Bayesian Analysis of Software Engineering Data

Robert Feldt

robert.feldt@chalmers.se

Chalmers University of Technology, Sweden

Int. Summer School on Search- and Machine Learning-based Software Engineering

Cordoba, Spain 2022-06-22

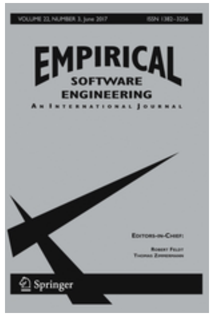
@drfeldt

About **Me**

- Originally from Gothenburg, Sweden

- Programmer/hacker for 40+ years, sold my first program at age 13
- **Broad interests**: Software Engineering, (Applied) Machine Learning & Artificial Intelligence, SW Testing, Human factors & Psychology, Optimisation & Search, Info. theory & Statistics/Math & Visualisation, Methodology in SE

- Master of Computer Science and Engineering 1997, Phd in Software Engineering 2002, Full prof since 2013 (both at Chalmers and BTH)
- Co-founded 4 companies, worked as consultant in SE and Applied AI/ML since 1992
- One Korean patent (testing DNNs) granted, one Chinese patent in submission
- co-Editor-in-Chief for Springer's Journal of Empirical Software Engineering since 2017



Credit where it is due: our ICSE 2021 tutorial & joint papers

Part 1: Why?

Part 2: How?

Part 3: Now what?

Part 1: Why Should I Use Bayesian Data Analysis?

When should I not use Bayesian data analysis?

Ten reasons for using Bayesian data analysis

When Bayesian statistics are at a disadvantage

Performance:
fitting may take a long time

Effort:
more modeling work & learning is needed

Research standards:
your research area may require frequentist statistics

Ten reasons
for using Bayesian data analysis

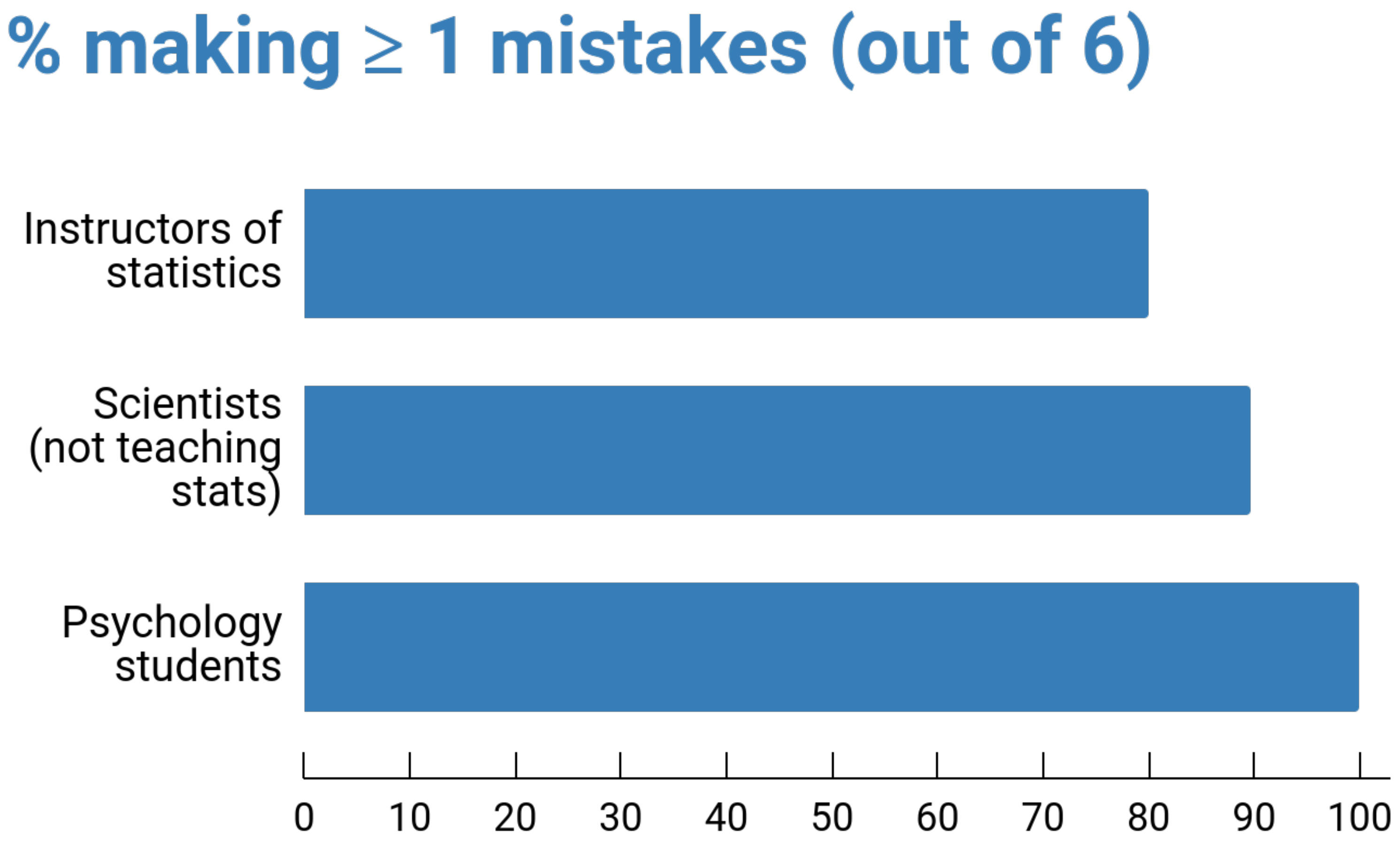
Ten key reasons for using Bayesian data analysis

1. Beware of the replication crisis
2. More and more disciplines find practical value in Bayesian techniques
3. Bayesian models are easier to understand
4. Avoid dichotomous (yes/no) reasoning
5. Safeguard against overfitting
6. Quantitative distributional information
7. Flexibility of modeling
8. Find fitting problems and test assumptions
9. Let's talk practical significance
10. Plan and connect follow-up studies

#3
Bayesian models are easier to understand

Ten reasons for using Bayesian data analysis

Misunderstanding *p*-values



Haller and Krauss: Misinterpretation of significance, 2002

Misunderstanding confidence intervals

Both researchers and students endorsed, on average, more than three [incorrect] statements [out of six], indicating a gross misunderstanding of CIs.

> Psychon Bull Rev. 2014 Oct;21(5):1157-64. doi: 10.3758/s13423-013-0572-3.

Robust misinterpretation of confidence intervals

Rink Hoekstra [1], Richard D Morey, Jeffrey N Rouder, Eric-Jan Wagenmakers

#4
Avoid dichotomous (yes/no) reasoning

Ten reasons for using Bayesian data analysis

Yes-or-no scientific questions

Scientific conclusions and business or policy decisions should not be based only on whether a p-value passes a specific threshold.

Editorial

The ASA Statement on *p*-Values: Context, Process, and Purpose

Ronald L. Wasserstein & Nicole A. Lazar

Pages 129-133 | Accepted author version posted online: 07 Mar 2016, Published online:09 Jun 2016

Taylor & Francis
Taylor & Francis Group

The American Statistician

ISSN: 0003-1305 (Print) 1537-2731 (Online) Journal homepage: https://www.tandfonline.com/loi/utas20

Moving to a World Beyond "$p < 0.05$"

Ronald L. Wasserstein, Allen L. Schirm & Nicole A. Lazar

To cite this article: Ronald L. Wasserstein, Allen L. Schirm & Nicole A. Lazar (2019) Moving to a World Beyond "$p<0.05$", The American Statistician, 73:sup1, 1-19, DOI: 10.1080/00031305.2019.1583913

To link to this article: https://doi.org/10.1080/00031305.2019.1583913

2. Don't Say "Statistically Significant"

The *ASA Statement on P-Values and Statistical Significance* stopped just short of recommending that declarations of "statistical significance" be abandoned. We take that step here. We conclude, based on our review of the articles in this special issue and the broader literature, that it is time to stop using the term "statistically significant" entirely. Nor should variants such as "significantly different," "$p < 0.05$," and "nonsignificant" survive, whether expressed in words, by asterisks in a table, or in some other way.

We are calling for a stop to the use of P values in the conventional, dichotomous way — to decide whether a result refutes or supports a scientific hypothesis.

nature

COMMENT · 20 MARCH 2019

Scientists rise up against statistical significance

Valentin Amrhein, Sander Greenland, Blake McShane and more than 800 signatories call for an end to hyped claims and the dismissal of possibly crucial effects.

Valentin Amrhein, Sander Greenland & Blake McShane

#5
Safeguard against overfitting

Ten reasons for using Bayesian data analysis

The risk of overfitting

Techniques against overfitting

Regularizing priors:

Do not learn only from the data, utilise also existing knowledge

Partial pooling:

Propagate information across data clusters

Out-of-sample predictive effectiveness:

Penalize models that make poor predictions

#6
Quantitative distributional information

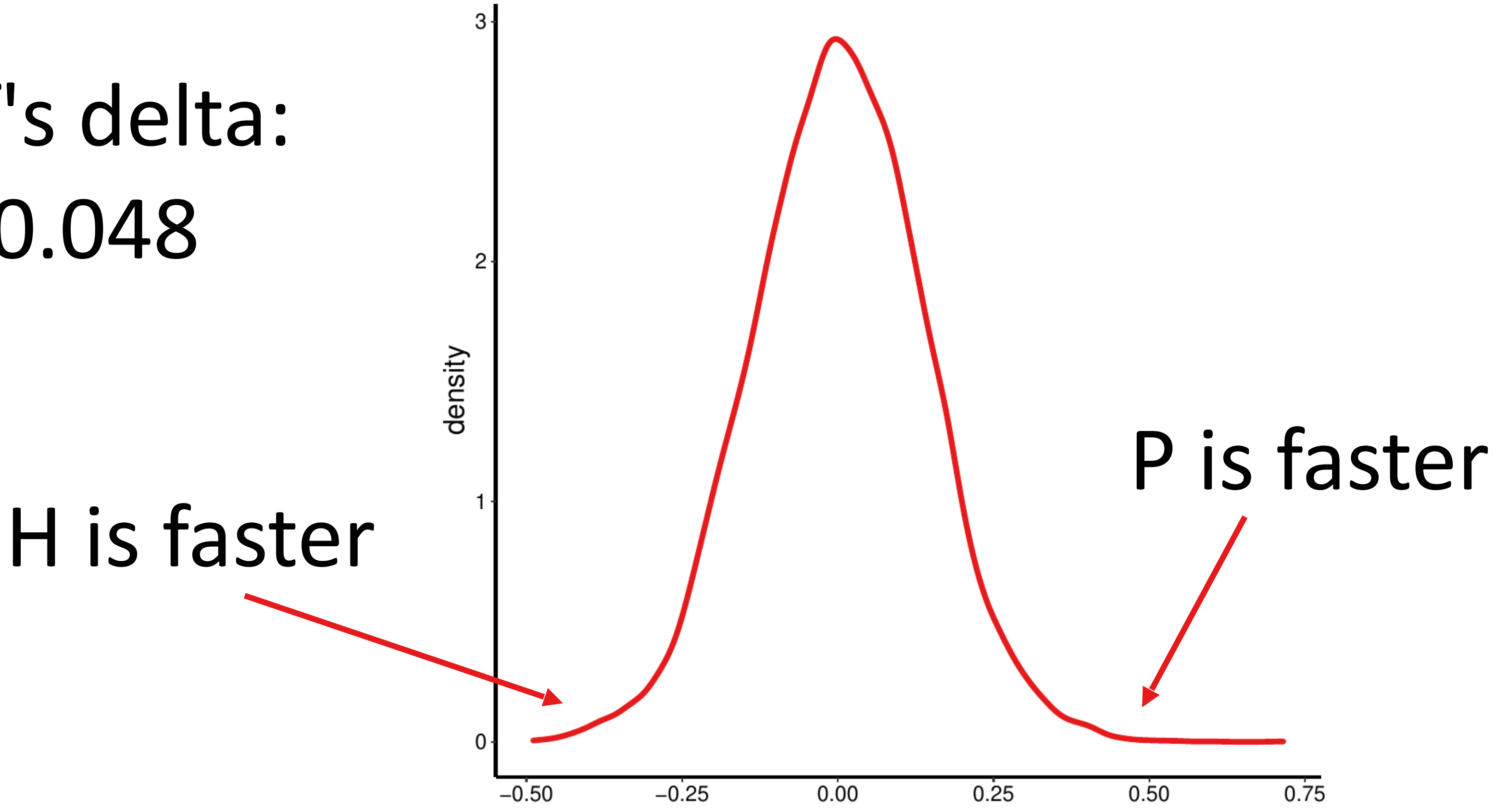
Ten reasons for using Bayesian data analysis

Outcomes of a statistical analysis

Is language H usually faster than language P?

p-value:
$p = 0.334$

Cliff's delta:
$\delta = 0.048$

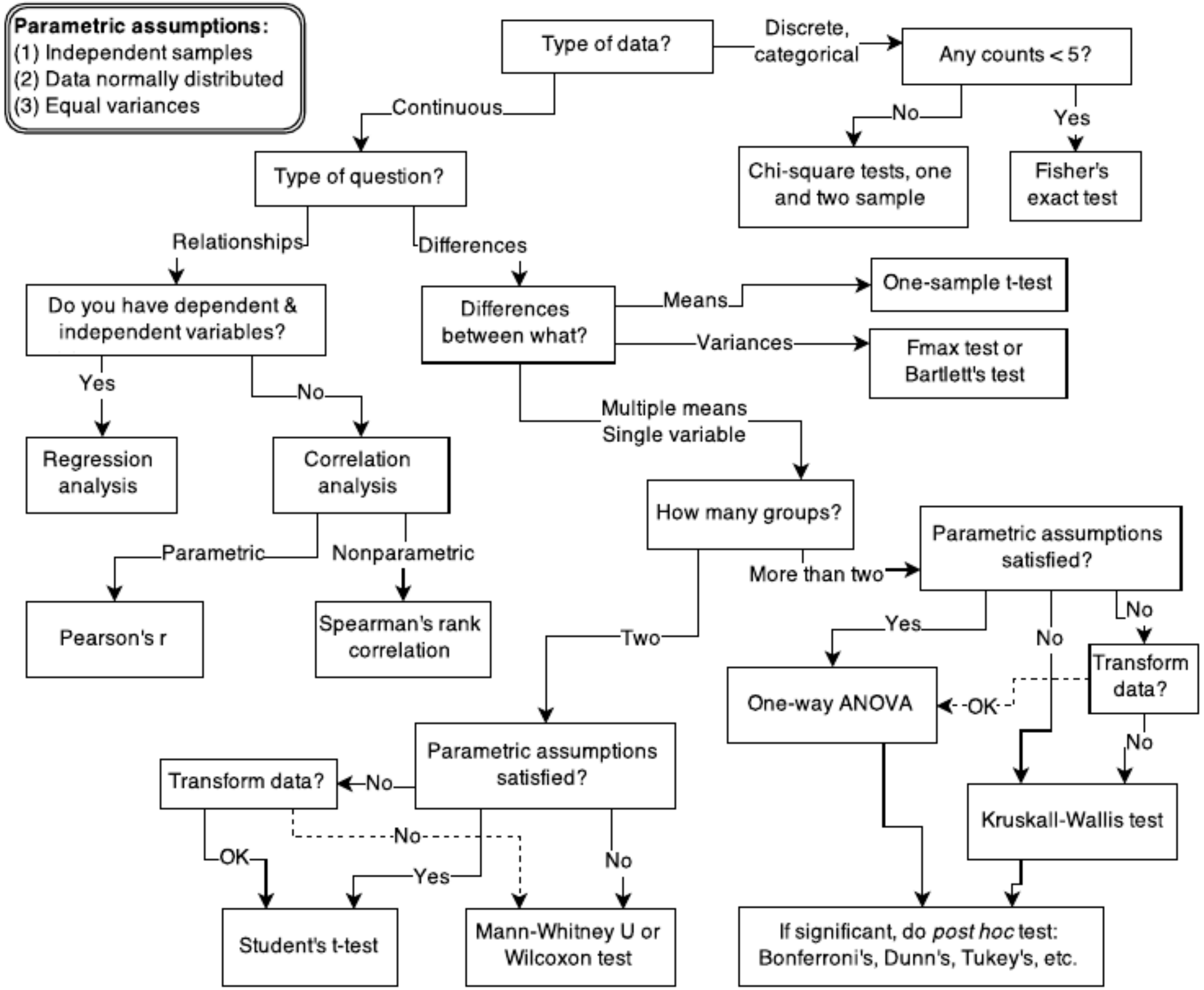


#7
Flexibility of modeling

Ten reasons for using Bayesian data analysis

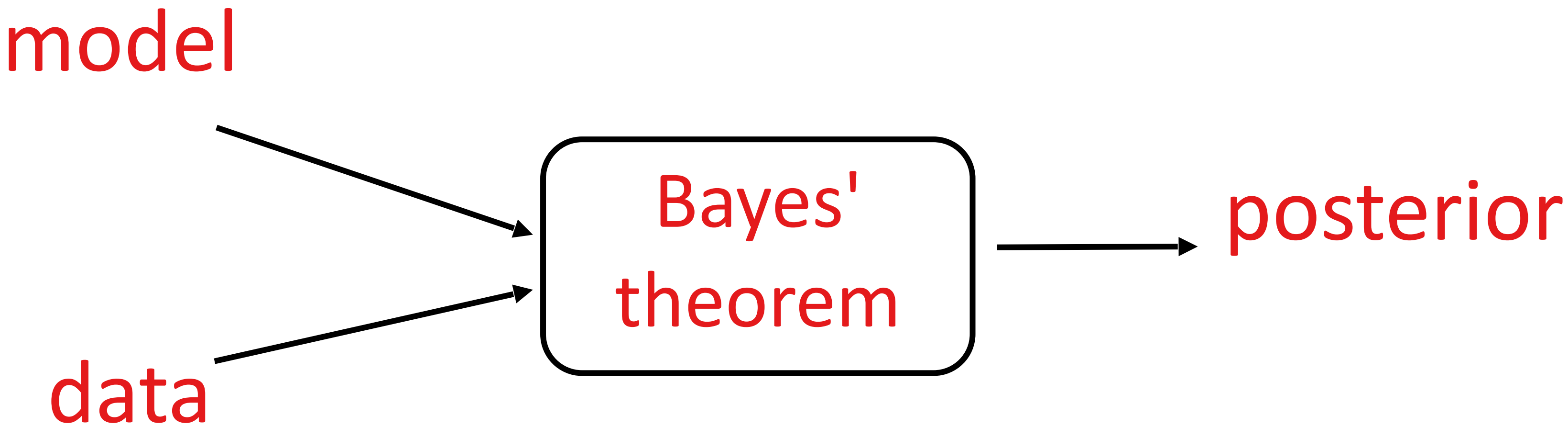
Frequentist statistics:

What statistical model and pre-canned test should you use?



Source: R. McElreath, Statistical Rethinking (2nd ed), CRC press, 2020

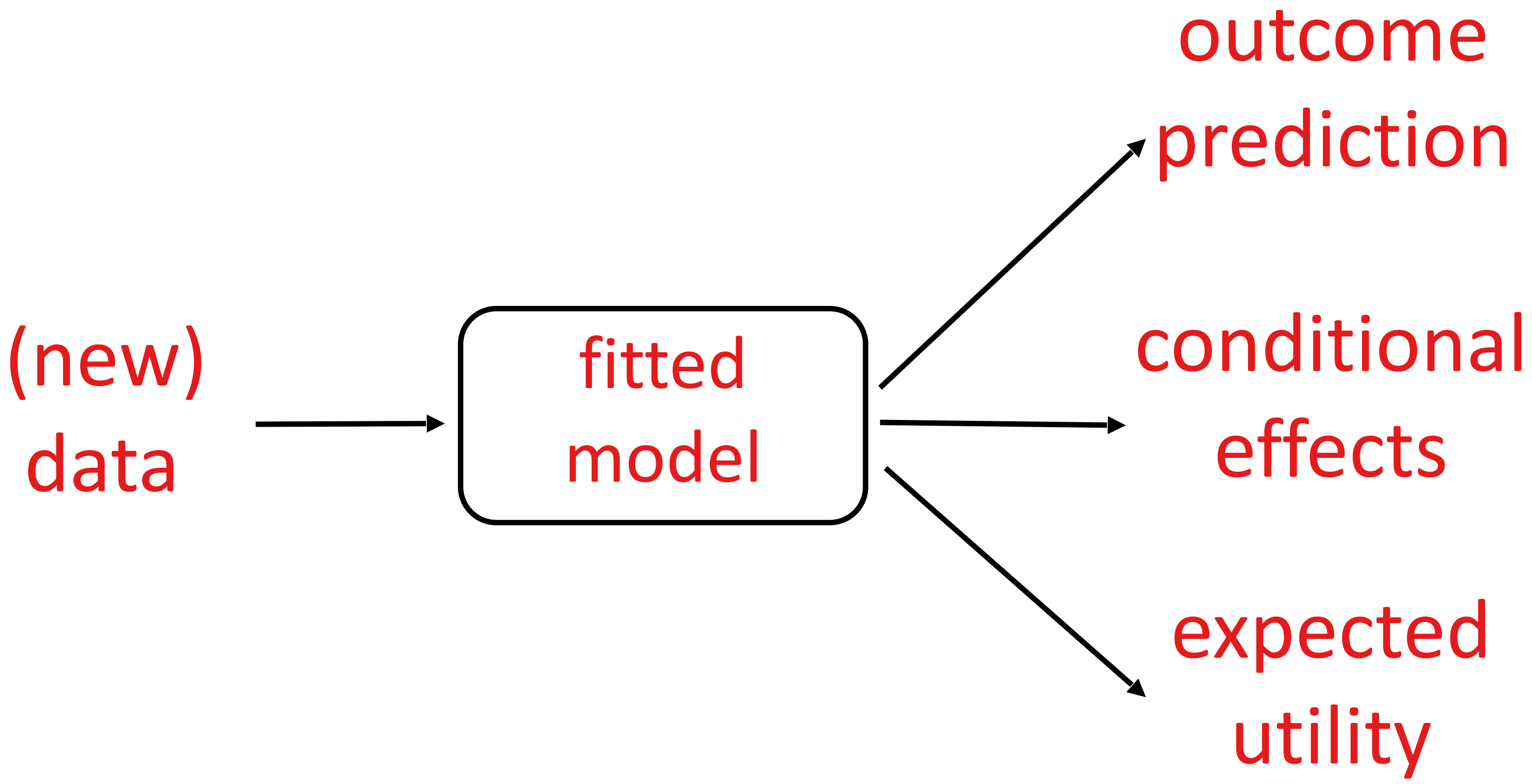
One rule to rule them all



#9
Let's talk practical significance

Ten reasons for using Bayesian data analysis

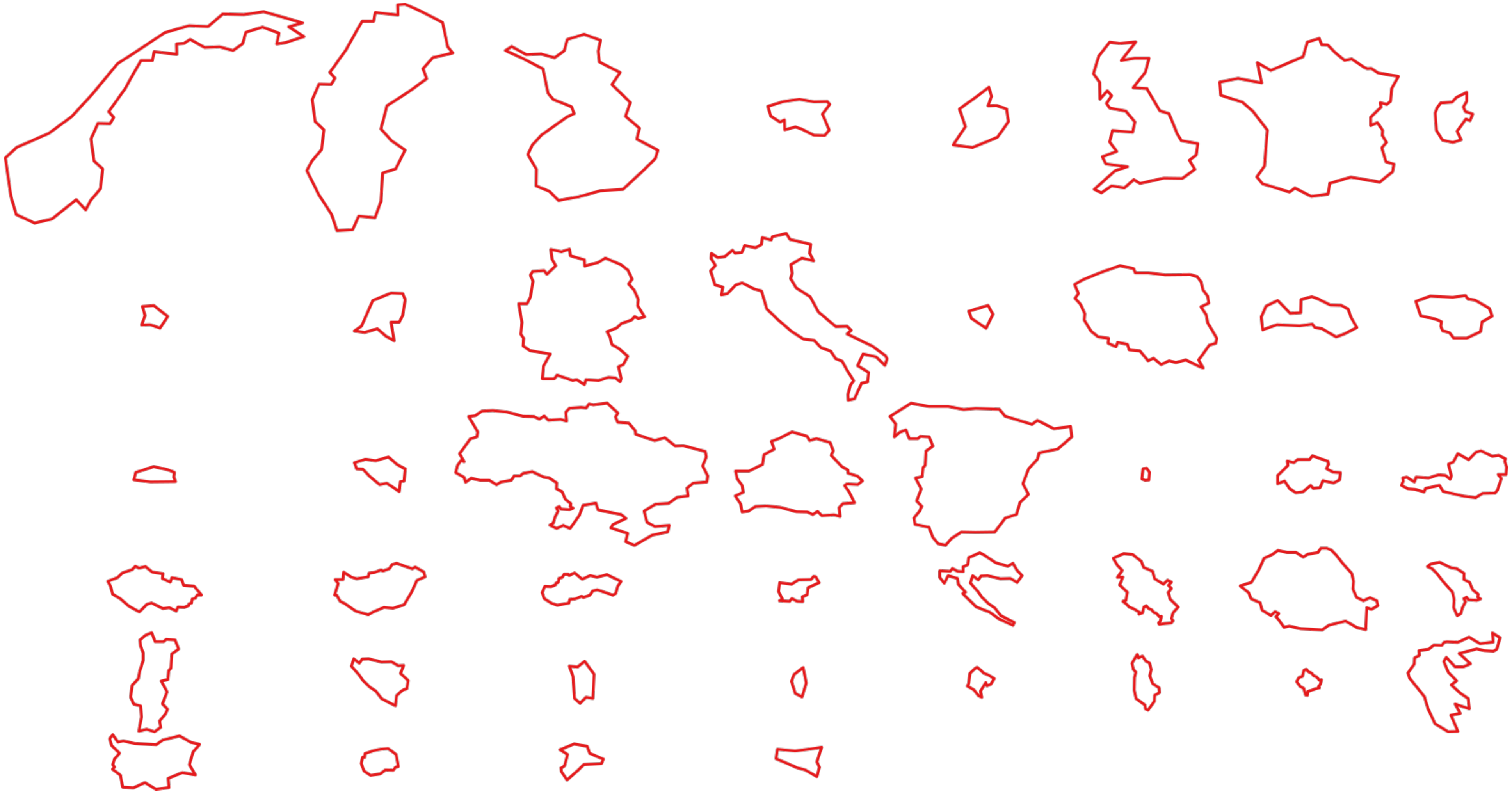
Derived distributions



#10
Plan and connect follow-up studies

Ten reasons for using Bayesian data analysis

From isolated studies...



To a connected whole

Part 2: How does Bayesian Data Analysis work?

prior $\times$ likelihood $\propto$ posterior

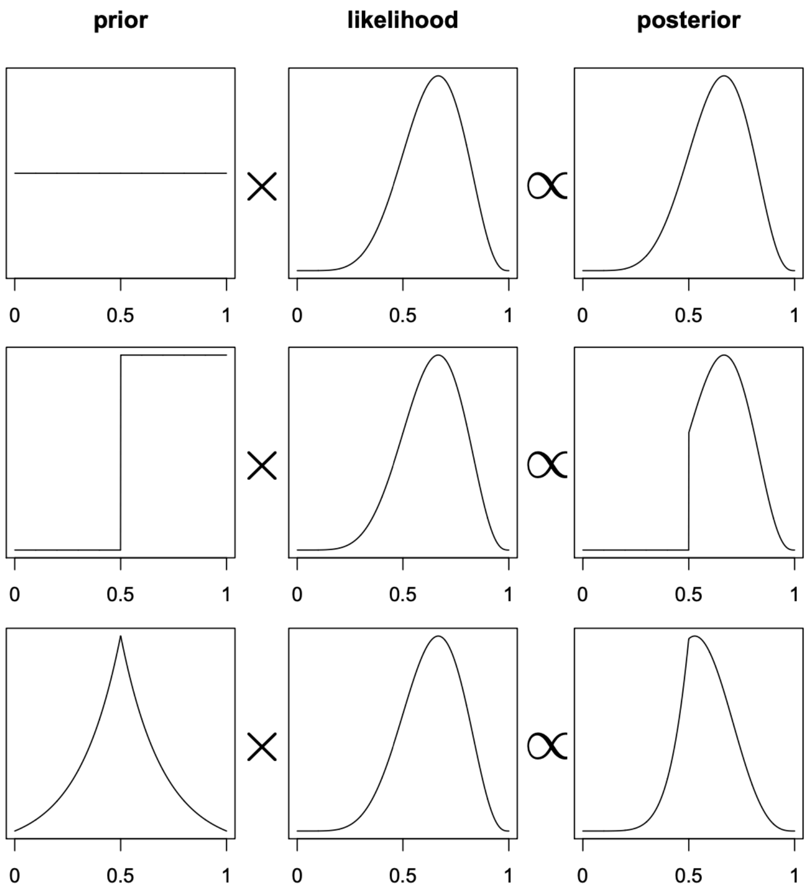


Figure from McElreath, 2020

- Prior knowledge
- Likelihood encodes our assumptions (& knowledge/hypothesis/theories) about the data generation process
- Multiplying the two leads to a posterior distribution
- Posterior typically joint (>1 parameter) so not only (marginal) distributions of multiple individual parameters

Example SBSE problem: Which algorithm is best?

- Compare performance of 5 different search/optimization algorithms and answer:
  - RQ. Which search/optimization algorithm performs best?
  - On a set of (5) benchmark problems,
  - Of dimensions in range 2 to 300,
  - With fixed time controls (1 second + 0.2 second * dimension)
- Some challenges:
  - 5*5*300=7500 settings
  - Stochastic algorithms => repeated runs
  - We could easily be looking at many days of optimisation runs!

The idea: (pair-wise) Sampling + Statistics

- Rather than running all settings:
  - Pair-wise comparisons on random problem of random dimension
  - Learn a statistical model that can answer our question(s)!

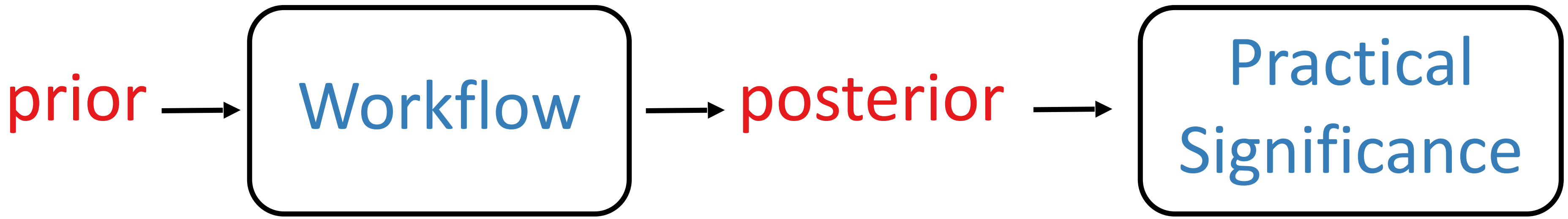
```
julia> Algs
5-element Vector{String}:
 "adaptive_de_rand_1_bin_radiuslimited"
 "de_rand_2_bin"
 "generating_set_search"
 "probabilistic_descent"
 "random_search"
```

```
julia> Problems
5-element Vector{String31}:
 "ackley"
 "deceptive_cuccu2011_15_2"
 "griewank"
 "rastrigin"
 "rosenbrock"
```

Problem	Dim	Time	Alg1	Alg2	Winner	F1	F2
ackley	6	2.2	generating_set_search	random_search	0	3.7697844845752115e-11	7.175133414650787
griewank	141	29.2	adaptive_de_rand_1_bin_radiuslimited	generating_set_search	1	0.012320988875106575	0.0
rosenbrock	208	42.6	adaptive_de_rand_1_bin_radiuslimited	random_search	0	6.119948637418005e-12	1.9034656814051404e9
ackley	30	7.0	de_rand_2_bin	probabilistic_descent	0	3.552713678800501e-15	17.034606228794516
griewank	6	2.2	random_search	adaptive_de_rand_1_bin_radiuslimited	1	1.2587654933394752	0.012320988875106575

Bigger picture of using BDA in SE

- In addition to motivation and steps of Bayesian inference we need:
  - A Bayesian workflow for SE data
- Using the output:
  1. Science: SE Chains of evidence, i.e. “My posterior is their prior” (& vice versa)
  2. Engineering: Practical significance of SE results



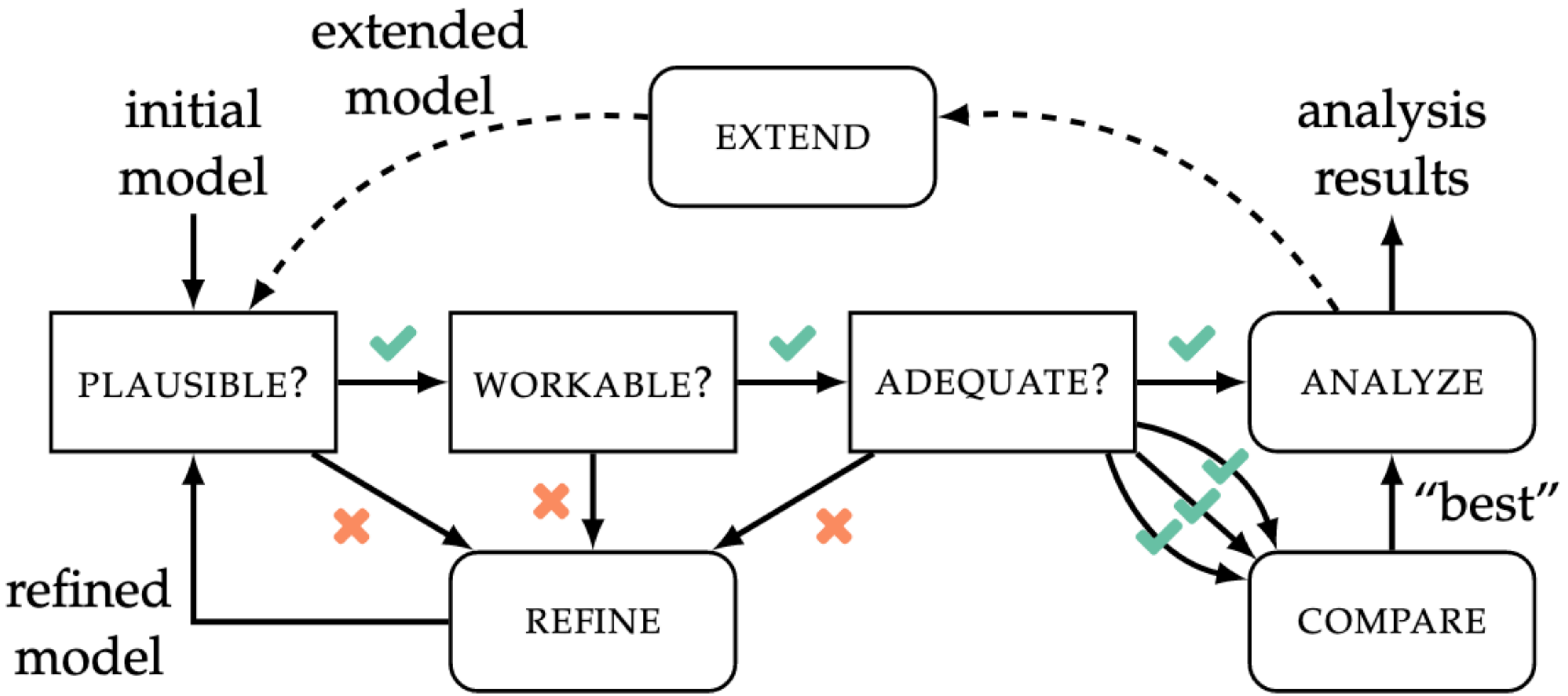
Incremental modeling is key

- We build model in steps, incrementally
  - Unlikely: find right model “level” or ”resolution” directly
  - Explore/adapt guided by results and reasoning (Agile)
- Two main starting points:
  - Simplest thinkable model (then build up = add complexity/realism)
  - Existing/frequentist model (then explore by simplifying or adding)

Applying Bayesian Analysis Guidelines to Empirical Software Engineering Data

The Case of Programming Languages and Code Quality

Carlo A. Furia[1] · Richard Torkar[2,3] · Robert Feldt[2]



STEP	PRIOR	LIKELIHOOD	ALTERNATIVE MODELS	EMPIRICAL DATA	NEW DATA FOR PREDICTION
plausible?	✓				
workable?	✓	✓			
adequate?	✓	✓		✓	
compare			✓		
analyze	✓	✓		✓	✓

Model specifications

```julia
@model function linear_regression(x, y)
    σ ~ Exponential(1)
    α ~ Normal(50, 10)
    βₗ ~ LogNormal(-5, 2)

    N = length(y)
    for n ∈ 1:N
        μ[n] ~ α + βₗ * x
    end

    y ~ Normal(μ, σ)
end
```

turing (Julia)

$$y_i \sim \mathrm{Normal}(\mu_i,\ \sigma)$$
$$\mu_i = \alpha + \beta_l \bullet x_i$$
$$\alpha \sim \mathrm{Normal}(50,10)$$
$$\beta_l \sim \mathrm{LogNormal}(-5,\ 2)$$
$$\sigma \sim \mathrm{Exponential}(1)$$

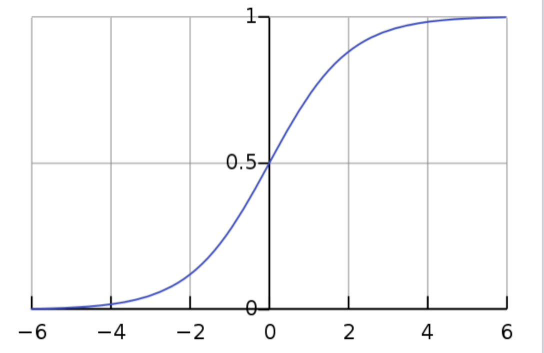
```r
ulam(
  alist(
    y ~ dnorm(mu, sigma),
    mu <- alpha + b_loc * x,
    alpha ~ dnorm(50, 10),
    b_loc ~ dlnorm(-5, 2),
    sigma ~ dexp(1)
    )
)
```

rethinking (R)

```r
brm(
    y ~ 1 + LOC,
    data = d,
    family = normal(),
    prior = c(
        set_prior(normal(50, 10), class = Intercept),
        set_prior(lognormal(-5, 2), class = b,
        set_prior(exponential(1), class = sd))
    )
)
```

brms (R)

Our first model

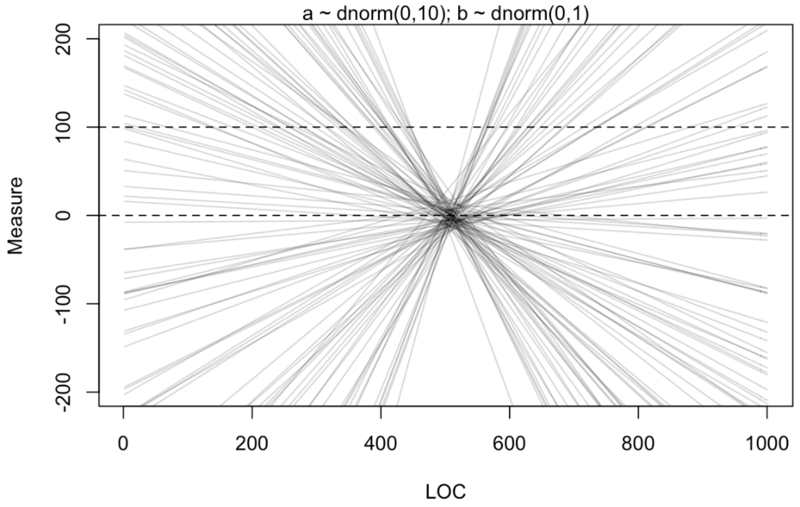


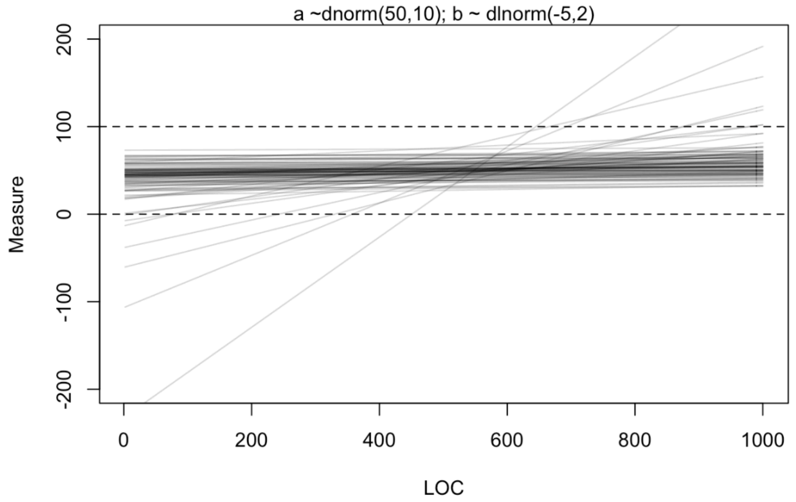
Standard logistic function where $L = 1, k = 1, x_0 = 0$

```
@model function m1(alg1, alg2, twowins)
    # Hyperparam
    s ~ Exponential(1.0)

    # Logits per algorithm
    a ~ filldist(Normal(0, s), N_alg)

    for i in 1:length(twowins)
        # Since we model if alg 2 will "win" that should be the positive logit
        v = logistic(a[alg2[i]] - a[alg1[i]])
        twowins[i] ~ Bernoulli(v)
    end
end;
```

Prior predictive checks

- Generate data from priors only
- Many graphical checks exists
- Purpose is to assess suitability of priors
- Often called sensitivity analysis, i.e., how sensitive is the model to different priors?





Prior Predictive Check

```julia
#
# Prepare data to be input in the model
#
Nobs = nrow(df)
println("Using $Nobs observations")
alg1 = Int[findfirst(==(a), Algs) for a in df.Alg1]
alg2 = Int[findfirst(==(a), Algs) for a in df.Alg2]
twowins = df.Winner


#
# Prior predictive check. We sample using only the prior and then check if looks Plausible?
#
pripd_chains = sample(m1(alg1, alg2, twowins), Prior(), MCMCThreads(), 2_000, 5)
```

```
julia> sum(rand(Bernoulli(logistic(2.88 - (-3.1))), 1000))
993
```

How do we get the posterior?

- Grid approximations
- Quadratic approximation
- Sample-Importance-Resample
- Approximate Bayesian Computation
- …
- Markov Chain Monte Carlo
  - Metropolis-Hasting
  - Gibbs
  - **Hamiltonian Monte Carlo**

Arianna Rosenbluth (1927–2020)

Stanislaw Ułam (1909–1984)

Sample to get a posterior distribution

```
# Get 10_000 samples from posterior by sampling 2000 from 5 threads:
chains = sample(m1(alg1, alg2, twowins), DynamicNUTS(), MCMCThreads(), 2_000, 5)
```

Summary Statistics

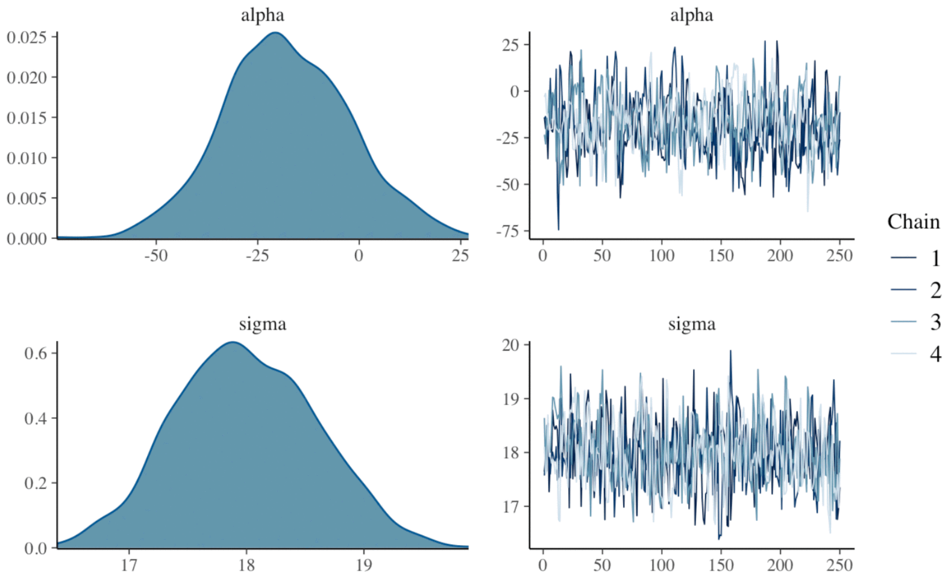
parameters	mean	std	naive_se	mcse	ess	rhat	ess_per_sec
Symbol	Float64	Float64	Float64	Float64	Float64	Float64	Float64
s	2.7122	0.9755	0.0098	0.0194	2995.6438	1.0014	1604.5226
a[1]	2.6361	1.4280	0.0143	0.0317	2047.0316	1.0028	1096.4283
a[2]	0.4509	1.4017	0.0140	0.0295	2093.4651	1.0021	1121.2989
a[3]	2.3221	1.4345	0.0143	0.0320	2036.2532	1.0030	1090.6552
a[4]	-0.6808	1.3952	0.0140	0.0288	2088.7111	1.0024	1118.7526
a[5]	-4.5471	1.9973	0.0200	0.0379	2381.7830	1.0011	1275.7274

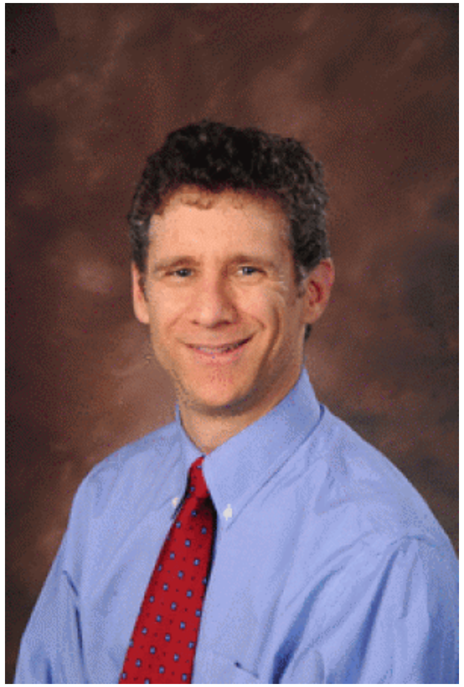
Quantiles

parameters	2.5%	25.0%	50.0%	75.0%	97.5%
Symbol	Float64	Float64	Float64	Float64	Float64
s	1.2962	2.0175	2.5326	3.2343	5.0450
a[1]	0.0827	1.7247	2.5402	3.4262	5.7826
a[2]	-2.1740	-0.4151	0.3882	1.2561	3.4435
a[3]	-0.2460	1.4206	2.2180	3.1263	5.4427
a[4]	-3.4150	-1.5294	-0.7051	0.1305	2.2568
a[5]	-9.3293	-5.5529	-4.2586	-3.2129	-1.4553

Diagnostics

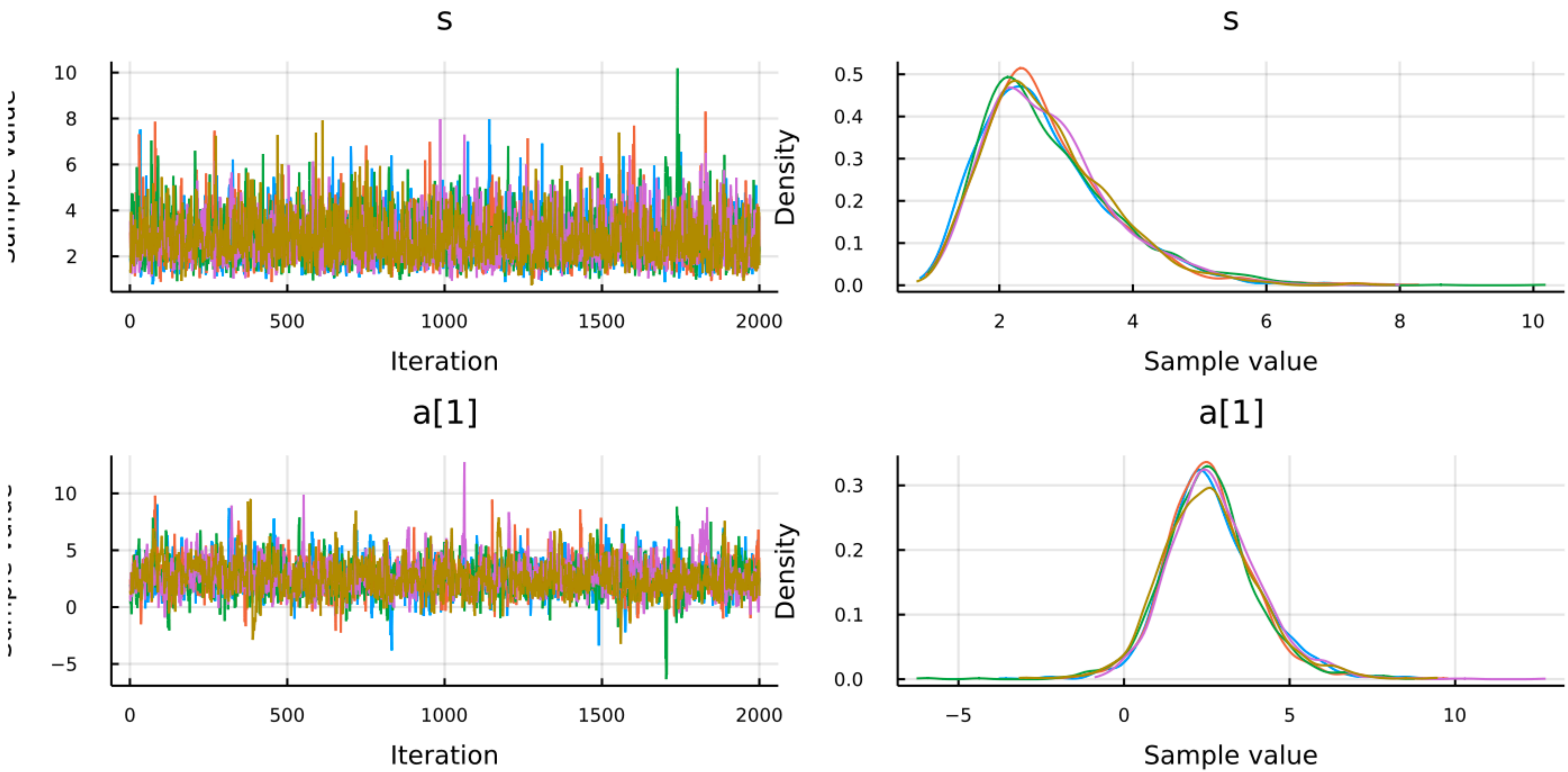
- Many diagnostics exists
    - $\hat{R}$
    - **Effective sample sizes**
    - **Traceplots**
    - Divergences
    - E-BFMI
    - Treedepth
- Sampler provides am warnings





Andrew Gelman, Columbia University, and many in the Stan team

Chains of model (m1) seems to mix well



Using the posterior

- Plot posterior probability distributions
- Compute stuff
  - Intervals
  - Point estimates
- Simulate / predict
  - Condition, i.e. fix some inputs/parameters and study difference in output

Use posterior to calculate ranks of algs

```
julia> ranks_m1 = calc_ranks(Algs, posterior_m1, "a")
10000×5 Matrix{Int64}:
 1  3  2  4  5
 1  3  2  4  5
 1  3  2  4  5
 1  3  2  4  5
 1  3  2  4  5
 1  3  2  4  5
 3  1  2  4  5
 1  3  2  4  5
```

And summarise ranks per algorithm

```
julia> summarize_ranks(ranks_m1, Algs)
5×8 DataFrame
 Row │ Algorithm                          MedianRank  MeanRank  Std      Q2_5     Q97_5    Q25      Q75
     │ String                             Float64     Float64   Float64  Float64  Float64  Float64  Float64
─────┼──────────────────────────────────────────────────────────────────────────────────────────────────────
   1 │ adaptive_de_rand_1_bin_radiuslim…         1.0      1.49     0.86      1.0      3.0      1.0      1.0
   2 │ generating_set_search                     2.0      2.11     0.45      2.0      4.0      2.0      2.0
   3 │ de_rand_2_bin                             3.0      2.51     0.86      1.0      3.0      3.0      3.0
   4 │ probabilistic_descent                     4.0      3.89     0.45      2.0      4.0      4.0      4.0
   5 │ random_search                             5.0      5.0      0.0       5.0      5.0      5.0      5.0
```

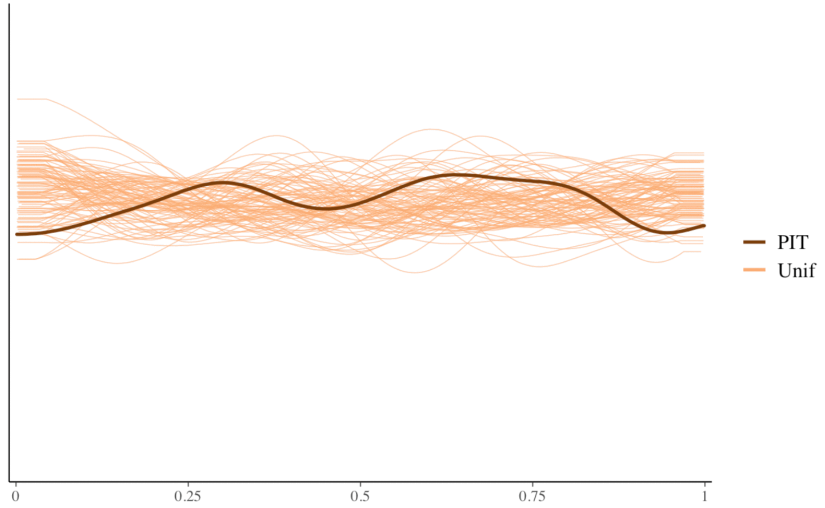
Use posterior again: Calc pair-wise win probabilities

```
julia> dfpred[dfpred.Prob .>= 0.50, :]
10×6 DataFrame
```

Row	Alg2 String	Alg1 String	A2 Int64	A1 Int64	Prob Float64	Std Float64
1	adaptive_de_rand_1_bin_radiuslim…	random_search	1	5	1.0	0.04
2	generating_set_search	random_search	3	5	1.0	0.04
3	de_rand_2_bin	random_search	2	5	0.99	0.1
4	probabilistic_descent	random_search	4	5	0.98	0.15
5	adaptive_de_rand_1_bin_radiuslim…	probabilistic_descent	1	4	0.94	0.24
6	generating_set_search	probabilistic_descent	3	4	0.92	0.27
7	adaptive_de_rand_1_bin_radiuslim…	de_rand_2_bin	1	2	0.89	0.31
8	generating_set_search	de_rand_2_bin	3	2	0.86	0.35
9	de_rand_2_bin	probabilistic_descent	2	4	0.66	0.47
10	adaptive_de_rand_1_bin_radiuslim…	generating_set_search	1	3	0.57	0.49

Model comparisons

- Traditionally used WAIC/BIC/DIC etc.
- An information theoretical relative comparison of >1 models
- PSIS-LOO is state of art
  - Handles all type of likelihoods
  - Compared to WAIC,
    - LOO provides ample diagnostics giving you confidence in results



Aki Vehtari,
Aalto University

Danielle Navarro,
U. New South Wales

Our second model: Add “change” based on problem dimension

```
@model function m2(alg1, alg2, logDim, twowins)
    # Hyperparams
    sigma_alg ~ Exponential(1.0)

    # Logits per algorithm
    alg ~ filldist(Normal(0, sigma_alg), N_alg)

    # log(Dim) coefficients per algorithm
    b_alg ~ filldist(Normal(0, 2), N_alg)

    for i in 1:length(twowins)
        a2, a1, logD = alg2[i], alg1[i], logDim[i]
        # Since we model if alg 2 will "win" that should be the positive logit
        v = logistic(alg[a2] + b_alg[a2]*logD - alg[a1] - b_alg[a1]*logD)
        twowins[i] ~ Bernoulli(v)
    end
end;
```

Results are similar for model 1 and 2:

```
julia> ranks_m2 = summarize_ranks_from_posterior_sample(posterior_m2, logDims)
5×8 DataFrame
```

Row	Algorithm String	MedianRank Float64	MeanRank Float64	Std Float64	Q2_5 Float64	Q97_5 Float64	Q25 Float64	Q75 Float64
1	adaptive_de_rand_1_bin_radiuslim…	1.0	1.53	0.88	1.0	3.0	1.0	3.0
2	generating_set_search	2.0	2.69	0.92	2.0	4.0	2.0	4.0
3	de_rand_2_bin	3.0	2.41	0.88	1.0	3.0	1.0	3.0
4	probabilistic_descent	4.0	3.37	0.93	2.0	4.0	2.0	4.0
5	random_search	5.0	5.0	0.01	5.0	5.0	5.0	5.0

```
julia> summarize_ranks(ranks_m1, Algs)
5×8 DataFrame
```

Row	Algorithm String	MedianRank Float64	MeanRank Float64	Std Float64	Q2_5 Float64	Q97_5 Float64	Q25 Float64	Q75 Float64
1	adaptive_de_rand_1_bin_radiuslim…	1.0	1.49	0.86	1.0	3.0	1.0	1.0
2	generating_set_search	2.0	2.11	0.45	2.0	4.0	2.0	2.0
3	de_rand_2_bin	3.0	2.51	0.86	1.0	3.0	3.0	3.0
4	probabilistic_descent	4.0	3.89	0.45	2.0	4.0	4.0	4.0
5	random_search	5.0	5.0	0.0	5.0	5.0	5.0	5.0

Model comparison: Model 2 is preferred

```
julia> models = (
                m1=m1_psis_loo,
                m2=m2_psis_loo,
       );

julia> comps = loo_compare(models)
```

	cv_elpd	cv_avg	weight
m2	0.00	0.00	0.99
m1	-4.27	-0.02	0.01

But we can also check different Dim ranges

```
julia> ranks_m2_lowdim = summarize_ranks_from_posterior_sample(posterior_m2, log10.(collect(2:1:20)))
5×8 DataFrame
 Row │ Algorithm                           MedianRank  MeanRank  Std      Q2_5     Q97_5    Q25      Q75
     │ String                              Float64     Float64   Float64  Float64  Float64  Float64  Float64
─────┼──────────────────────────────────────────────────────────────────────────────────────────────────────
   1 │ adaptive_de_rand_1_bin_radiuslim…          1.0      1.29     0.65      1.0      3.0      1.0      1.0
   2 │ generating_set_search                      2.0      2.2      0.48      2.0      3.0      2.0      2.0
   3 │ de_rand_2_bin                              3.0      2.54     0.77      1.0      3.0      2.0      3.0
   4 │ probabilistic_descent                      4.0      3.98     0.26      4.0      4.0      4.0      4.0
   5 │ random_search                              5.0      4.99     0.1       5.0      5.0      5.0      5.0

julia> ranks_m2_highdim = summarize_ranks_from_posterior_sample(posterior_m2, log10.(collect(150:10:200)))
5×8 DataFrame
 Row │ Algorithm                           MedianRank  MeanRank  Std      Q2_5     Q97_5    Q25      Q75
     │ String                              Float64     Float64   Float64  Float64  Float64  Float64  Float64
─────┼──────────────────────────────────────────────────────────────────────────────────────────────────────
   1 │ de_rand_2_bin                              1.0      1.97     1.0       1.0      3.0      1.0      3.0
   2 │ probabilistic_descent                      2.0      2.36     0.77      2.0      4.0      2.0      2.0
   3 │ adaptive_de_rand_1_bin_radiuslim…          3.0      2.03     1.0       1.0      3.0      1.0      3.0
   4 │ generating_set_search                      4.0      3.64     0.77      2.0      4.0      4.0      4.0
   5 │ random_search                              5.0      5.0      0.0       5.0      5.0      5.0      5.0
```

adaptive_de and generating_set seems worse for high-dimensional problems but note variance is higher so need more data

Or check pairs of algs for different Dims

```
julia> # random_search so bad so let's skip it and show rest only if >=50%
       dfpred[dfpred.Prob .>= 0.50 .&& dfpred.Alg1 .!== "random_search", :]
12×7 DataFrame
```

Row	Alg2 String	Alg1 String	A2 Int64	A1 Int64	Dim Int64	Prob Float64	Std Float64
1	generating_set_search	de_rand_2_bin	3	2	200	0.98	0.13
2	adaptive_de_rand_1_bin_radiuslim…	de_rand_2_bin	1	2	200	0.98	0.13
3	generating_set_search	probabilistic_descent	3	4	200	0.95	0.22
4	adaptive_de_rand_1_bin_radiuslim…	probabilistic_descent	1	4	200	0.95	0.22
5	adaptive_de_rand_1_bin_radiuslim…	probabilistic_descent	1	4	10	0.93	0.25
6	generating_set_search	probabilistic_descent	3	4	10	0.9	0.3
7	de_rand_2_bin	probabilistic_descent	2	4	10	0.8	0.4
8	adaptive_de_rand_1_bin_radiuslim…	de_rand_2_bin	1	2	10	0.79	0.41
9	generating_set_search	de_rand_2_bin	3	2	10	0.69	0.46
10	probabilistic_descent	de_rand_2_bin	4	2	200	0.69	0.46
11	adaptive_de_rand_1_bin_radiuslim…	generating_set_search	1	3	10	0.62	0.48
12	generating_set_search	adaptive_de_rand_1_bin_radiuslim…	3	1	200	0.51	0.5

generating_set seems relatively better for high-dimensional problems

Include more runs => variance (std) decreases

```
julia> dfpred[dfpred.Prob .>= 0.50 .&& dfpred.Alg1 .!== "random_search", :]
12×7 DataFrame
```

Row	Alg2 String	Alg1 String	A2 Int64	A1 Int64	Dim Int64	Prob Float64	Std Float64
1	adaptive_de_rand_1_bin_radiuslim…	de_rand_2_bin	1	2	200	0.99	0.11
2	generating_set_search	de_rand_2_bin	3	2	200	0.99	0.11
3	adaptive_de_rand_1_bin_radiuslim…	probabilistic_descent	1	4	10	0.95	0.22
4	adaptive_de_rand_1_bin_radiuslim…	probabilistic_descent	1	4	200	0.95	0.21
5	generating_set_search	probabilistic_descent	3	4	200	0.95	0.23
6	de_rand_2_bin	probabilistic_descent	2	4	10	0.9	0.3
7	generating_set_search	probabilistic_descent	3	4	10	0.88	0.33
8	probabilistic_descent	de_rand_2_bin	4	2	200	0.83	0.37
9	adaptive_de_rand_1_bin_radiuslim…	generating_set_search	1	3	10	0.72	0.45
10	adaptive_de_rand_1_bin_radiuslim…	de_rand_2_bin	1	2	10	0.69	0.46
11	de_rand_2_bin	generating_set_search	2	3	10	0.54	0.5
12	adaptive_de_rand_1_bin_radiuslim…	generating_set_search	1	3	200	0.52	0.5

But not much variance reduction => more detailed models might pay off, more data not likely needed
(Of course more data gives more “true” means etc, point is that variance is high in actual data so more data unlikely to reduce it

Part 3: Now what!? Bigger picture

of Bayesian Data Analysis in SE?

Practical significance: what it really means?

- Posterior distribution includes the uncertainty
- Simulate from posterior and answer practical questions of domain
- For Language to project effort models:
  - Which language should we choose in this new project?
  - Is it cost-effective to switch language given the project costs we have?
- Typically need to add cost-related information
  - But engineers/practitioners often good at “ballpark” estimates

A Method to Assess and Argue for Practical Significance in Software Engineering

Richard Torkar, Carlo A. Furia, Robert Feldt, Francisco Gomes de Oliveira Neto, Lucas Gren, Per Lenberg, and Neil A. Ernst

Abstract—A key goal of empirical research in software engineering is to assess practical significance, which answers whether the observed effects of some compared treatments show a relevant difference in practice in realistic scenarios. Even though plenty of standard techniques exist to assess statistical significance, connecting it to practical significance is not straightforward or routinely done; indeed, only a few empirical studies in software engineering assess practical significance in a principled and systematic way.
In this paper, we argue that Bayesian data analysis provides suitable tools to assess practical significance rigorously. We demonstrate our claims in a case study comparing different test techniques. The case study's data was previously analyzed (Afzal et al., 2015) using standard techniques focusing on statistical significance. Here, we build a multilevel model of the same data, which we fit and validate using Bayesian techniques. Our method is to apply cumulative prospect theory on top of the statistical model to quantitatively connect our statistical analysis output to a practically meaningful context. This is then the basis both for assessing and arguing for practical significance.

arxiv.org/pdf/1809.09849.pdf

TABLE 3
Expected number of faults detected for different combinations of predictors in $\mathcal{M}_2$. Each row reports the range of *faults* corresponding to 94% probability and the mean on the posterior.

developers	fixed predictors	94% CI	mean
low experience	*experience* = 0	1, 8	4.02
high experience	*experience* = 1	1, 11	5.67
exploratory testing	*approach* = 0	6, 11	8.27
test-case testing	*approach* = 1	1, 2	1.42
exploratory and low	*approach* = 0 *experience* = 0	6, 8	6.92
exploratory and high	*approach* = 0 *experience* = 1	8, 12	9.61

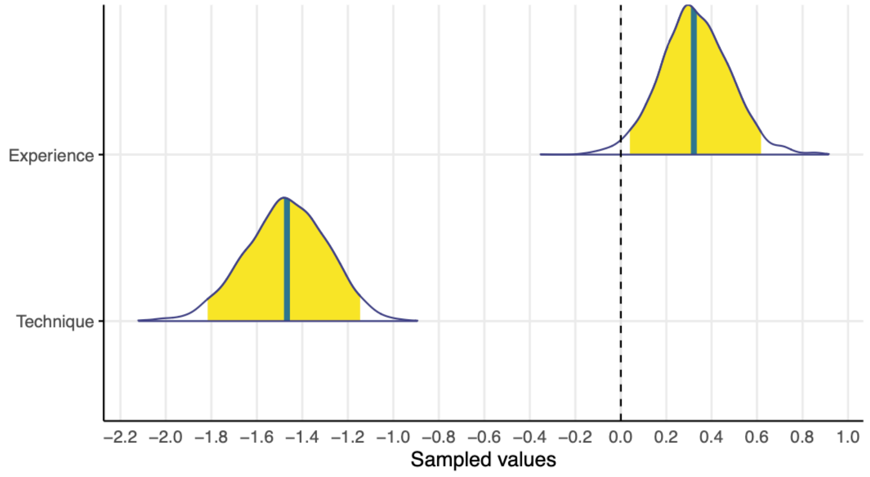
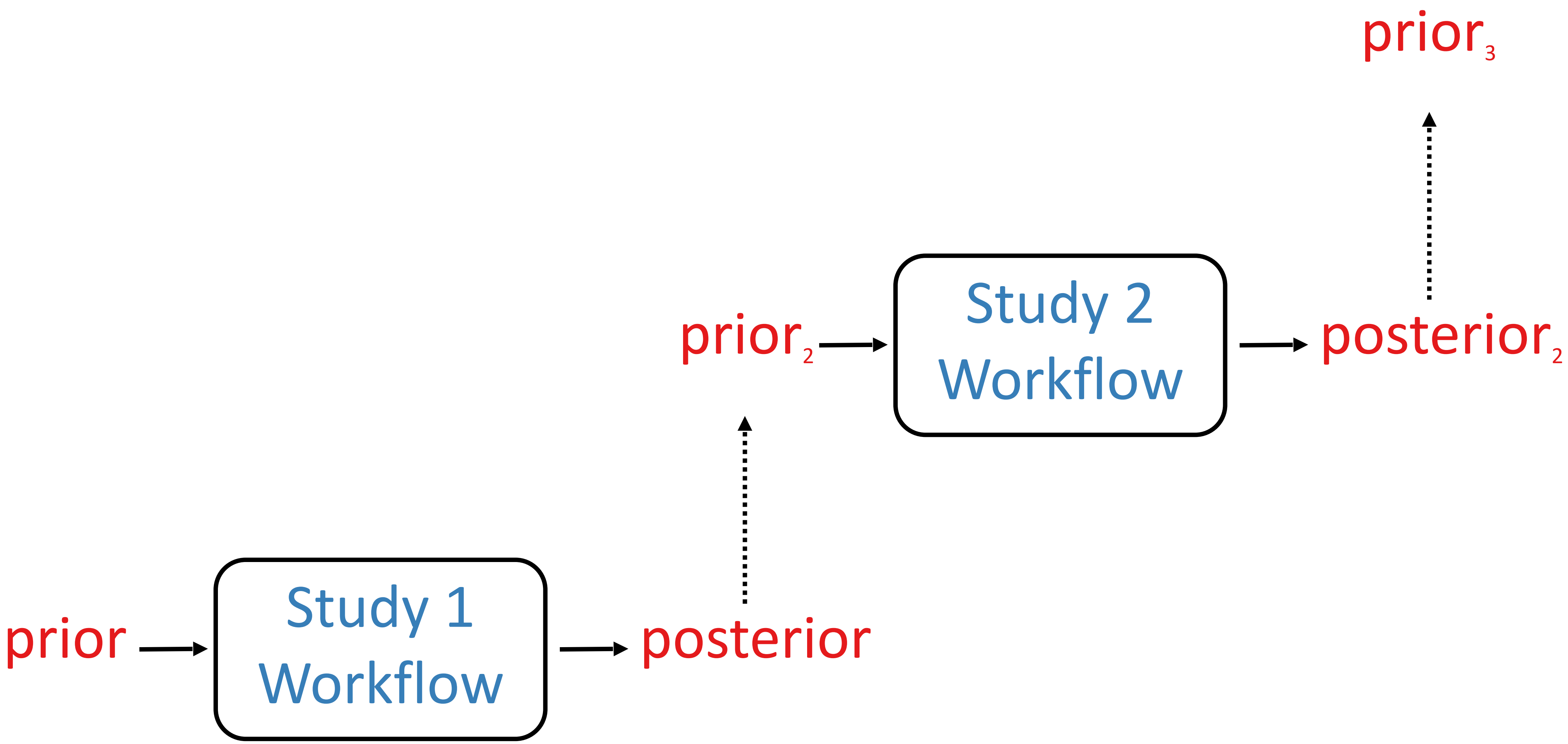


Fig. 3. Posterior marginal probability distributions of β_e (*experience*, top) and β_a (*approach*, bottom named 'Technique'). The thick lines mark the medians, and the yellow areas cover 94% of probability.

Simulations + Estimated costs => Practical implications & significance

Building SE chains of evidence



Links

- https://github.com/robertfeldt/robertfeldt.github.io/blob/master/research/bayesian_se/index.md
- https://github.com/robertfeldt/robertfeldt.github.io/blob/master/research/bayesian_se/replication_packages_by_model_type.md
- https://github.com/torkar/icse_tutorial (try it yourself)
- http://tiny.cc/bayes-icse21 for more polished videos of what we've talked about!

Do you want to know more?

- McElreath, R. (2020). *Statistical Rethinking: A Bayesian Course with Examples in R and Stan, 2nd Edition*, CRC Press
- Jaynes, E. (2003). *Probability Theory: The Logic of Science* (G. Bretthorst, Ed.). Cambridge: Cambridge University Press
- Navarro, D. J. (2019). Between the devil and the deep blue sea: Tensions between scientific judgement and statistical model selection. *Computational Brain and Behavior,* 2:28–34
- Frank, S. A. (2009), The common patterns of nature. *Journal of Evolutionary Biology*, 22:1563–1585

- https://mc-stan.org (`Stan` is state of practice concerning HMC sampling)
- https://mc-stan.org/users/interfaces/brms (`lme4` syntax for models)
- https://mc-stan.org/bayesplot/ (plots galore for Bayesian MLMs)
- https://mc-stan.org/users/interfaces/loo (model comparison)
- https://turing.ml/ (model design in Julia, very flexible and powerful but less mature than stan/R)

License of these slides