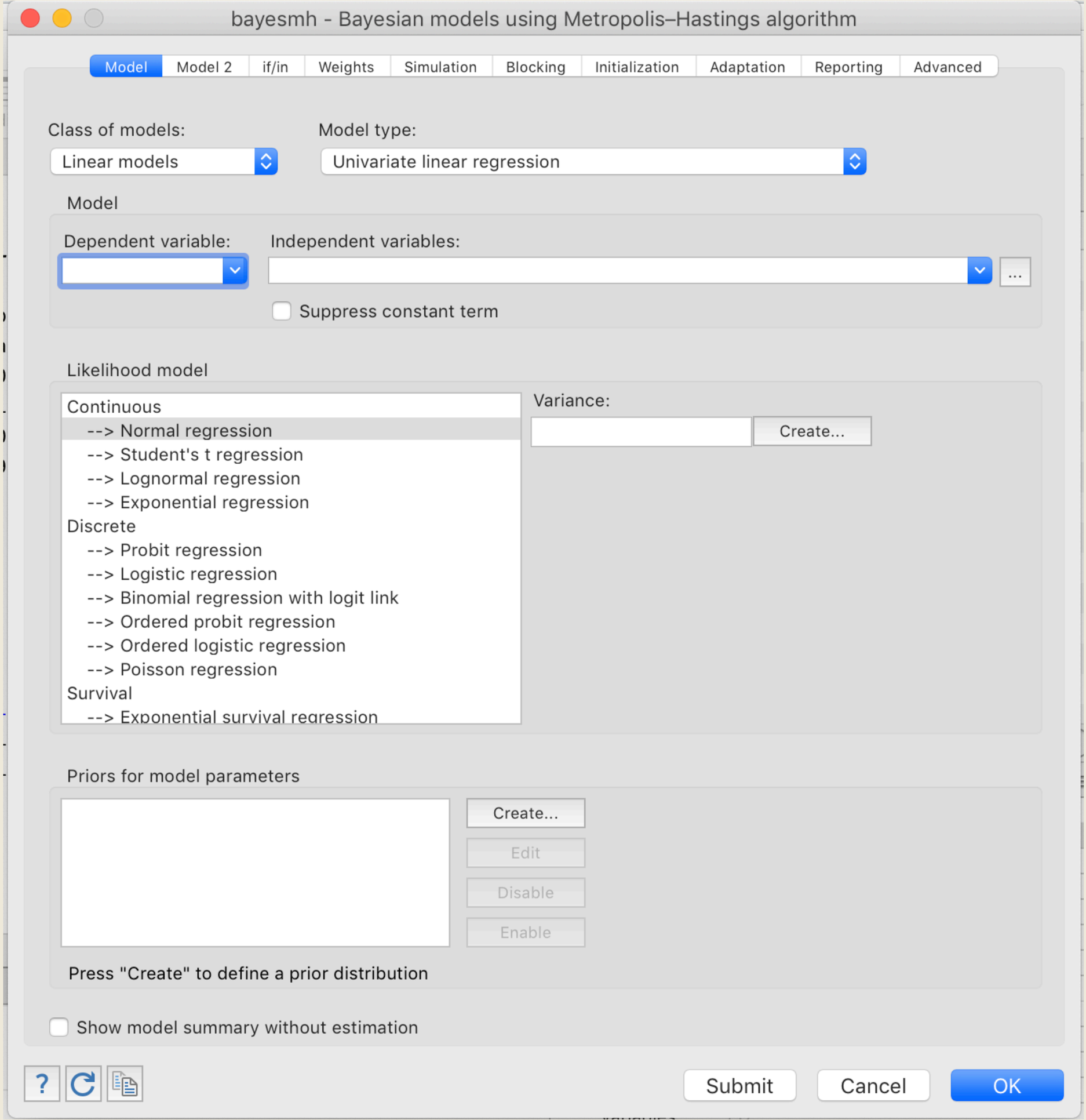

BayesCamp

# A bird's-eye view of Bayesian software in 2021

## Opportunities for Stata?

Robert Grant
robert@bayescamp.com

# bayesmh (Stata 17)



# bayesmh (Stata 17)

```
sysuse auto

gen price1000=price/1000

bayesmh foreign price1000 mpg, ///
  likelihood(logit) //
  prior({foreign: _cons}, normal(0, 10)) ///
  prior({foreign: price1000}, normal(0, 1)) ///
  prior({foreign: mpg}, normal(-0.2, 1))
```

1. Interface
2. Programming
3. Quick start
4. Specific applications
5. Sampler algorithms
6. Interoperability
7. Guidance, community, support
8. Crypto-Bayesians

# Not one of those “best software” talks


montage by Darren Dahly, who never gets any credit

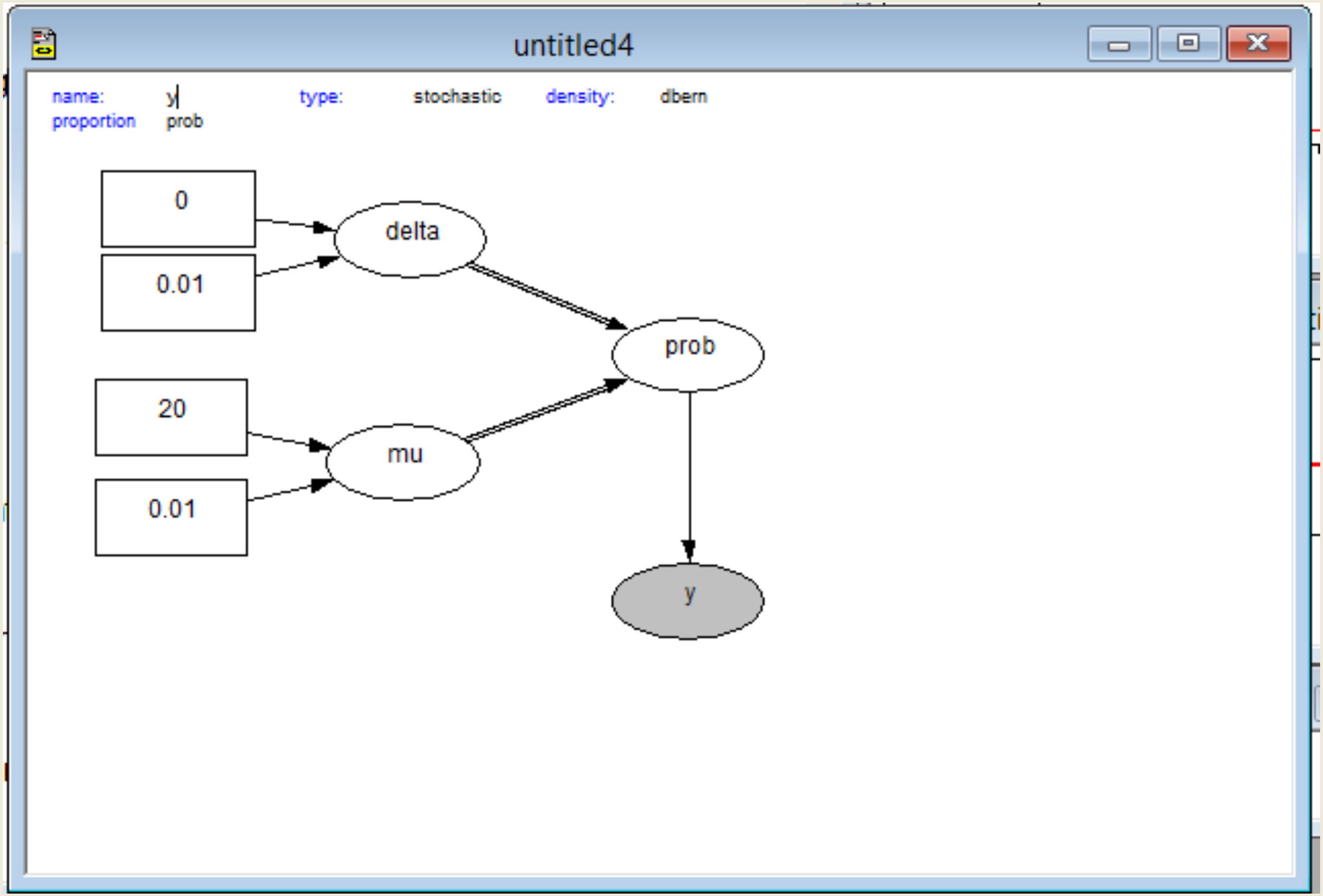
# Perspective

1. I am a regular user of Stata, Stan and R. Sometimes I have to use Python. Hey, I have a kid to feed.
2. I mostly work in biomedical research and training
3. I am especially active in Bayesian meta-analysis at present
4. I mostly do Bayes the Gelman way (WIPs, predictive checking, Hamiltonian Monte Carlo)
5. Like Spiegelhalter, I think probability does not exist, so we can all get over the old philosophical fights

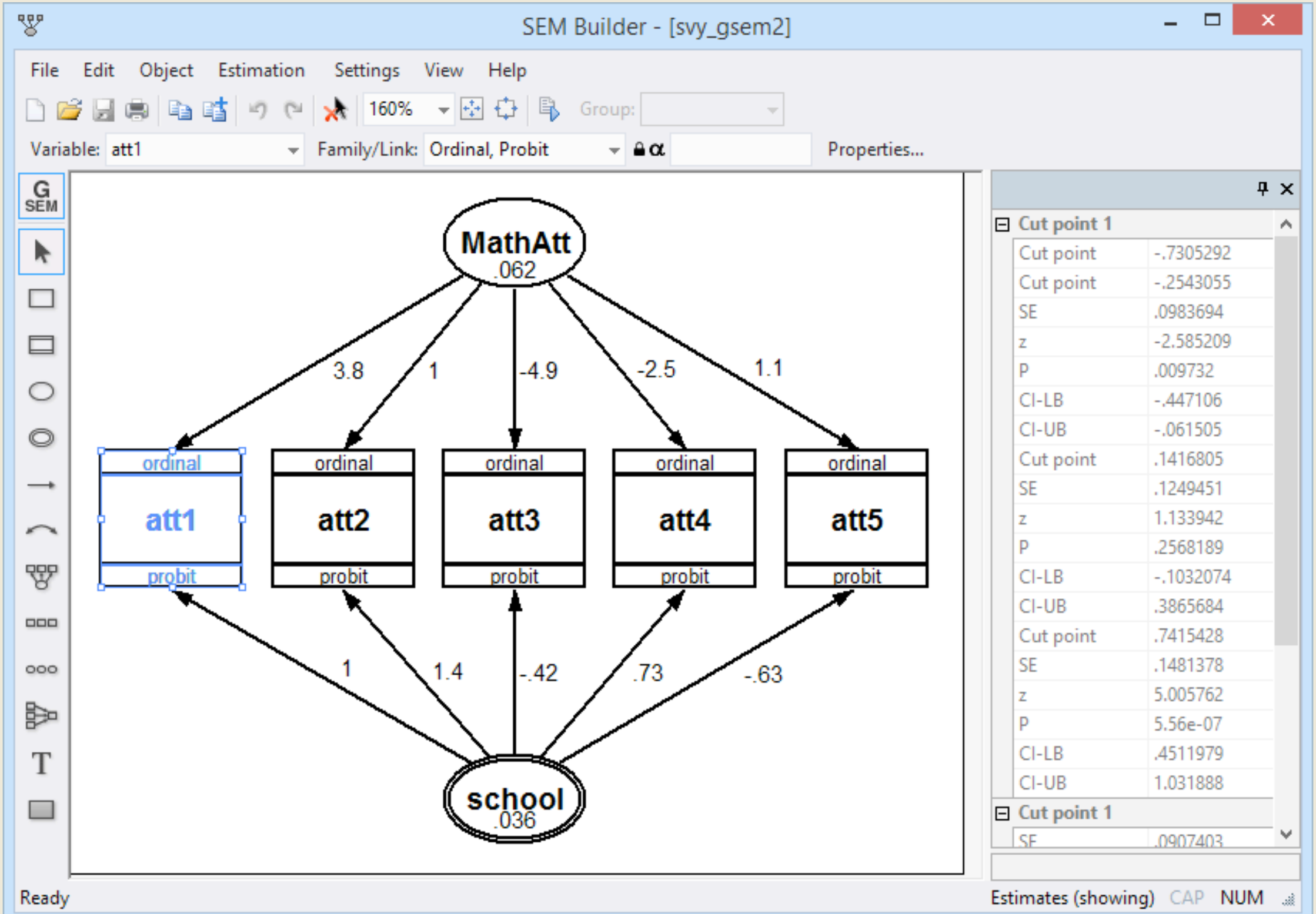
# Interface

1. Provide options for beginners, intermediate, advanced (making transitions as easy as possible)
2. Almost all options now are code-driven
3. Setting up the model, and/or sampling, and/or examining the output?
4. WinBUGS doodle was interesting but never that popular (early adopter bias?)
5. But… increasing use of GUIs in the data science workflow and autoML market
6. And… more causal inference = more DAGs…

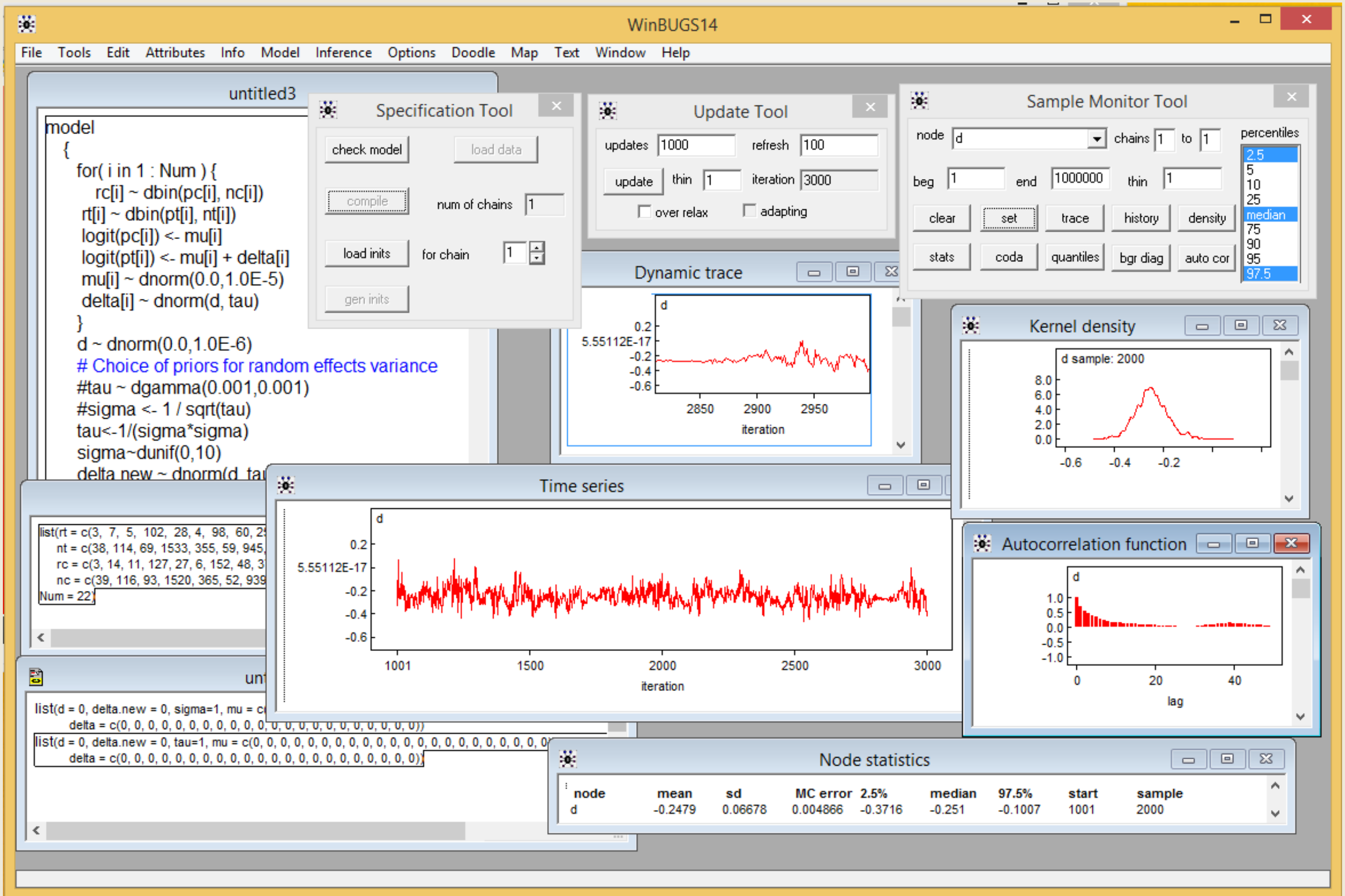
# WinBUGS doodle



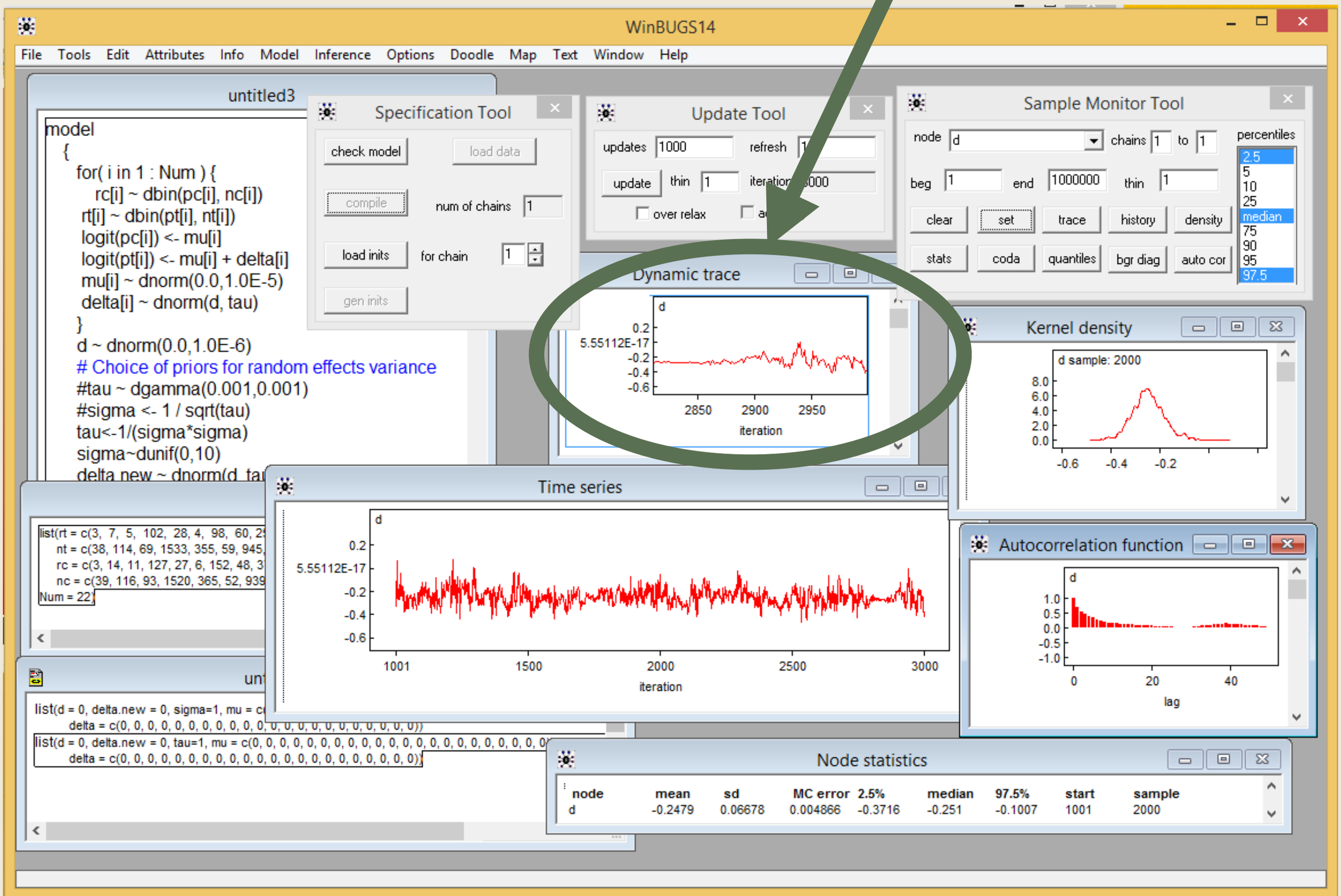
# Stata GSEM GUI



# WinBUGS main interface



# Wait, is that a live traceplot??



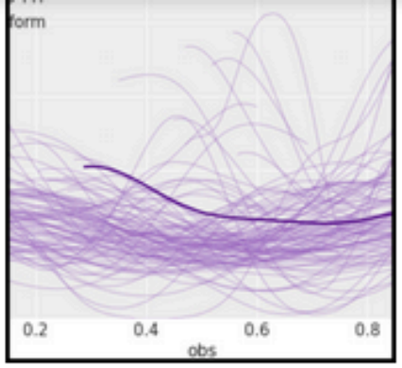
# ShinyStan


Save & Close
SHINYSTAN
DIAGNOSE
ESTIMATE
EXPLORE
MORE
Select parameter
mu
Rhat n_eff mean sd 2.5% 50% 97.5%
1 695 7.61 5.26 -2.45 7.66 18.05
Multiview
Include warmup
Bivariate
Trivariate
Density
Histogram
Kernel Density Estimate
-20 0 20 40
mu
Autocorrelation
ac
0.5
0
0 10 20
lag
Trace
40
20
0
-20
1

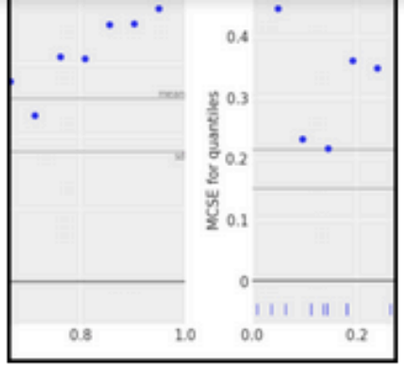
# ShinyStan


NUTS (plots)
HMC/NUTS (stats)
$\hat{R}, n_{eff}, \text{se}_{mean}$
Autocorrelation
PPcheck
All chains
0
Parameter
tau
Transformation
identity
Transform
WARNINGS -- Diverging error: 11 iterations.
By model parameter
Sample information
Treedepth information
Divergence information
Step size information
Help
Use your mouse or the sliders to select areas in the traceplot to zoom into. The other plots on the screen will update accordingly. Double-click to reset.
40
20
0
Parameter: tau
Lines are mean (solid) and median (dashed)
0
10
20
30
40
50
tau
Large red points indicate which (if any) iterations encountered a divergent transition. Yellow indicates a transition hitting the maximum treedepth.
Log Posterior
0
-10
-20
-30
0
10
20
30
40
50
Mean Metrop. Acceptance
1.00
0.75
0.50
0.25
0.00
0
10
20
30
40
50

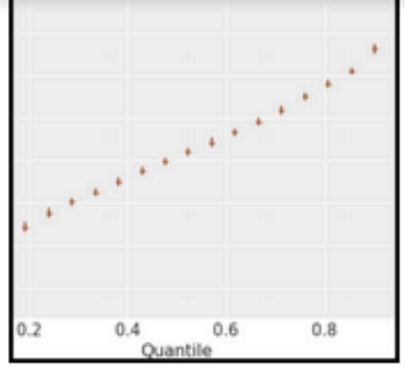
# Arviz



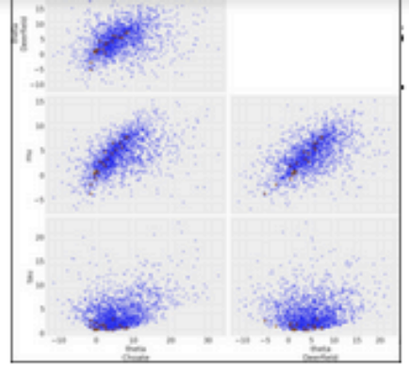
Plot LOO-PIT KDE overlaid on KDEs of



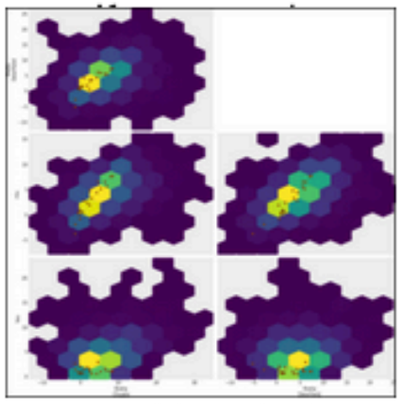
Quantile Monte Carlo Standard Error Plot



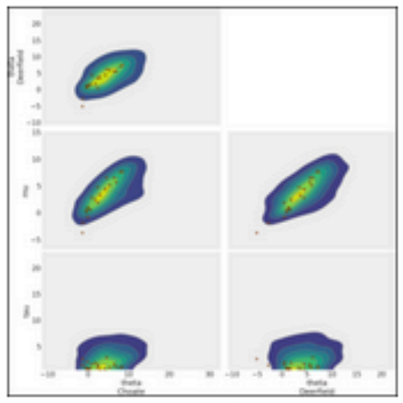
Quantile MCSE Errobar Plot



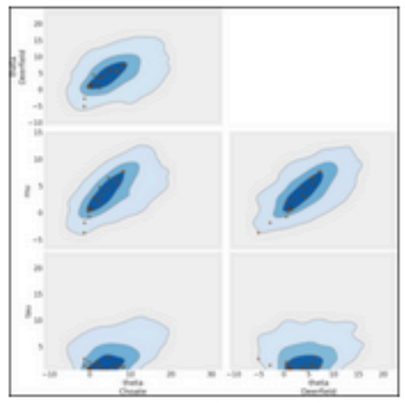
Pair Plot



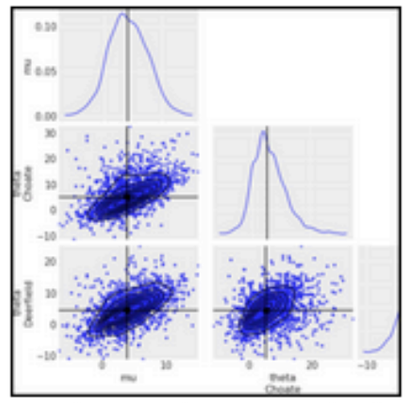
Hexbin PairPlot



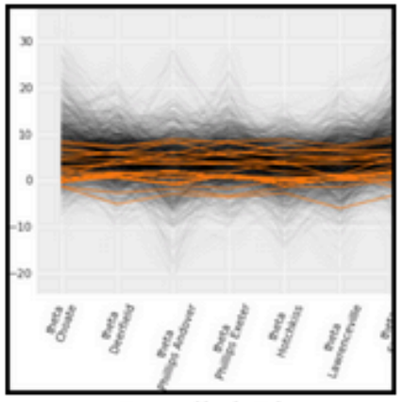
KDE Pair Plot



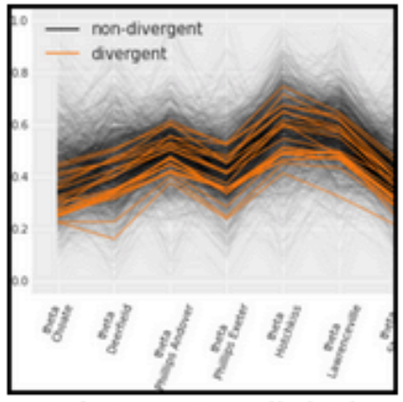
KDE Pair Plot with HDI Contours



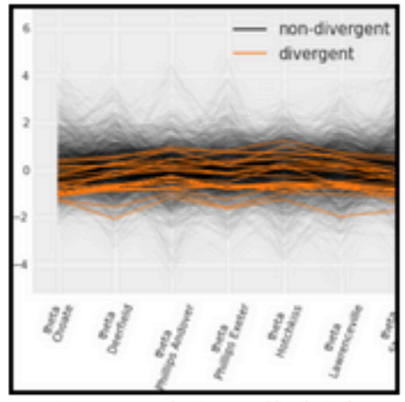
Point Estimate Pairplot



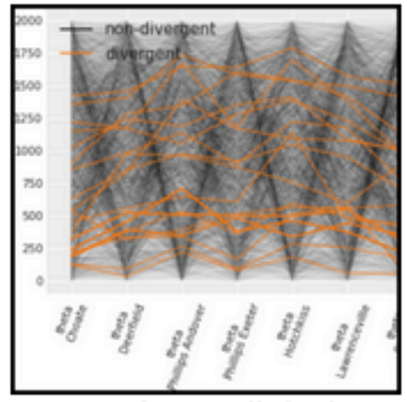
Parallel Plot



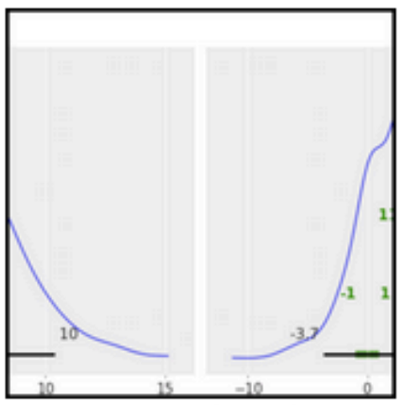
MinMax Parallel Plot



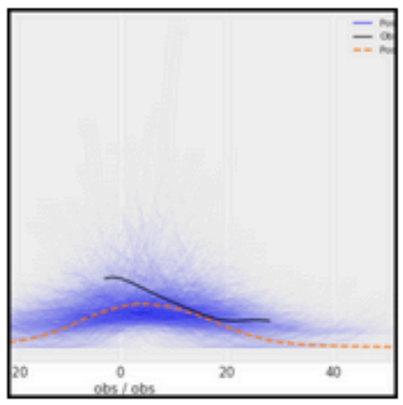
Normal Parallel Plot



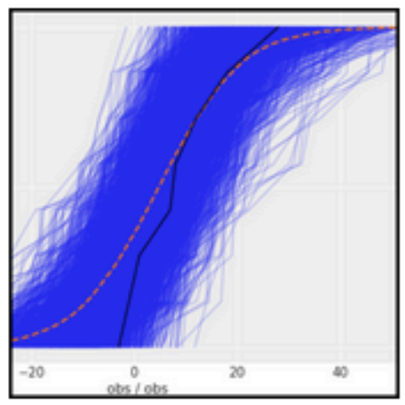
Rank Parallel Plot



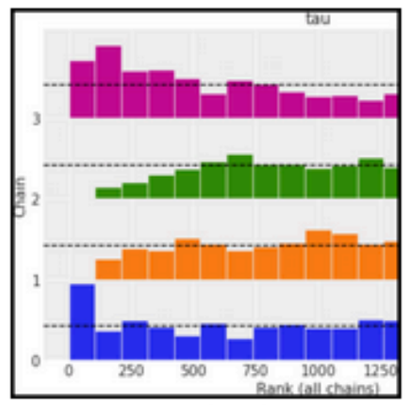
Posterior Plot



Posterior Predictive Check Plot



Posterior Predictive Check Cumulative Plot



Rank plot

# Programming (big picture)

1. Mimic familiar frequentist code (e.g. bayes:, brms)
2. Probabilistic programming language (PPL) (e.g. BUGS, JAGS, Stan, PyMC*)
3. PPL wrapped up in definition - compilation - sampling - outputs
4. Off-the-shelf preset models (e.g. bayes:, some R/Python functions that wrap PPLs, PyMC*)

# Programming (BUGS PPL)

```
or <- exp(beta1)

beta0 ~ dnorm(-0.3, 11)

beta1 ~ norm(0, 25)

for(i in 1:n) {

 logit(prob[i]) <- beta0 + beta*x[i]

 y[i] ~ dbern(prob[i])

}
```

# Programming (Stan PPL)

```
data {
 int n;
 real x[n];
 int y[n];
}
parameters{
 real beta0;
 real beta1;
}
```

```
transformed parameters {
 real or = exp(beta1);
}
model {
 beta0 ~ norm(-0.3, 0.3);
 beta1 ~ norm(0, 0.2);
 y ~ bernoulli_logit(beta0 +
beta*x);
 }
```

# Programming (Stan PPL)

```
// old:

 y ~ bernoulli_logit(beta0 +
beta*x);
```

```
// new:

target += bernoulli_lpmf(y |
beta0 + beta*x);
```

# Programming (PyMC3 preset)

```
with pm.Model() as
logistic_model:

    pm.glm.GLM.from_formula(
            "y ~ x", data,
family=pm.glm.families.Binomial()
    )
```

# Programming (big picture) for Stata

1. Off-the-shelf preset models (e.g. bayes:, rstanarm, rethinking) ✓
2. PPL seems like a must. priorspec and likelihoodspec don't really cut it
3. PPL could be implemented like embedding Python or dynamic documents, perhaps with a Mata version for tricky specifications (PyMC4 switched to a TensorFlow backend for this reason)
4. Wrapped up in definition - compilation - sampling - outputs: the advantage to this more OOP thinking would be time saving through compilation. But not everyone wants it.

# Programming (detail)

1. Knowns and unknowns are not just data and parameters
2. Deterministic and probabilistic statements (a la BUGS), or log-posterior increments (a la Stan 3.0)?
3. How do you like your eggs? OOP or functional?

# Programming (detail) for Stata

1. Knowns and unknowns are not just data and parameters — this suggests extension of the {parametername} syntax, for example for coarsened data

2. Deterministic and probabilistic statements (a la BUGS), or log-posterior increments (a la Stan 3.0)? Easy to offer increments, even when masquerading as ~ statements; building and checking a DAG is (somewhat) harder

3. OOP or functional? Mata is OOP, but do Stata users think that way?

4. Allowing Mata functions to be passed to priorspec / likelihoodspec could be a middle ground, but feels unusually functional

# Quick start

1. User base is growing.
2. They need simple interfaces. (Don’t be fooled by conference/ blog bias)
3. Mimicking familiar non-Bayesian code is a popular way (e.g. brms)
4. You got this (bayes:) 👍
5. But as they learn, they want bespoke models rather than presets.

# Quick start

6. Nobody likes software that won’t grow with them.

7. Any jump from, for example, GUI to PPL, must be as painless as possible.

8. Suppose  they could export some Stata/Mata PPL code for their model, then tweak it. Or get it graphically in the Bayes Model Builder GUI???

# Specific applications

1. I found meta-analysts were adopting Bayes only when nothing else would (easily) do the job, and only at the last possible minute, specifically for network meta-analysis.[1]

2. There was very widespread copy-and-paste of "default" code from NICE DSU, suggesting that a library of such code snippets would be enjoyed.

3. The “BUGS Examples” have been adapted many, many times in just this way. Of course, beginners don't know when default code will cause them problems...

# Specific applications

4. Stata has strengths in time series and forecasting, which should be targeted (viz Prophet, which remains highly popular, cf Bob Muenchen, despite very limited modelling options).[2]

5. Gaussian processes have been very popular in Stan, and could be implemented without great difficulty. This could link to Stata spatial functionality.

6. It's unclear whether areas with rapid Bayes growth, e.g. archaeology, but little or no Stata use, would respond to their kinds of models becoming available as Stata presets. They may already have invested too much in open-source alternatives, although there are also often long-standing and unsatisfactory specialist Windows applications (viz OxCal)

# Sampler algorithms

1. Random walk Metropolis-Hastings is very inefficient and struggles badly with correlated posteriors [3]
2. Gibbs is a bit better
3. Stata added some bespoke programming to deal with random effects in v14.1 which helped a lot… for specific models
4. Hamiltonian Monte Carlo is popular, and essential for moderate-to-large multilevel models (and other correlated posteriors). It needs gradients; Stan does this by autodiff.
5. Once you have autodiff, you can easily add Metropolis-Adjusted Langevin diffusions and piecewise deterministic Markov processes too.

# Sampler algorithms

6. Sequential Monte Carlo and Sampling-importance-resampling are fast *sometimes*; they could be fairly easy additions.

7. Building everything in-house from scratch is slow

8. ABC is too bespoke each time. But there could be a small set of models provided for agent-based modelling in microeconometrics, or population genetics

9. Lots of people are using approximations like INLA, variational inference, pseudo-marginal likelihood and synthetic likelihood (maybe they shouldn't)

10. My medium-term money's on piecewise deterministic Markov processes (e.g. zig-zag sampler)[4,5]

# Interoperability

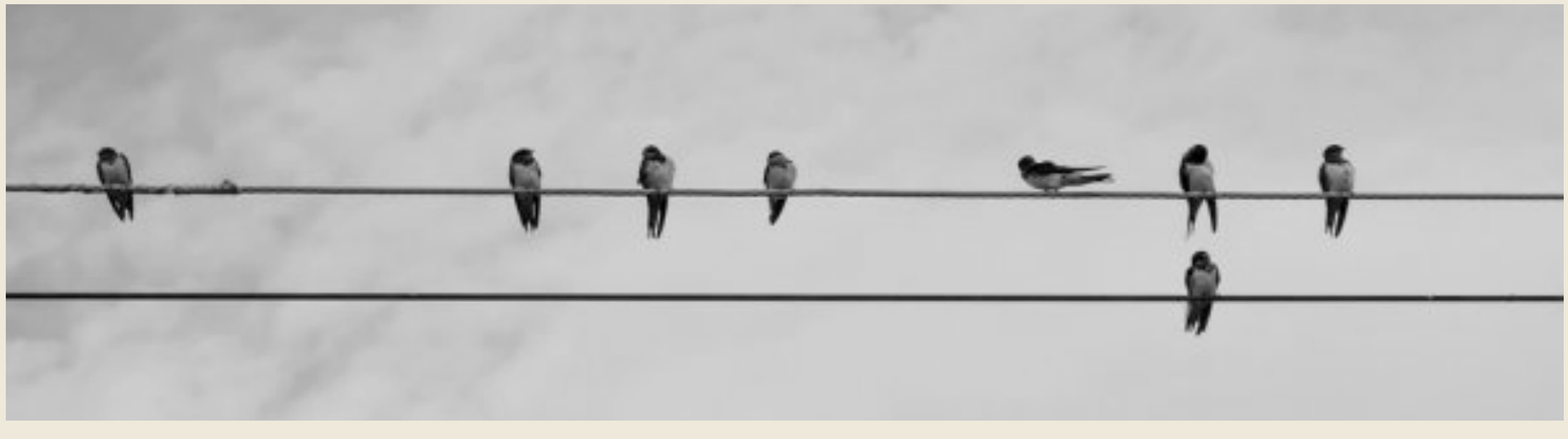
1. Much of this is likely to be bespoke and done through Python. Stata should make sure that calling Stata for a Bayesian model is at least as simple as using CmdStanPy or PyMC*.

2. Code translation/transpilation: Suppose you could just drop in some BUGS/Stan/PyMC code and have Stata interpret and run it? Or have Stata write your bayes: code out into a PyMC model object?

3. Make sure people can send a Stata posterior straight to coda / bayesplot / Arviz / ShinyStan, i.e. export to relevant R/Python/ Julia classes from PyMC*, rstan etc etc.

4. And don’t forget all the R people

# Guidance, community, support

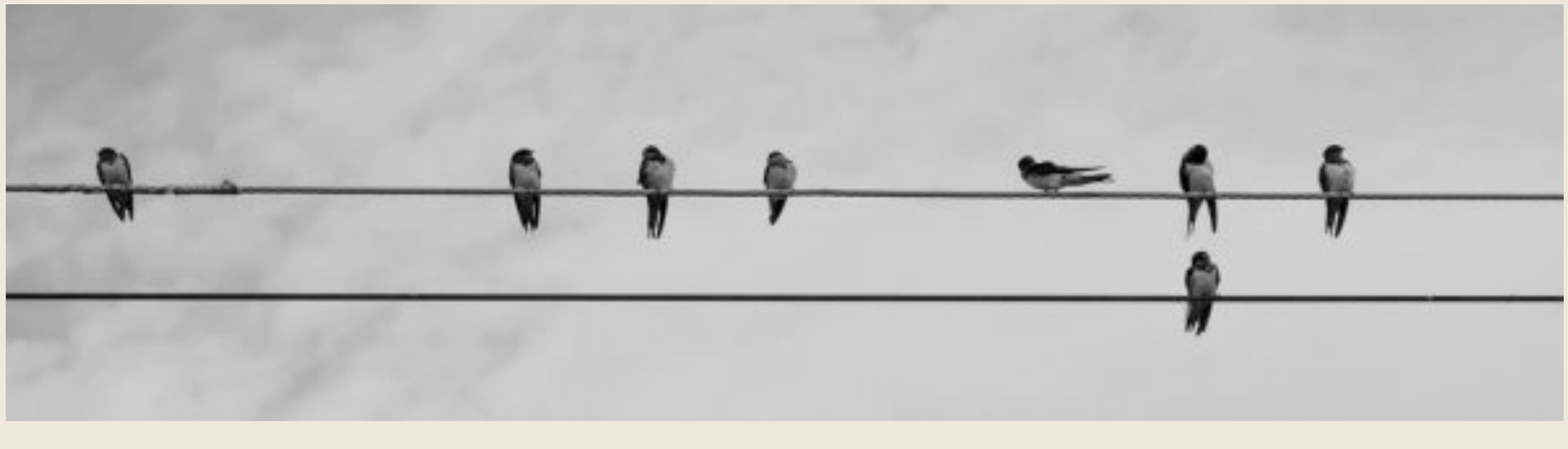


1. Stata community is excellent, better than anything in R, Python. Bayesian groups are active and densely connected if a tad obsessed with innovative methodology.

2. Although Bayes has been a headline feature in recent Stata releases, information such as videos and webinars have been limited to simple cases, where the need for Bayes is less obvious.

3. Need to update John Thompson's literature.

# Guidance, community, support



4. Will Statalist need a Bayesian sub-forum one day?

5. Consensus prior elicitation tools like SHELF could be adapted for a Stata version with authors' consent.

6. I suspect that something SHELF-like will just not be used because researchers want the protection of using a "validated" method.

# Crypto-Bayesians

Some people are secretive about their Bayesian analyses:

1. multiple imputation

2. network meta-analysis
   without thinking about priors

3. xt… postestimation BLUPs

Should this be integrated more with the Bayesian features?

# Top tips

1. DAG-based model builder GUI based on GSEM
2. Add PPL to extend priorspec and likelihoodspec by incrementing posterior in ado or Mata (or give an alternative to bayesmh)
3. Make more realistically complex tutorials and bring out some new books
4. Import/export BUGS, Stan, PyMC* model code, and Python objects for PyMC*, JAGS, PyStan
5. Consider new preset models for new markets
6. Develop autodiff code, thence HMC and maybe other samplers

# References


1. Grant RL. The uptake of Bayesian methods in biomedical meta-analyses: a scoping review, 2005-2016. Journal of Evidence-Based Medicine (2019); 12(1): 69-75. http://www.robertgrantstats.co.uk/papers/bayes-MA-scoping-review-postprint.pdf

2. Bob Muenchen, "The popularity of data science software" r4stats.com/articles/popularity/

3. Green PJ, Łatuszyński K, Pereyra M, Robert CP. Bayesian Computation: A Summary of the Current State, and Samples Backwards and Forwards. Statistics and Computing (2015);25:835–862.

4. Bierkens J, Fearnhead P, Roberts G. The Zig-Zag process and super-efficient sampling for Bayesian analysis of big data. Annals of Statistics (2019);47(3):1288–1320.

5. Fearnhead P, Bierkens J, Pollock M, Roberts GO. Piecewise Deterministic Markov Processes for Continuous-Time Monte Carlo. Statistical Science (2018);33(3):386–412.

The BUGS Book, CRC Press.

Stan User Guide, mc-stan.org/users/documentation

PyMC3, https://docs.pymc.io/

# Thanks for listening

robert@bayescamp.com

I knew I should have used Stata