2011) | 1 line

first commit with README
------------------------------------------------------------------------
```

4/28/2011... Release 1.0?



Stan 1.0 release plans

Bob Carpenter <carp@alias-i.com>
to Andrew, Michael, bearlee, Matt, Wei

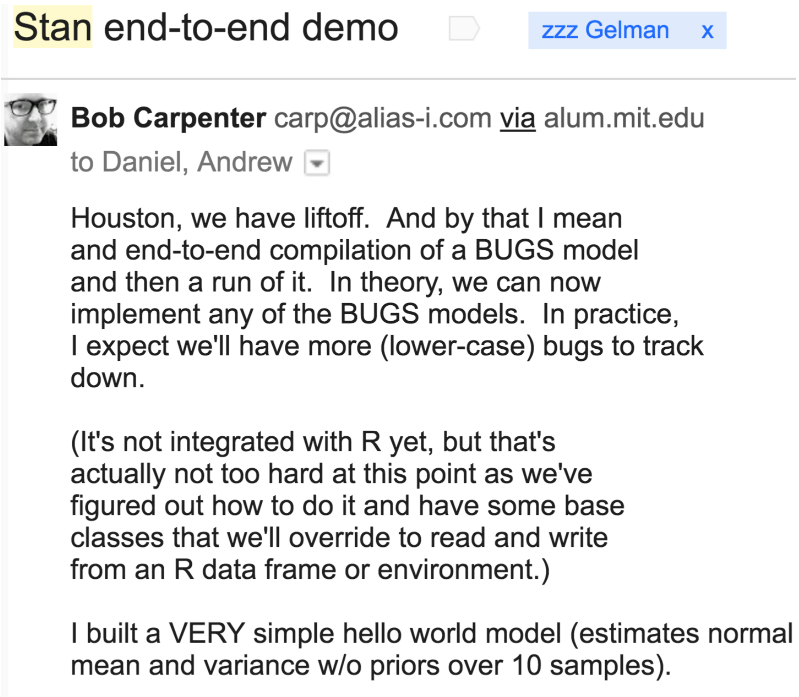
I think Stan will be ready for a release in
a few weeks. The auto-dif and templated distributions
are all in place and unit tested and documented.
As is the stack memory allocator Matt wrote and
I wrapped in a class to allow multiple instances.

Our gradeint auto-dif is awesome, by the way. It's both
cleaner and faster than the other implementations.
I've encapsulated in the release in two header files,
one for basic C++ built-ins and one for special functions
from Boost and the C99 libs and from Matt. This'll make
it very easy to use standalone.

It'll be even more awesome when we push specialized
implementaions for matrices and distributions through,
but it'll work without that using their basic defs.
I've figured out how to multithread AD a couple
different ways (Boost and __thread declarations),
but haven't tested speed -- the rest of the components
are easy to multithread.

Matt has model1 implemented and I have eight schools (I
drew the easy assignment there!) as well as the usual
bivariate normal. I don't think we need

10/10/2011... Release 1.0?



Stan end-to-end demo

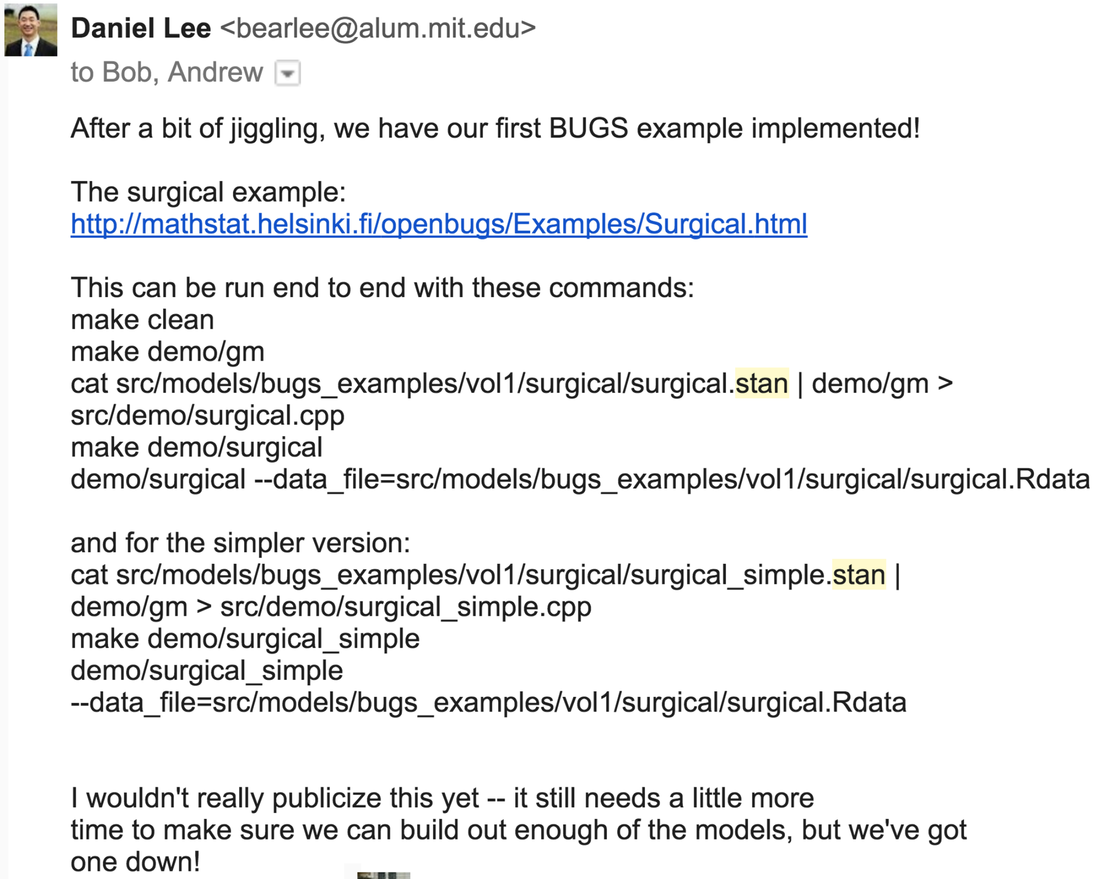
Bob Carpenter carp@alias-i.com via alum.mit.edu

to Daniel, Andrew

Houston, we have liftoff. And by that I mean and end-to-end compilation of a BUGS model and then a run of it. In theory, we can now implement any of the BUGS models. In practice, I expect we'll have more (lower-case) bugs to track down.

(It's not integrated with R yet, but that's actually not too hard at this point as we've figured out how to do it and have some base classes that we'll override to read and write from an R data frame or environment.)

I built a VERY simple hello world model (estimates normal mean and variance w/o priors over 10 samples).



Daniel Lee <bearlee@alum.mit.edu>

to Bob, Andrew

After a bit of jiggling, we have our first BUGS example implemented!

The surgical example:
http://mathstat.helsinki.fi/openbugs/Examples/Surgical.html

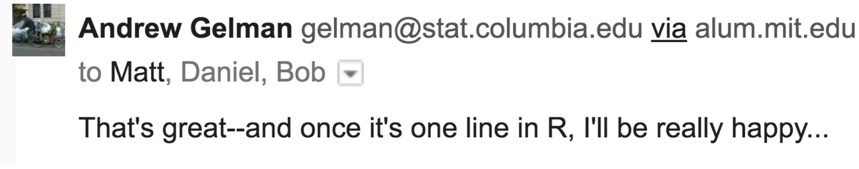
This can be run end to end with these commands:

```
make clean
make demo/gm
cat src/models/bugs_examples/vol1/surgical/surgical.stan | demo/gm >
src/demo/surgical.cpp
make demo/surgical
demo/surgical --data_file=src/models/bugs_examples/vol1/surgical/surgical.Rdata
```

and for the simpler version:

```
cat src/models/bugs_examples/vol1/surgical/surgical_simple.stan |
demo/gm > src/demo/surgical_simple.cpp
make demo/surgical_simple
demo/surgical_simple
--data_file=src/models/bugs_examples/vol1/surgical/surgical.Rdata
```

I wouldn't really publicize this yet -- it still needs a little more time to make sure we can build out enough of the models, but we've got one down!



Andrew Gelman gelman@stat.columbia.edu via alum.mit.edu

to Matt, Daniel, Bob

That's great--and once it's one line in R, I'll be really happy...

Manual… then and now

- Thanks, Bob!
  - Now: 620 pages
  - v1.0: 178 pages

Want to see the first Stan demo?

First Stan demo: bivariate normal

```
emacs bivar_norm.cpp

./bivar_norm
```

Second Stan demo: eight schools!

```
emacs eight_schools.cpp
```

Want to see the first Stan program?

BUGS: Rats. Oct 11, 2011.

```
# http://www.mrc-bsu.cam.ac.uk/bugs/
winbugs/Vol1.pdf
# Page 3: Rats
data {
    int N;
    int T;
    double x[T];
    double xbar;
}
parameters {
    double mu[N,T];
    double(0,) tau_c;
    double alpha[N];
    double beta[N];
    double alpha_c;
    double alpha_tau;
    double beta_c;
    double beta_tau;
    double(0,) Y[N,T];
    double sigma;
    double alpha0;
}
```

```
model {
  for (i in 1:N) {
    for (j in 1:T) {
      Y[i, j] ~ normal(mu[i , j], tau_c);
      mu[i, j] <- alpha[i] + beta[i] * (x[j] - xbar);
    }
    alpha[i] ~ normal(alpha_c, alpha_tau);
    beta[i] ~ normal(beta_c, beta_tau);
   }
   tau_c ~ gamma(0.001,0.001);
   sigma <- 1 / sqrt(tau_c);
   alpha_c ~ normal(0.0,1.0E-6);
   alpha_tau ~ gamma(0.001,0.001);
   beta_c ~ normal(0.0,1.0E-6);
   beta_tau ~ gamma(0.001,0.001);
   alpha0 <- alpha_c - xbar * beta_c;
}
```

BUGS: Rats. v1.0

```
# http://www.mrc-bsu.cam.ac.uk/bugs/winbugs/Vol1.pdf
# Page 3: Rats
data {
  int<lower=0> N;
  int<lower=0> T;
  real x[T];
  real y[N,T];
  real xbar;
}
parameters {
  real alpha[N];
  real beta[N];

  real mu_alpha;
  real mu_beta;          // beta.c in original bugs model

  real<lower=0> sigmasq_y;
  real<lower=0> sigmasq_alpha;
  real<lower=0> sigmasq_beta;
}
transformed parameters {
  real<lower=0> sigma_y;       // sigma in original bugs
model
  real<lower=0> sigma_alpha;
  real<lower=0> sigma_beta;

  sigma_y <- sqrt(sigmasq_y);
  sigma_alpha <- sqrt(sigmasq_alpha);
  sigma_beta <- sqrt(sigmasq_beta);
}
```

```
model {
  mu_alpha ~ normal(0, 100);
  mu_beta ~ normal(0, 100);
  sigmasq_y ~ inv_gamma(0.001, 0.001);
  sigmasq_alpha ~ inv_gamma(0.001, 0.001);
  sigmasq_beta ~ inv_gamma(0.001, 0.001);
  alpha ~ normal(mu_alpha, sigma_alpha); // vectorized
  beta ~ normal(mu_beta, sigma_beta);  // vectorized
  for (n in 1:N)
    for (t in 1:T)
      y[n,t] ~ normal(alpha[n] + beta[n] * (x[t] - xbar),
                      sigma_y);

}
generated quantities {
  real alpha0;
  alpha0 <- mu_alpha - xbar * mu_beta;
}
```

Take aways

Stan is rad!

- Development is a team effort

- Community
  - online: http://discourse.mc-stan.org

Thanks!