

a Pelican Book

How to Lie with Statistics

Darrell Huff



“Statistical thinking
will one day be as
necessary for
efficient citizenship
as the ability to
read and write.”
H.G. Wells

Empowering Spreadsheet Users with Probabilistic Programming

Andrew D. Gordon
(MSR and Edinburgh)

#ProbProgSchool2017, Minho

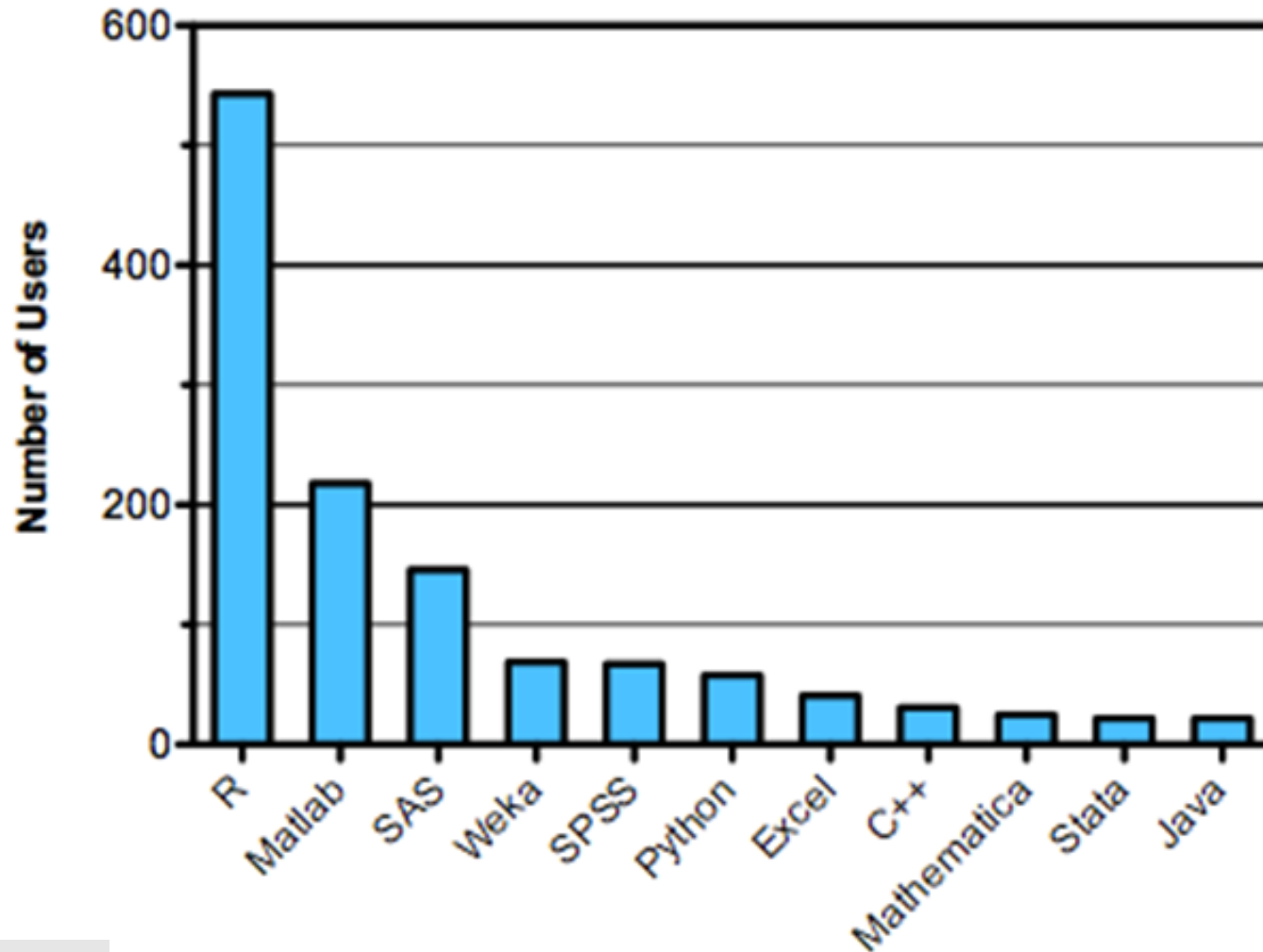
Based on joint work with Johannes Borgström (Uppsala), Thore Graepel (MSR),
Long Ouyang (Stanford), Nicolas Rolland (MSR), Claudio Russo (MSR),
Adam Scibior (Cambridge), Marcin Szymczak (Edinburgh), and Danny Tarlow (MSR)

<https://aka.ms/tabular>

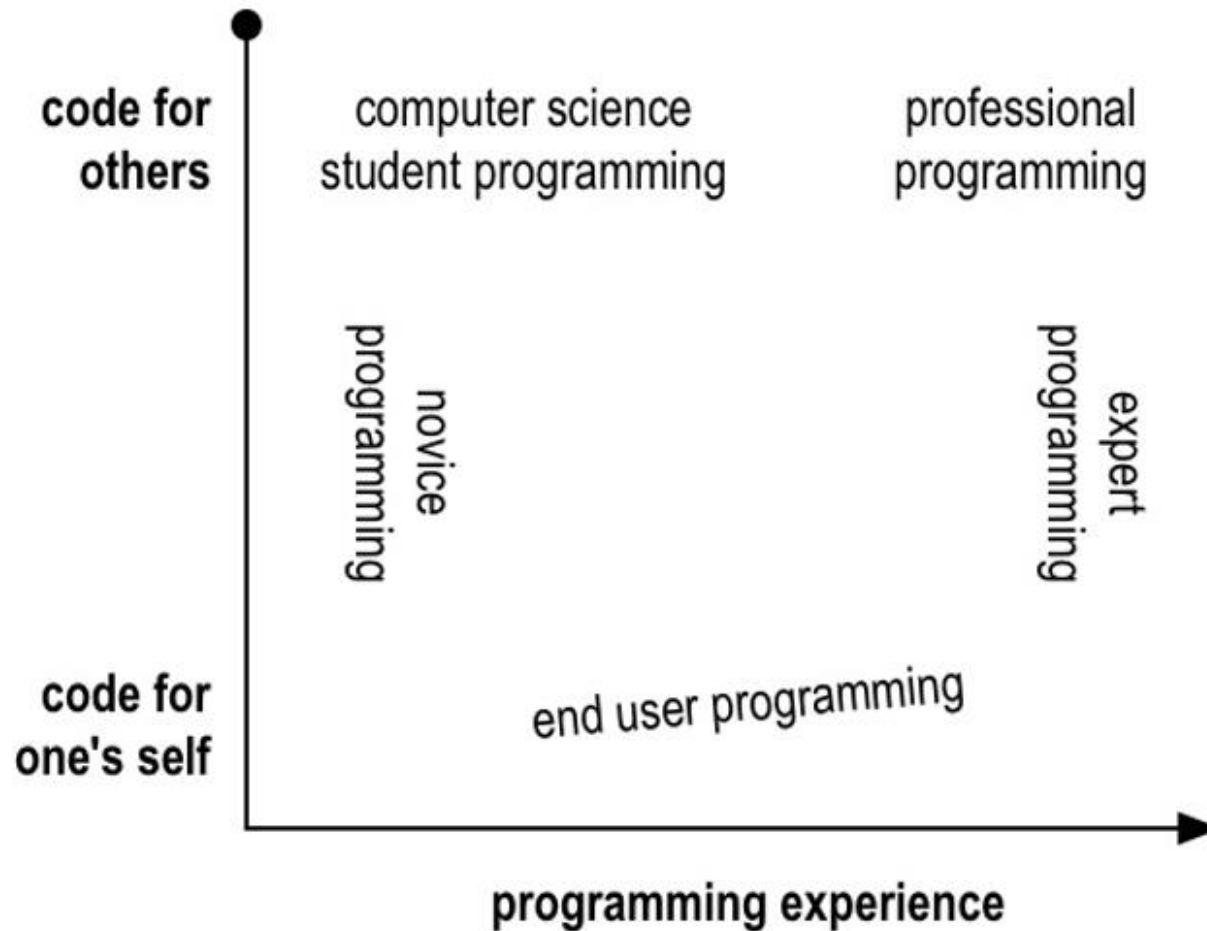
Probabilistic Programming

- Start with your favourite paradigm (fun, logic, imp), add **random** to get probabilistic behaviour, add **constraints** to condition on observed data, and indicate which variables' distributions to be **inferred**.
- Better than writing code for probabilistic inference from scratch.
- Many academic and commercial systems: BUGS, Figaro, pcc, Church, Dimple, Hansei, STAN, **Infer.NET/Fun**, R2, Factorie, BLOG, ProbLog, Alchemy, Venture, Anglican, Wolfe, Hakaru, Edward,
- Academic prizes (eg in computer vision), and large-scale adoption in some commercial services (eg Microsoft Office Clutter, etc)
- Semantics pioneered by Kozen, Giry, Jones/Plotkin, Panangaden et al, Ramsey/Pfeffer

Data Scientists?



End-User Programmers?



Quora

Ask or Search

Microsoft Excel Spreadsheets

What do you need to know to be considered "advanced" in excel?

1. Beginner Level:

- | | |
|--------------------------|-------------------------|
| 1. Basic Math Operators | 6. Date Functions |
| 2. SUM Function | 7. LEN Function |
| 3. COUNT Function | 8. COUNTA Function |
| 4. AVERAGE Function | 9. ROUND Function |
| 5. MAX and MIN Functions | 10. SUMPRODUCT Function |

2. Medium Level:

- | | |
|---------------------|-----------------------------|
| 1. VLOOKUP Function | 6. CONCATENATE Function |
| 2. IF Function | 7. HLOOKUP Function |
| 3. COUNTIF Function | 8. RIGHT and LEFT Functions |
| 4. SUMIF Function | 9. RAND Function |
| 5. IS Functions | 10. IFERROR Function |

3. Advanced Level:

- | | |
|------------------------------|----------------------------|
| 1. Advanced VLOOKUP Function | 6. INDEX Function |
| 2. IF – AND Combinations | 7. MID and SEARCH Function |
| 3. IF – OR Combinations | 8. OFFSET Function |
| 4. Nested IFs | 9. INDIRECT Function |
| 5. MATCH Function | 10. Array Formulas |

Traces of Users

TABLE I
AN OVERVIEW THE SPREADSHEETS IN THE ENRON AND EUSES SET

| | EUSES | Enron |
|---|-----------|------------|
| Number of spreadsheets analyzed | 4,447 | 15,770 |
| Number of spreadsheets with formulas | 1,961 | 9,120 |
| Number of worksheets | 16,853 | 79,983 |
| Maximum number of worksheets | 106 | 175 |
| Number of non-empty cells | 8,209,095 | 97,636,511 |
| Average number of non-empty cells per spreadsheet | 1,846 | 6,191 |
| Number of formulas | 730,186 | 20,277,835 |
| Average of formulas per spreadsheet with formulas | 372 | 2,223 |
| Number of unique formulas | 65,143 | 913,472 |
| Number of unique formulas per spreadsheet with formulas | 33 | 100 |

| | EUSES | Enron |
|--------------------------|-------|-------|
| Spreadsheets | 4447 | 15770 |
| With formulas | 1961 | 9120 |
| Proportion with formulas | 44% | 58% |

Felienne Hermans and Emerson Murphy-Hill: Enron's Spreadsheets and Related Emails: A Dataset and Analysis. ICSE-SEIP (2015)

TABLE V
LIST OF MOST USED FUNCTIONS, THE NUMBER OF SPREADSHEETS IN WHICH THEY OCCUR AND CORRESPONDING PERCENTAGES

| Rank | Functions | # Spreadsheets | Percentage |
|------|-----------|----------------|------------|
| 1 | SUM | 6493 | 72.0% |
| 2 | + | 5571 | 61.8% |
| 3 | - | 4866 | 54.0% |
| 4 | * | 3527 | 39.1% |
| 5 | / | 3112 | 34.5% |
| 6 | IF | 1827 | 20.3% |
| 7 | NOW | 1501 | 16.7% |
| 8 | AVERAGE | 879 | 9.8% |
| 9 | VLOOKUP | 763 | 8.5% |
| 10 | ROUND | 606 | 6.7% |
| 11 | TODAY | 537 | 6.0% |
| 12 | SUBTOTAL | 385 | 4.3% |
| 13 | MONTH | 325 | 3.6% |
| 14 | CELL | 321 | 3.6% |
| 15 | YEAR | 287 | 3.2% |
| | Any above | 8961 | 99.4% |

of functions.

TABLE VI
THE MOST USED FUNCTIONS, THE NUMBER OF SPREADSHEETS IN WHICH ONLY THEY OCCUR AND CORRESPONDING PERCENTAGES, LISTED CUMULATIVELY

| Rank | Functions | # Spreadsheets | Percentage |
|------|-----------|----------------|------------|
| 1 | SUM | 578 | 6.4% |
| 2 | + | 1259 | 14.0% |
| 3 | - | 2262 | 25.1% |
| 4 | / | 2625 | 29.1% |
| 5 | * | 3959 | 43.9% |
| 6 | IF | 4260 | 47.3% |
| 7 | NOW | 5322 | 59.1% |
| 8 | AVERAGE | 5664 | 62.8% |
| 9 | VLOOKUP | 5733 | 63.6% |
| 10 | ROUND | 5990 | 66.5% |
| 11 | TODAY | 6182 | 68.6% |
| 12 | SUBTOTAL | 6480 | 71.9% |
| 13 | MONTH | 6520 | 72.3% |
| 14 | CELL | 6774 | 75.2% |
| 15 | YEAR | 6812 | 75.6% |

Probabilistic Programming already Empowers (Some) Spreadsheet Users!

| | A | B | C |
|---|---------|--------------|--------------|
| 1 | | <i>Best</i> | <i>Worst</i> |
| 2 | Widgets | 75000 | 125000 |
| 3 | Cost | \$6.50 | \$7.50 |
| 4 | Total | \$487,500.00 | \$937,500.00 |
| 5 | Budget | \$750,000.00 | \$750,000.00 |
| 6 | Over? | FALSE | TRUE |
| 7 | | | |

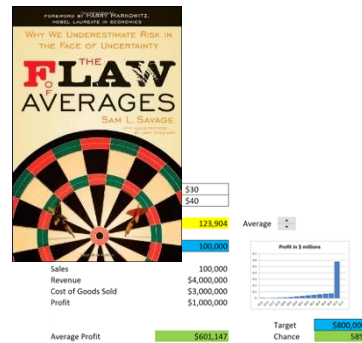
Best **Worst** (+)

Scenario Manager

Scenarios:

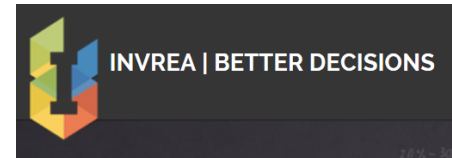
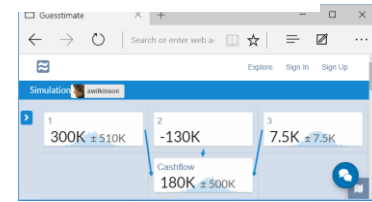
Best

Worst



ORACLE®
CRYSTAL BALL

BETA
Guesstimate



Gartner.

WHY GARTNER ANALYSTS RESEARCH EVENTS CONSULTING AB

Hype Cycle for Data Science, 2016

🕒 25 July 2016 📄 G00303293

On the Rise

Probability Management

Cost per Unit
Selling Price per Unit

| |
|------|
| \$30 |
| \$40 |

Actual Demand (Uncertain)

100,000

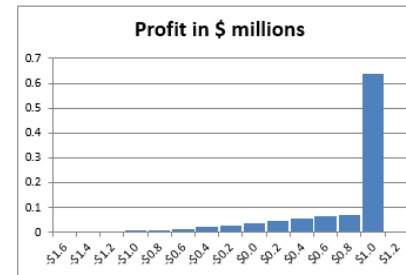
Average



Decision: Quantity Ordered

95,000

Sales 95,000
Revenue \$3,800,000
Cost of Goods Sold \$2,850,000
Profit \$950,000



Average Profit

\$643,195

Target
Chance

\$800,000

64%

© Copyright 2013, ProbabilityManagement.org

Demo: Flaw of Averages

Source: <http://probabilitymanagement.org/>

Empowering Spreadsheet Users

- Modern spreadsheets incorporate end-user database functionality previously in separate applications
- “A **data enthusiast** is an educated person who believes that data can be used to answer a question or solve a problem. These people are not mathematicians or programmers, and only know a bit of statistics.” (Pat Hanrahan 2012)
- **Goal:** empower data enthusiasts, not professional programmers or data scientists
- Existing spreadsheet tools for probabilistic models
 - accessible to highly advanced spreadsheet users,
 - but out of reach of most data enthusiasts

Example: InfernoDB

Automatic Machine Learning for Relational DBs

Sameer Singh

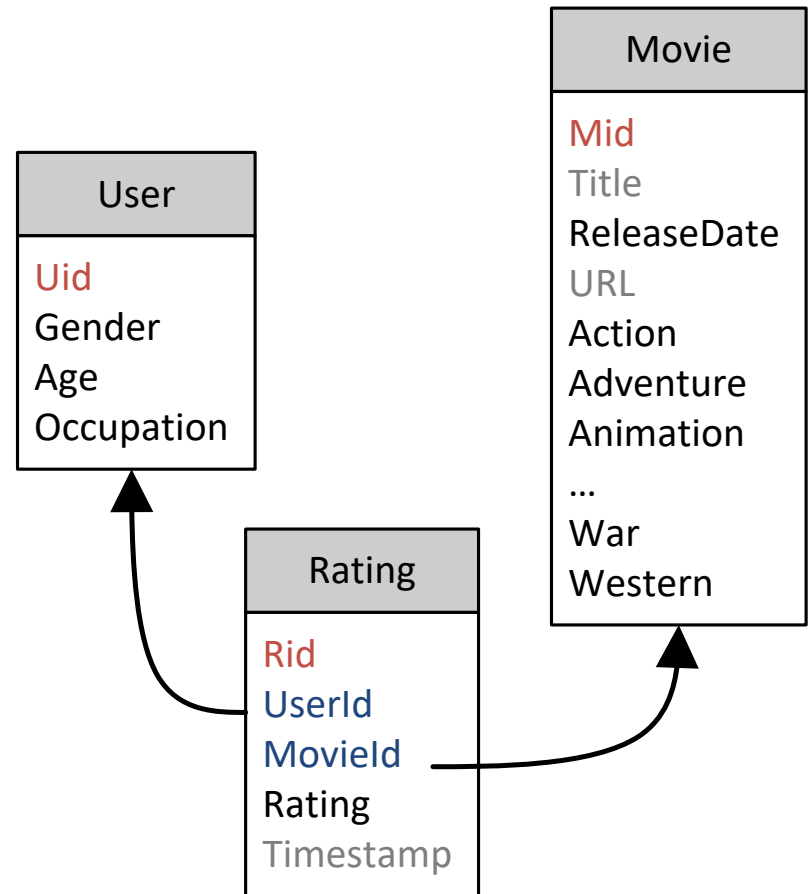
(with Thore Graepel)

Automated Machine Learning

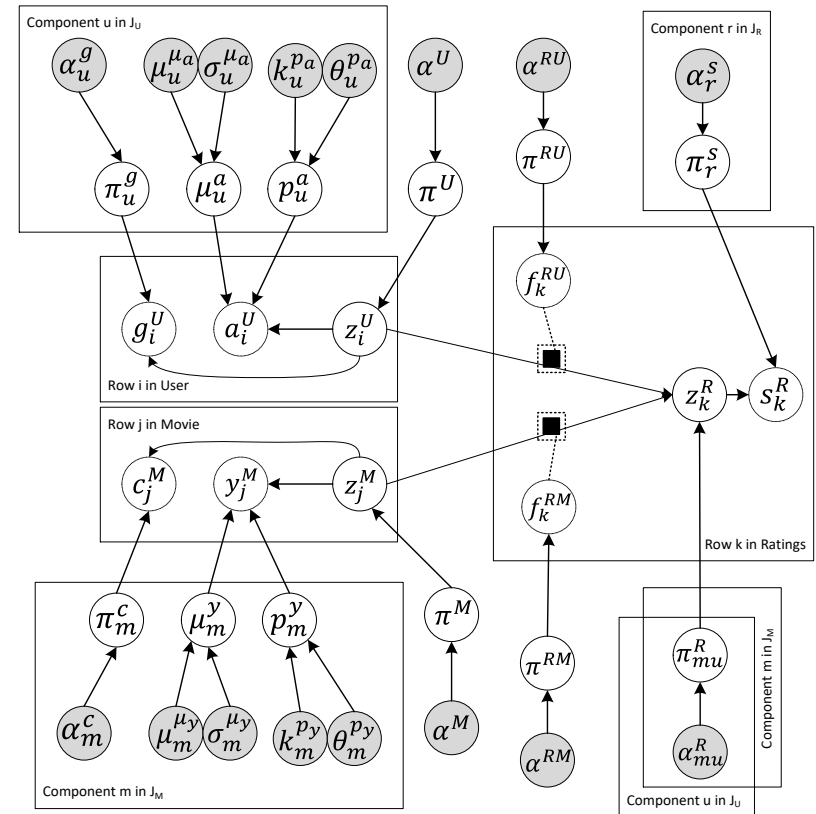
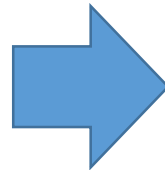
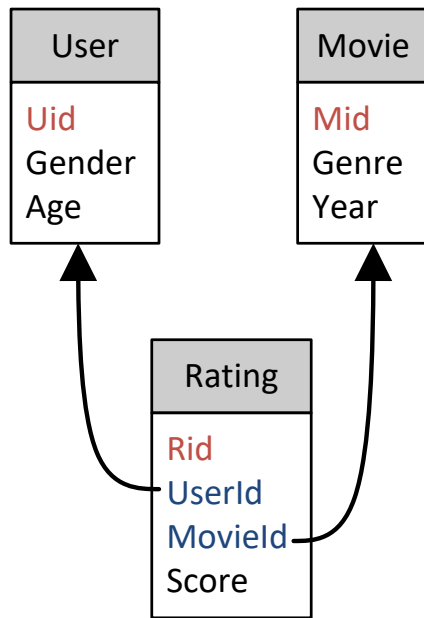
- Abstractions: Classification, Clustering
 - ML experts improve methods
 - Data scientists translate problem into the abstraction
- Not so with Probabilistic Programming
 - Not an easy abstraction for the data enthusiast
- **Objective:** Use probabilistic programming
 - Without data enthusiast needing to write code
 - Without ML expert learning a new domain
- Idea: derive probabilistic program from schema
 - Leverage data enthusiast's knowledge
 - Lot of effort spent in designing databases

Example: Recommendations

- Predict Age, Gender, Occupation given the ratings
- Predict Movie genres given user details
- Predict ratings for a new movie given its genres



Automatic Model Suggestion



The InfernoDB demo is a UI showing each table.

Colors on the left show the inferred cluster for each node.

The pink cells were missing in the raw database, and have been automatically filled in by inference.

ication.EnableVisualStyle() isual styles for the application.

InfernoDB Results

Inference

Movie User Rating

Rating

Display Data

☒ Show All Data

☐ Show Missing Data

☐ Show Query Data

Size: 4 x 29

Components: <comps>

☒ Show Predictions

Save Query

Delete Row

(2, 15) : foreign key to Movie

0 Silent 1920

0:xxxxxx 1: 2: 3: 4: 5: 6:

7: 8: 9: 10: 11: 12:

13: 14: 15: 16: 17: 18:

| | RatingId | UserId | MovieId | Score |
|----|----------|--------|---------|-------|
| 0 | 0 | 0 | 0 | 10 |
| 1 | 0 | 0 | 1 | 9 |
| 2 | 0 | 0 | 2 | 0 |
| 3 | 0 | 0 | 3 | 1 |
| 4 | 0 | 0 | 4 | 2 |
| 5 | 1 | 0 | 0 | 8 |
| 6 | 1 | 1 | 1 | 10 |
| 7 | 1 | 1 | 2 | 3 |
| 8 | 1 | 1 | 3 | 1 |
| 9 | 1 | 1 | 4 | 0 |
| 10 | 2 | 0 | 0 | 5 |
| 11 | 2 | 1 | 1 | 4 |
| 12 | 2 | 2 | 2 | 7 |
| 13 | 2 | 3 | 3 | 8 |
| 14 | 2 | 4 | 4 | 9 |
| 15 | 3 | 0 | 0 | 1 |
| 16 | 3 | 1 | 1 | 3 |
| 17 | 3 | 2 | 2 | 10 |
| 18 | 3 | 3 | 3 | 10 |
| 19 | 3 | 4 | 4 | 9 |
| 20 | 4 | 0 | 0 | 8 |
| 21 | 4 | 1 | 1 | 8 |
| 22 | 4 | 2 | 2 | 2 |
| 23 | 4 | 3 | 3 | 2 |
| 24 | 4 | 4 | 4 | 2 |
| 25 | 4 | 0 | 0 | 7 |
| 26 | 4 | 4 | 4 | 2 |
| 27 | 3 | 4 | 4 | 9 |
| 28 | 4 | 4 | 4 | 1 |

A simple example

(slides in this section adapted from a talk by Tom Minka)

A simple example

I secretly toss two coins



I tell you that they are NOT both heads.

What is the probability that the first coin is heads?

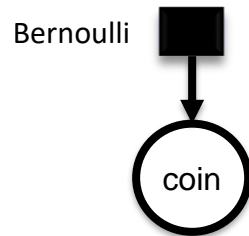
How do we specify the model?

- Probability distribution
 - most flexible, dense, opaque
- Factor graph
 - less flexible, clear (for small graphs)
- Simulator
 - more flexible, clearer for large graphs

Model for a fair coin

$$p(\textit{coin}) = 0.5^{\textit{coin}} 0.5^{1-\textit{coin}}$$

(Bernoulli distribution)



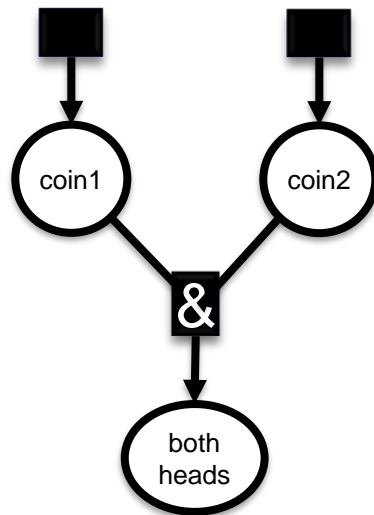
(Factor graph)

```
bool coin = random() < 0.5;
```

(Simulator)

Model for two coins problem

$$p(\text{coin}_1)p(\text{coin}_2)p(\text{bothHeads}|\text{coin}_1, \text{coin}_2)$$



```
bool coin1 = random()<0.5;  
bool coin2 = random()<0.5;  
bool bothHeads = coin1 & coin2;
```

Bayesian inference

- Mathematical approach
 - solving integrals
- Simulation approach
 - rejection sampling
- Message-passing approach
 - local operations on factor graph



Simulation approach

```
bool coin1 =  
    random() < 0.5;
```



```
bool coin2 =  
    random() < 0.5;
```



```
bool bothHeads =  
    coin1 & coin2;
```



After many, many runs...

```
bool coin1 =  
    random() < 0.5;
```

 ~50%
 ~50%

```
bool coin2 =  
    random() < 0.5;
```

 ~50%
 ~50%

```
bool bothHeads =  
    coin1 & coin2;
```

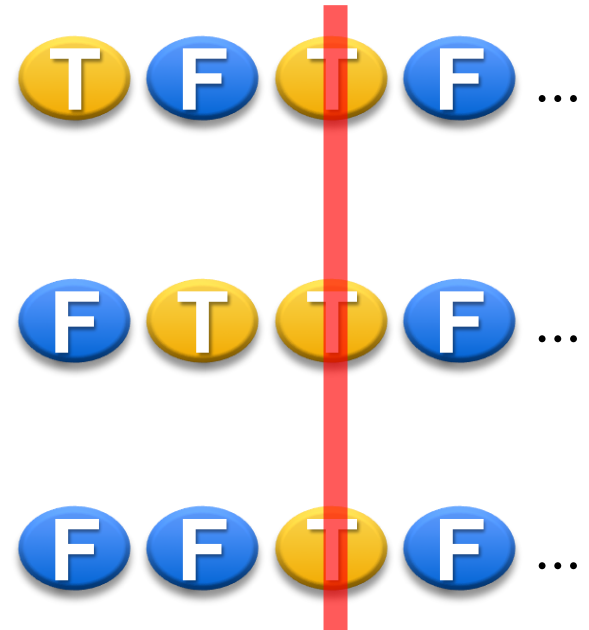
 ~25%
 ~75%

Attaching data to the model

```
bool coin1 =  
    random() < 0.5;
```

```
bool coin2 =  
    random() < 0.5;
```

```
bool bothHeads =  
    coin1 & coin2;
```



We *observe* that bothHeads is **F**

Throw away runs that don't match...

```
bool coin1 =  
    random() < 0.5;
```

T ~33%
F ~67%

```
bool coin2 =  
    random() < 0.5;
```

T ~33%
F ~67%

```
bool bothHeads =  
    coin1 & coin2;
```

T ~0%
F ~100%

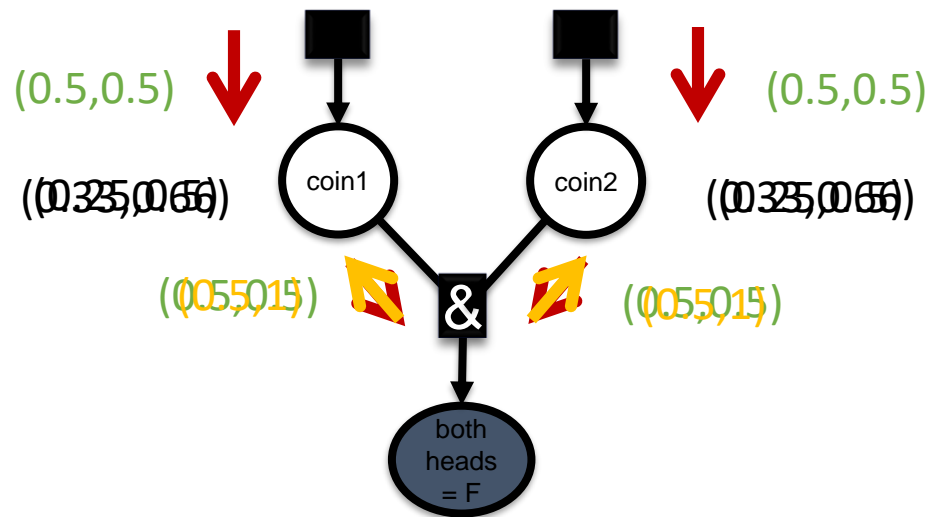
Bayesian inference

- Mathematical approach
 - solving integrals
- Simulation approach
 - rejection sampling
- Message-passing approach
 - local operations on factor graph



Message passing

(T,F)



Belief propagation
(Pearl, 1982)

Efficiency

- Want to reproduce known efficient algorithms for machine learning
 - Message passing in factor graphs
- Motivates using a compiler
- Goal of compiler is to output same code that person would write for solving that model

Infer.NET
an inference engine

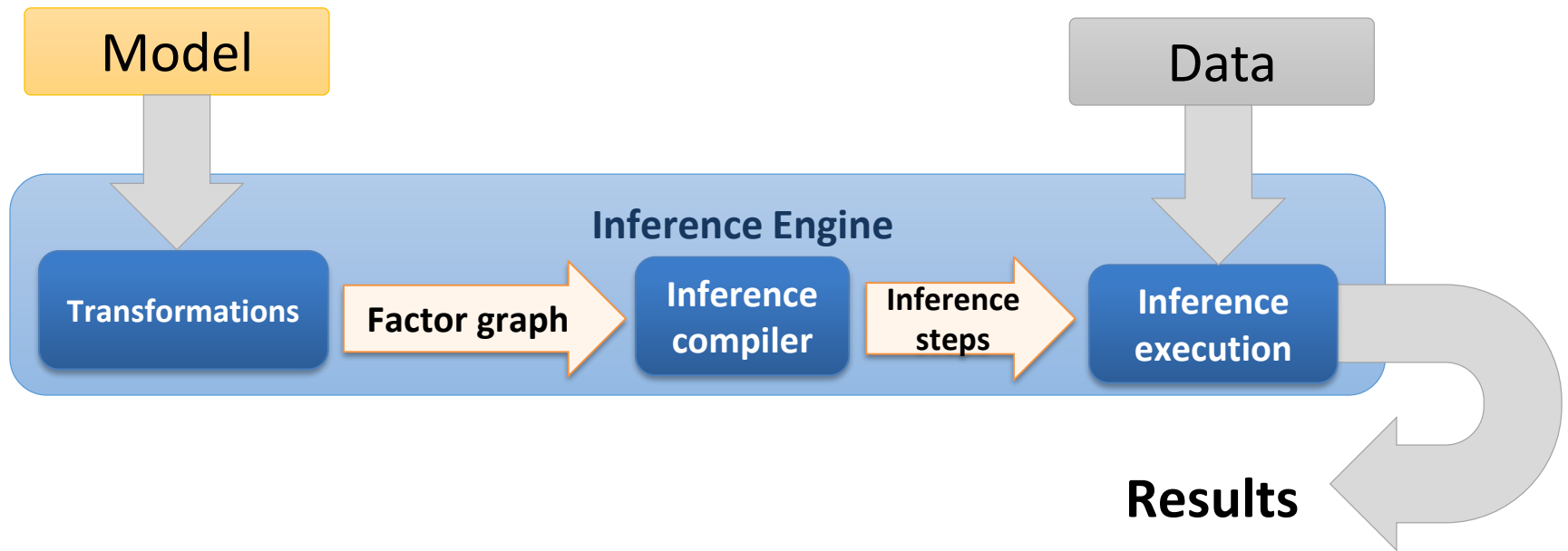


<http://infernet.azurewebsites.net/>

Established 2004

Tom Minka, John Winn, John Guiver, and many others

How Infer.NET works



Coins model in Infer.NET

```
var coin1 =  
    Variable.Bernoulli(0.5); // (random() < 0.5)
```

```
var coin2 =  
    Variable.Bernoulli(0.5);
```

```
var bothHeads =  
    coin1 & coin2;
```


Running inference in the model

```
var engine = new InferenceEngine();  
  
Bernoulli result =  
    engine.Infer<Bernoulli>(bothHeads);  
  
// 'result' is Bernoulli(0.25)
```

Attaching data to the model

We observe that bothHeads is 

```
bothHeads.ObservedValue = false;
```

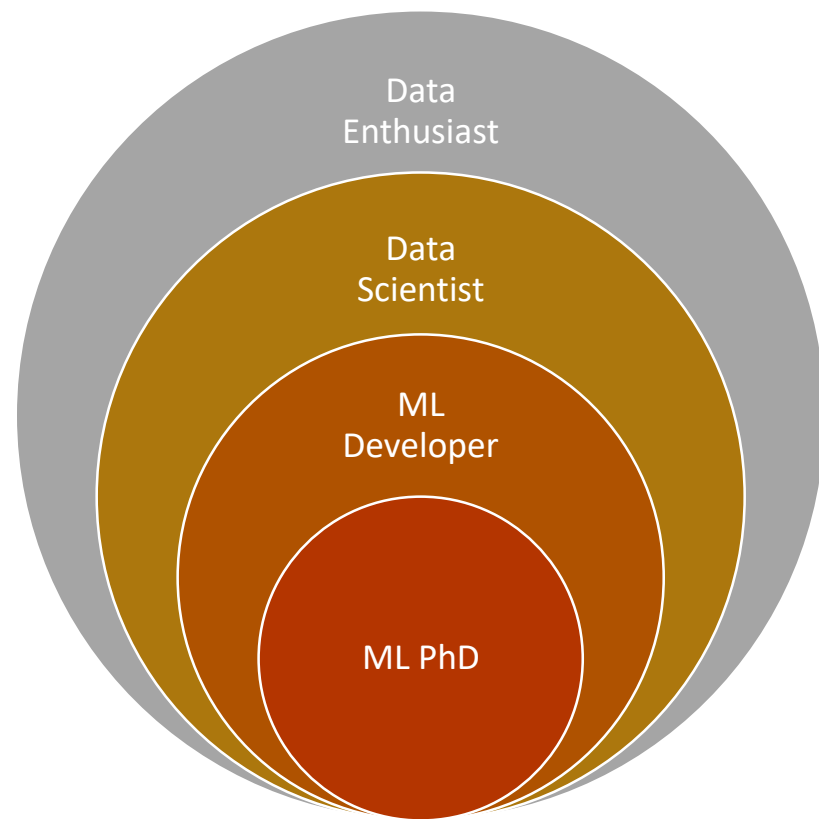
```
Bernoulli dist1 =  
    engine.Infer<Bernoulli>(coin1);
```

```
// 'dist1' is Bernoulli(0.333...)
```

[illegible]

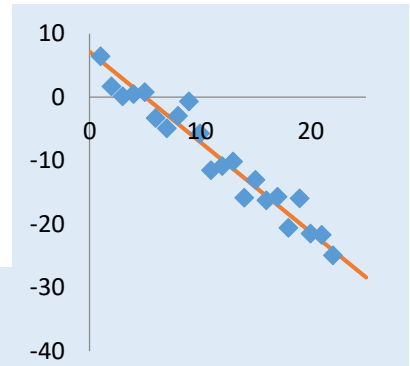
Tabular

- **DSL to get insight from data with probabilistic programs**



- Tabular has three guiding principles:
 - 1) Language design based on annotations of the data schema
 - 2) Inference by querying latent variables and missing values
 - 3) Auto-suggest models based on the structure of the data

Structure of a Model



A Bayesian model

$$y = Ax + B + e$$

where **noise** $e \sim N(0, 2^2)$ and **parameters** $A, B \sim N(0, 1)$

| XY | | | | |
|----|------|---------------|--------------------------------|--|
| A | real | static latent | Gaussian(0,1) | } Parameter distribution $p(A, B)$ |
| B | real | static latent | Gaussian(0,1) | |
| x | real | input | | } Sampling distribution $p(\vec{y} \mid A, B, \vec{x})$ (aka the likelihood) |
| y | real | output | Gaussian($A \cdot x + B, 4$) | |

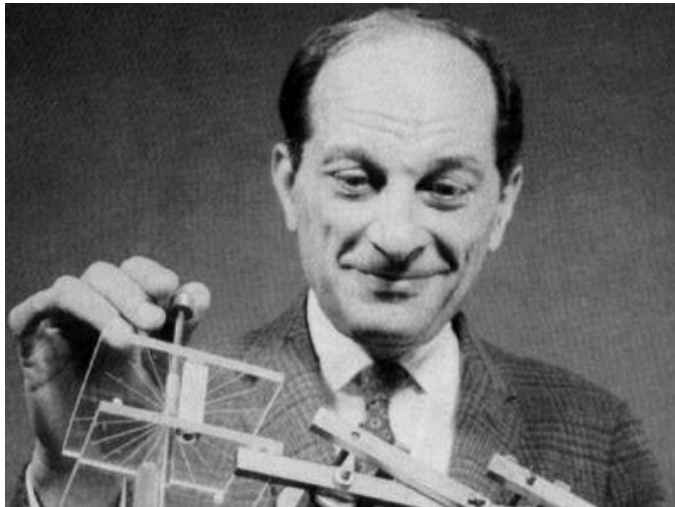
Predictive distribution $p(\vec{y} | \vec{x}) \triangleq \int p(A, B) p(\vec{y} \mid A, B, \vec{x}) d(A, B)$

With no observed outputs, these are the **prior** distributions

Conditioned on (some) outputs, we obtain the **posterior** distributions

Probabilistic Inference

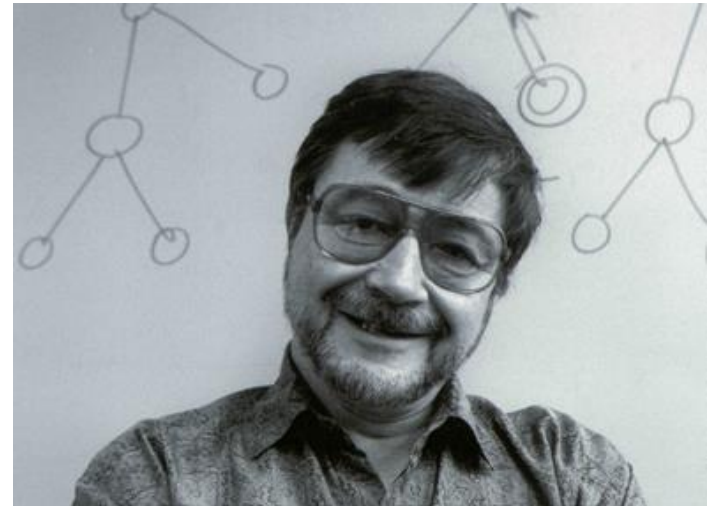
- Monte Carlo



Stanisław Ulam

- Set of samples for joint
- Slow but accurate

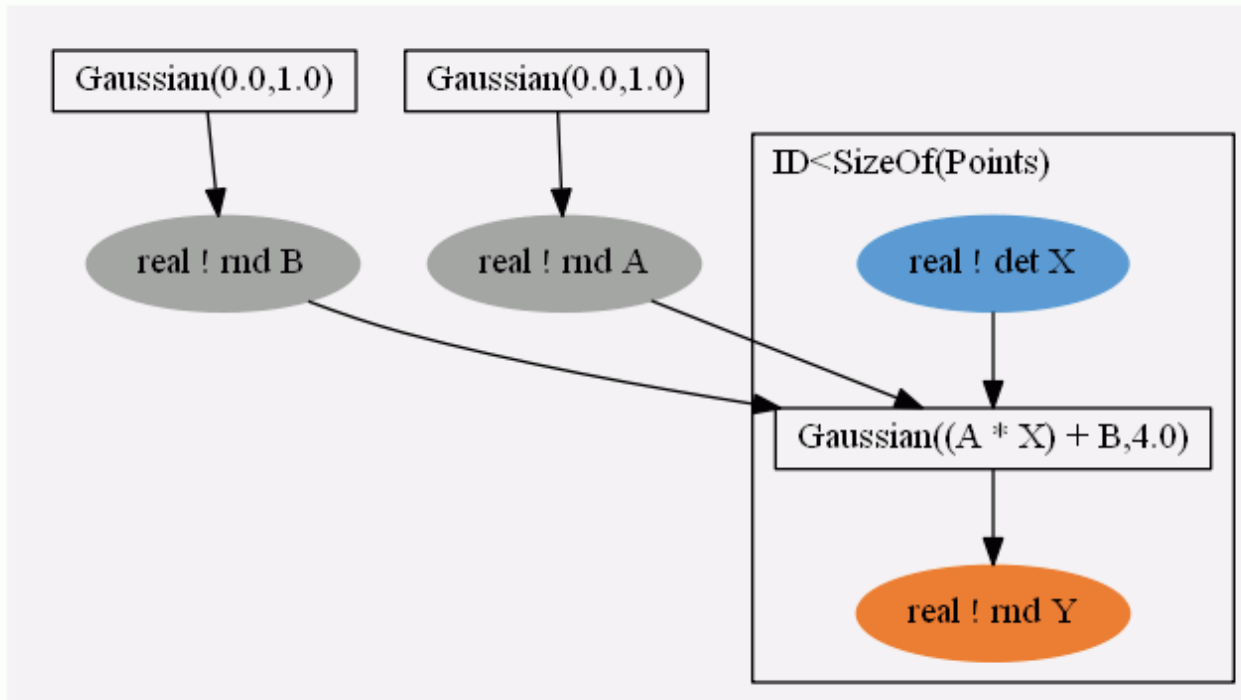
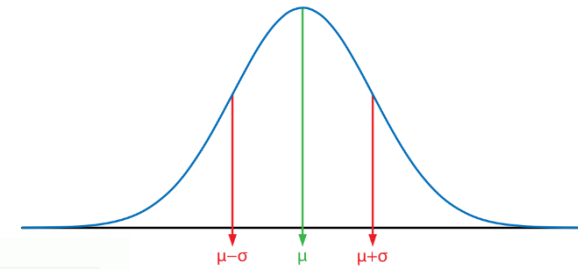
- Message Passing



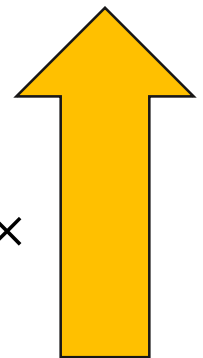
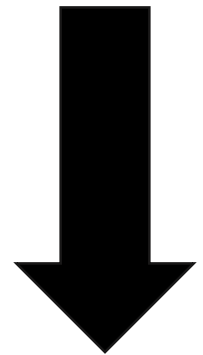
Judea Pearl

- Parameters for marginals
- Fast but approximate

Factor Graphs



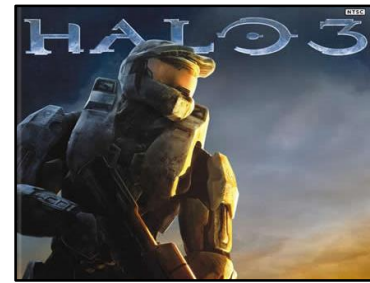
Sample



Infer

$$p(A, B, \vec{x}, \vec{y}) \triangleq pdf(A, Gaussian(0,1)) \times pdf(B, Gaussian(0,1)) \times \prod_i pdf(y_i, Gaussian(Ax_i + B, 4))$$

TrueSkill – Data



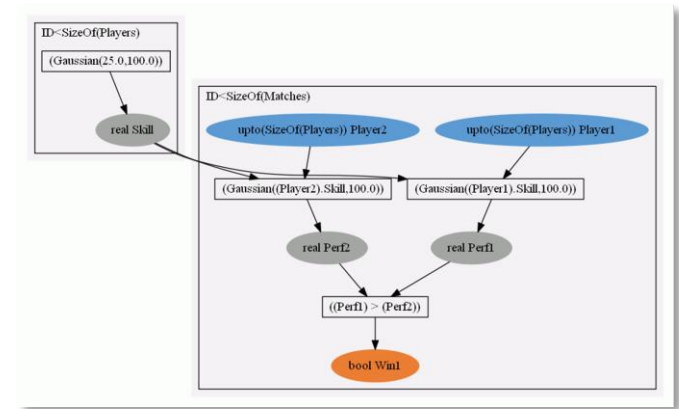
| Players | |
|---------|--------|
| Name | String |

| Matches | |
|---------|---------------|
| Player1 | Link(Players) |
| Player2 | Link(Players) |
| Win1 | Bool |

| | Name |
|---|---------|
| 0 | Alice |
| 1 | Bob |
| 2 | Cynthia |

| | Player1 | Player2 | Win1 |
|---|---------|---------|-------|
| 0 | 0 | 1 | False |
| 1 | 1 | 2 | False |

TrueSkill – Model



Players

| Name | string | input | |
|-------|--------|--------|----------------------|
| Skill | real | latent | Gaussian(25.0,100.0) |

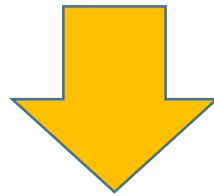
Matches

| Player1 | link(Players) | input | |
|---------|---------------|--------|---------------------------------|
| Player2 | link(Players) | input | |
| Perf1 | real | latent | Gaussian (Player1.Skill, 100.0) |
| Perf2 | real | latent | Gaussian (Player2.Skill, 100.0) |
| Win1 | bool | output | Perf1 > Perf2 |

Query-by-Latent Column

| | Name |
|---|---------|
| 0 | Alice |
| 1 | Bob |
| 2 | Cynthia |

| | Player1 | Player2 | Win1 |
|---|---------|---------|-------|
| 0 | 0 | 1 | False |
| 1 | 1 | 2 | False |



| | Skill |
|---|----------------------|
| 0 | Gaussian(20.25,82.3) |
| 1 | Gaussian(25.0,70.7) |
| 2 | Gaussian(29.75,82.3) |

| | Perf1 | Perf2 |
|---|------------------------|------------------------|
| 0 | Gaussian(15.49, 129.1) | Gaussian(29.75, 123.6) |
| 1 | Gaussian(20.25, 123.6) | Gaussian(34.51, 129.1) |

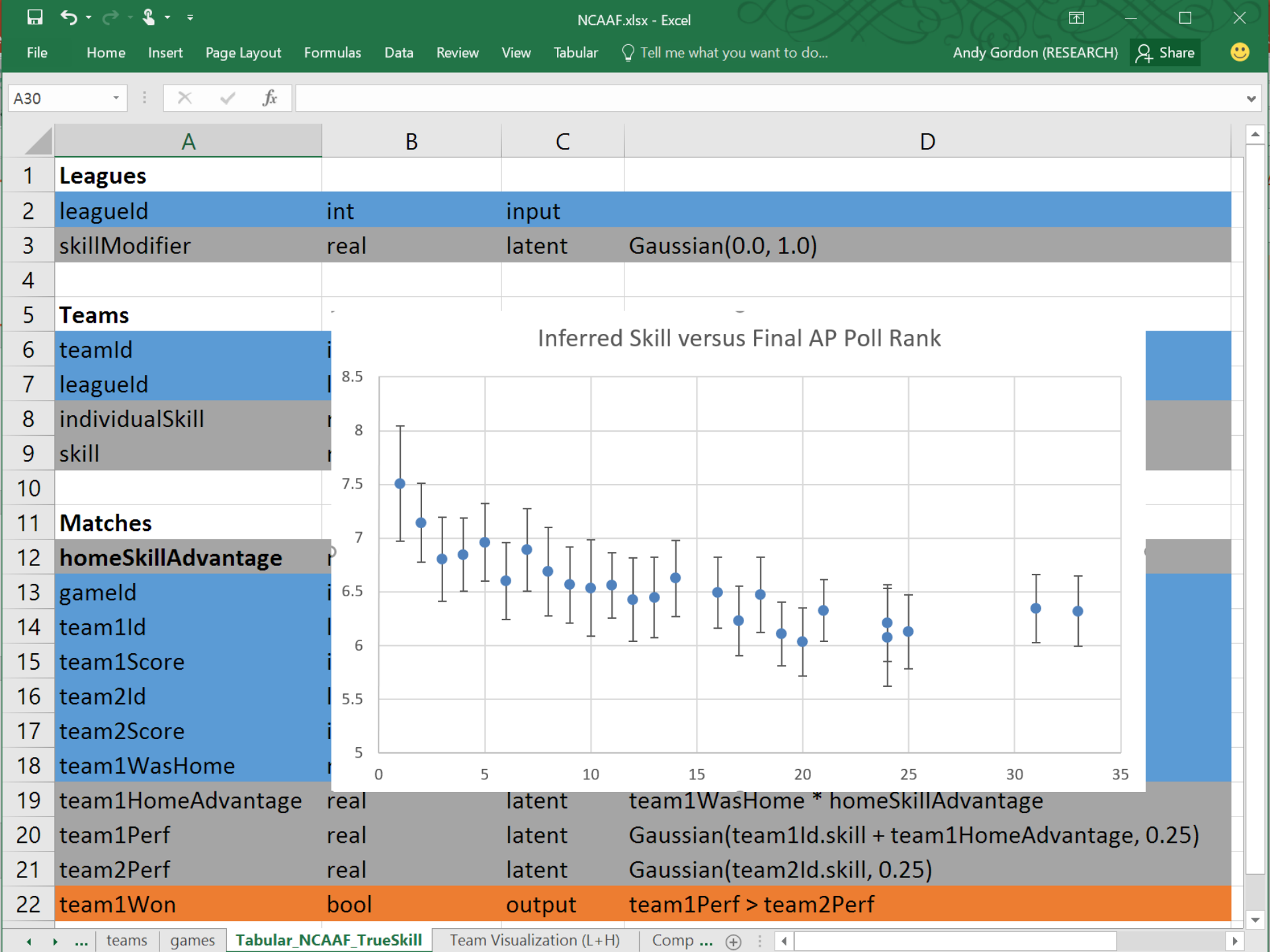
Query-by-Missing-Value

| | Name |
|---|---------|
| 0 | Alice |
| 1 | Bob |
| 2 | Cynthia |

| | Player1 | Player2 | Win1 |
|---|---------|---------|-------|
| 0 | 0 | 1 | False |
| 1 | 1 | 2 | False |
| 2 | 0 | 2 | |



| | Player1 | Player2 | Win1 |
|---|---------|---------|-----------------|
| 0 | 0 | 1 | False |
| 1 | 1 | 2 | False |
| 2 | 0 | 2 | Bernoulli(0.31) |



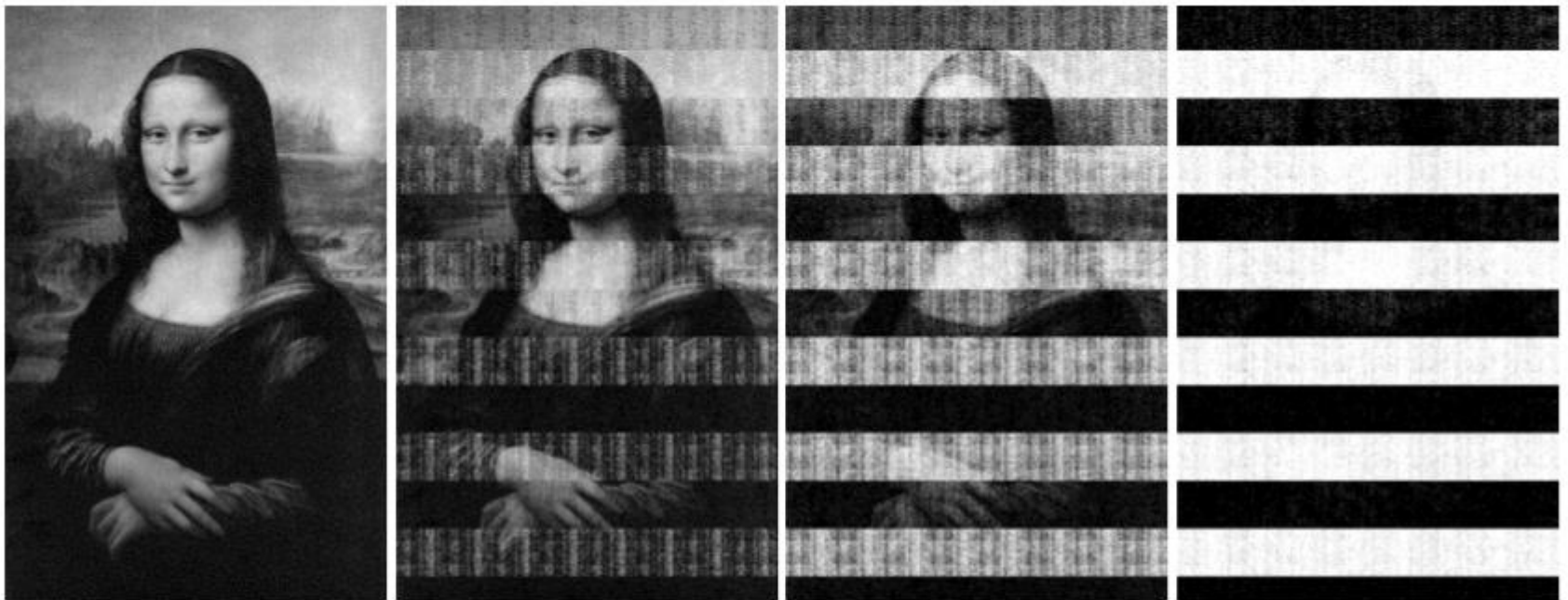
Lest We Remember: Cold Boot Attacks on Encryption Keys

J. Alex Halderman*, Seth D. Schoen†, Nadia Heninger*, William Clarkson*, William Paul‡, Joseph A. Calandrino*, Ariel J. Feldman*, Jacob Appelbaum, and Edward W. Felten*

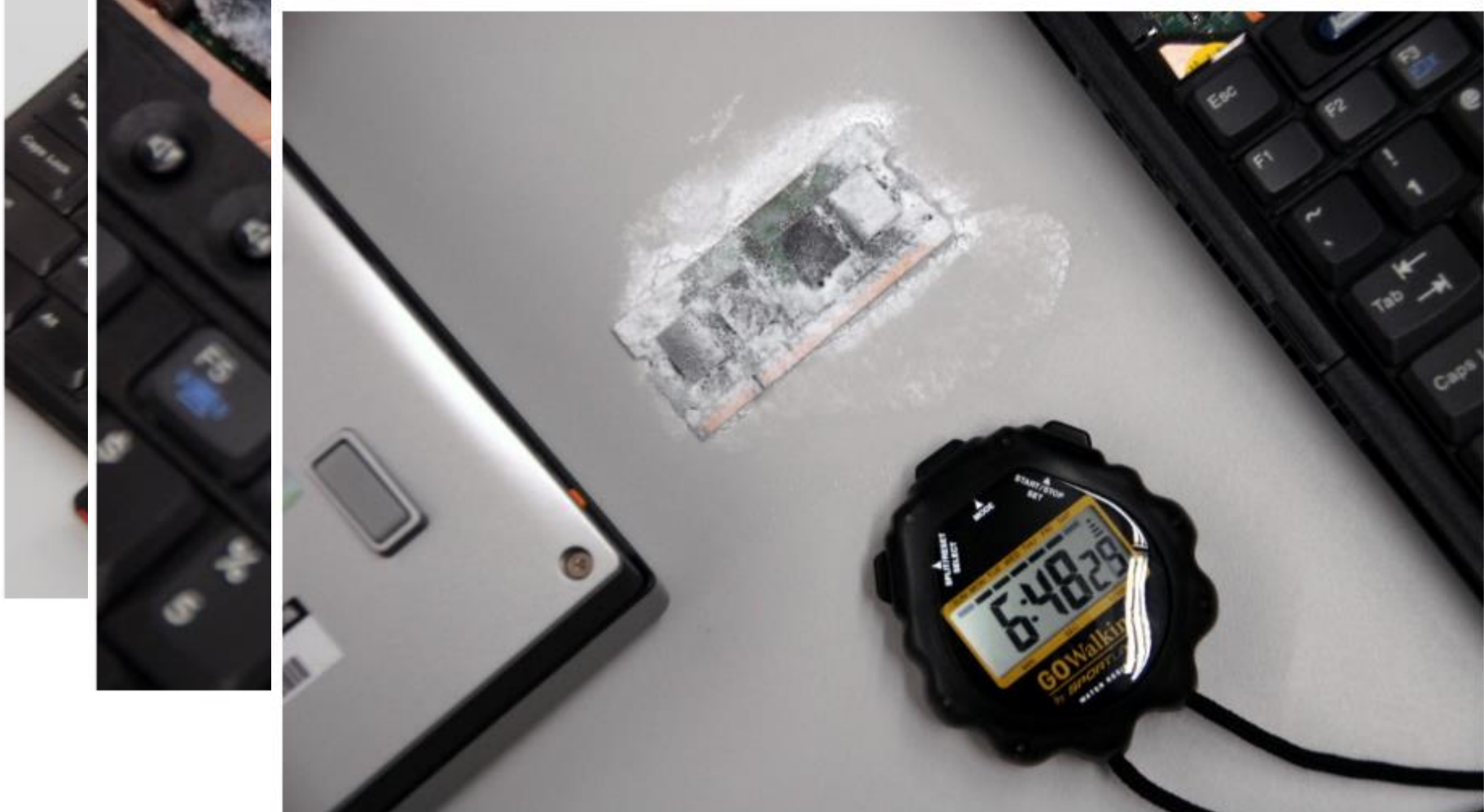
*Princeton University †Electronic Frontier Foundation ‡Wind River Systems
{jhalderm, nadiah, wclarkso, jcalandr, ajfeldma, felten}@cs.princeton.edu
schoen@eff.org, wpaul@windriver.com, jacob@appelbaum.net

Ex: Cold Boot Attacks

- DRAMs hold their values for long intervals without power or refresh.



Hence, sensitive information such as cryptographic keys can be extracted from memory, despite OS attempts to protect its contents.

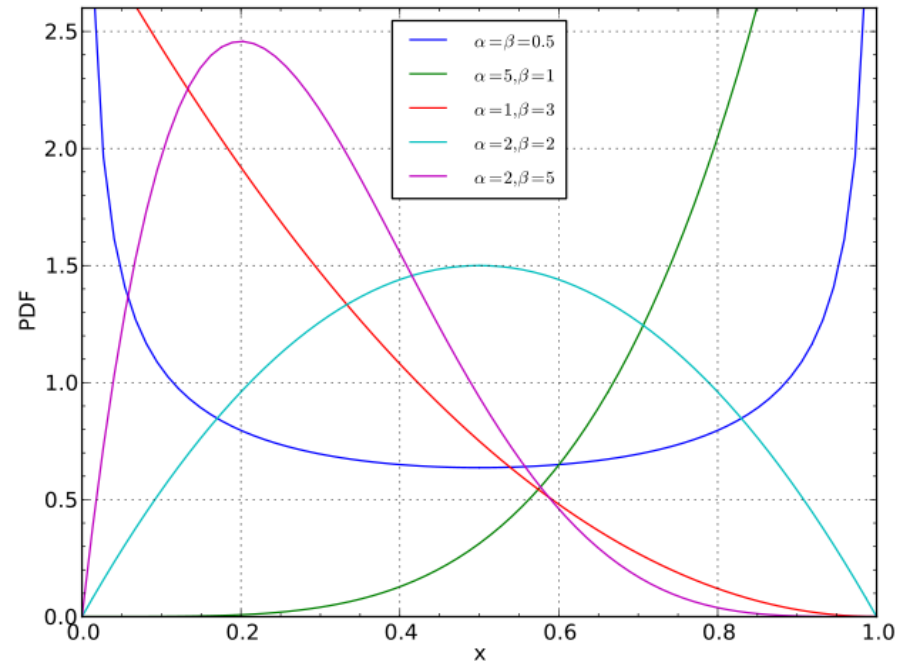


Recovering DES Keys

- The DES key schedule algorithm produces 16 subkeys, each a permutation of a 48-bit subset of bits from the original 56-bit key.
- Every bit of original key repeated in about 14 of 16 subkeys.
- Model the decay as a binary asymmetric channel
 - the probability of a 1 remaining 1 is some fixed Decay_{11}
 - the probability of a 0 flipping to a 1 is some fixed Decay_{01} .
- A probabilistic model can learn these parameters as well as estimating the latent key bits.

Bernoulli and Beta Distributions

- $Bernoulli(p)$ is the discrete distribution that returns true with probability p , returns false with probability $1 - p$
- $Beta(\alpha, \beta)$ is the continuous distribution over the bias p of a $Bernoulli(p)$, after observing true $\alpha - 1$ times, and false $\beta - 1$ times.
- For example, $Beta(1,1)$ is the uniform distribution.



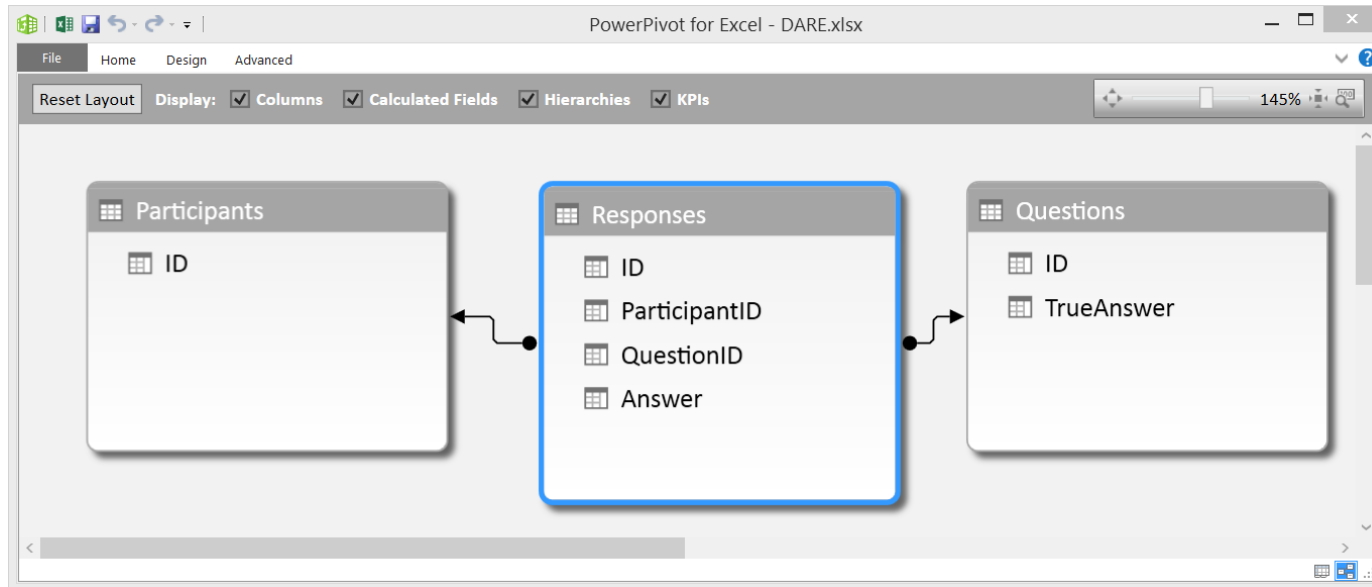
Source: Wikipedia

Tabular Demo

| Table1[ID] | | | |
|------------|------|--------|--|
| Decay01 | real | param | Beta(1.0,1.0) |
| Decay11 | real | param | Beta(1.0,1.0) |
| Bit | bool | output | Bernoulli(0.5) |
| Copy0 | bool | output | if Bit then Bernoulli(Decay11) else Bernoulli(Decay01) |
| Copy1 | bool | output | if Bit then Bernoulli(Decay11) else Bernoulli(Decay01) |
| Copy2 | bool | output | if Bit then Bernoulli(Decay11) else Bernoulli(Decay01) |
| Copy3 | bool | output | if Bit then Bernoulli(Decay11) else Bernoulli(Decay01) |

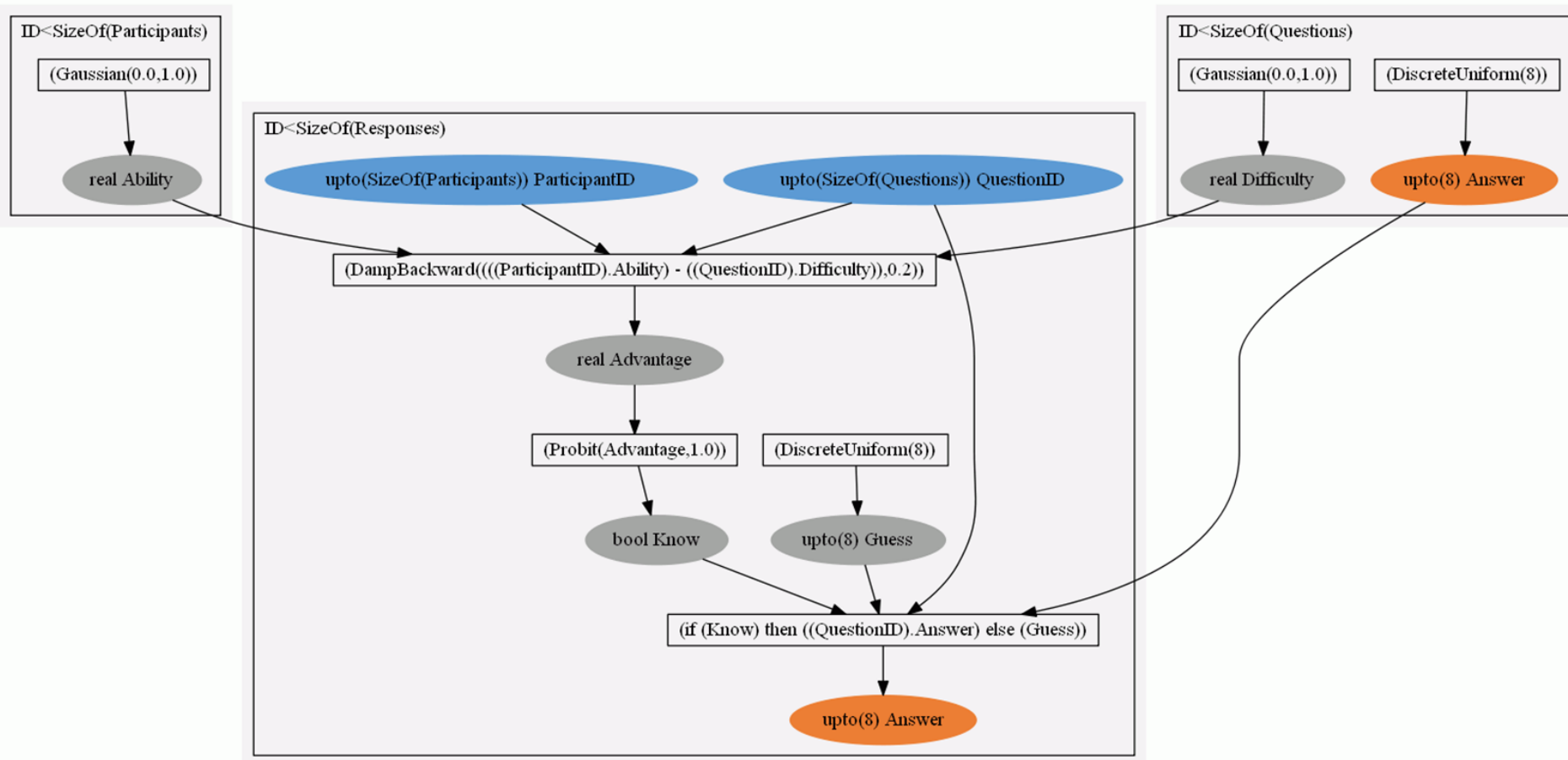
| Name | Value |
|---------|--------------------------------|
| Decay01 | Beta(1.524,11.75)[mean=0.1148] |
| Decay11 | Beta(21.04,31.5)[mean=0.4004] |

DARE (Bachrach et. al 2012)



- Data drawn from multiple choice intelligence test
- 60 questions with 8 possible answers (one correct)
- 121 participants
- Responses relate ParticipantID and QuestionID to Participant's answers.

Difficulty/Ability



Case study: Psychometrics

| Participants | | | | | | | | | var QuestionOfResponse = Variable.Array<int>(n).Named("questionOfResponse"); var ParticipantOfResponse = Variable.Array<int>(n).Named("participantOfResponse"); | |
|--------------|------------|----------|-----------|---------------|-----------|--|-----------|--------------------|---|-----------------------------|
| Model | Lang. | LOC Data | LOC Model | LOC Inference | LOC Total | Compile (s) | Infer (s) | Model log evidence | Avg. log prob. test Responses | Avg. log prob. test answers |
| A | Tabular II | 0 | 17 | 0 | 17 | 0.39 | 0.40 | -7499.74 | -1.432 | -3.424 |
| A | Infer.NET | 73 | 45 | 20 | 138 | 0.32 | 0.38 | -7499.74 | -1.432 | -3.425 |
| DA | Tabular II | 0 | 18 | 0 | 18 | 0.39 | 0.46 | -5933.52 | -1.118 | -0.739 |
| DA | Infer.NET | 73 | 47 | 21 | 141 | 0.34 | 0.43 | -5933.25 | -1.118 | -0.724 |
| DARE | Tabular II | 0 | 19 | 0 | 19 | 0.40 | 2.86 | -5820.40 | -1.119 | -0.528 |
| DARE | Infer.NET | 73 | 49 | 22 | 144 | 0.37 | 2.8 | -5820.40 | -1.119 | -0.528 |
| Know | | Bool | | Latent | | Probit(Advantage, QuestionID.Discrimination) | | | var TrainingQuestions.ObservedValue = nTrainingQuestions; ParticipantOfResponse.ObservedValue = participantOfResponse; QuestionOfResponse.ObservedValue = questionOfResponse; TrainingResponseIndices.ObservedValue = trainingResponseIndices; ObservedResponseAnswer.ObservedValue = Util.ArrayInt(nTrainingResponses, i => response(trainingResponseIndices[i])); TrainingQuestionIndices.ObservedValue = trainingQuestionIndices; ObservedQuestionAnswer.ObservedValue = Util.ArrayInt(nTrainingQuestions, i => answer(trainingQuestionIndices[i])); | |
| Guess | | Int | | Latent | | DiscreteUniform(8) | | | | |
| Response | | Int | | Output | | if Know then QuestionID.Answer else Guess | | | | |

```

public static void CreateAndRunDAREModel(
    Gaussian abilityPrior, Gaussian difficultyPrior, Gamma discriminationPrior,
    int nParticipants, int nQuestions, int nChoices,
    int[] participantOfResponse, int[] questionOfResponse,
    int[] response, int[] trainingResponseIndices,
    int[] answer, int[] trainingQuestionIndices)
{
    // Model
    var Evidence = Variable.Bernoulli(0.5).Named("evidence");
    var EvidenceBlock = Variable.If(Evidence);
    var NQuestions = Variable.New<int>().Named("nQuestions");
    var NParticipants = Variable.New<int>().Named("nParticipants");
    var NChoices = Variable.New<int>().Named("nChoices");
    var NResponses = Variable.New<int>().Named("nResponses");
    var NTrainingResponses = Variable.New<int>().Named("nTrainingResponses");
    var NTrainingQuestions = Variable.New<int>().Named("nTrainingQuestions");
    var p = new Range(NParticipants).Named("p");
    var q = new Range(NQuestions).Named("q");
    var c = new Range(NChoices).Named("c");
    var n = new Range(NResponses).Named("n");
    var tr = new Range(NTrainingResponses).Named("tr");
    var tq = new Range(NTrainingQuestions).Named("tq");
    var AnswerOfQuestion = Variable.Array<int>(q).Named("answer");
    AnswerOfQuestion[q] = Variable.DiscreteUniform(c).ForEach(q);
    var QuestionOfResponse = Variable.Array<int>(n).Named("questionOfResponse");
    var ParticipantOfResponse = Variable.Array<int>(n).Named("participantOfResponse");
}

```

```

// Training Questions Observed Value = nTrainingQuestions;
ParticipantOfResponse.ObservedValue = participantOfResponse;
QuestionOfResponse.ObservedValue = questionOfResponse;
TrainingResponseIndices.ObservedValue = trainingResponseIndices;
ObservedResponseAnswer.ObservedValue = Util.Array<int>(nTrainingResponses, i =>
    response[trainingResponseIndices[i]]);
TrainingQuestionIndices.ObservedValue = trainingQuestionIndices;
ObservedQuestionAnswer.ObservedValue = Util.Array<int>(nTrainingQuestions, i =>
    answer[trainingQuestionIndices[i]]);
var Engine = new InferenceEngine();
{ ShowTimings = true, ShowWarnings = false, ShowProgress = false, NumberOfIterations = 10 };
var AnswerOfResponsePosterior = Engine.Infer<Discrete>().ForEach(AnswerOfResponse);
var AnswerOfQuestionPosterior = Engine.Infer<Discrete>().ForEach(AnswerOfQuestion);
var LogEvidence = Engine.Infer<Bernoulli>().ForEach(Evidence).LogOdds;
var AbilityPosterior = Engine.Infer<Gaussian>().ForEach(ability);
var DifficultyPosterior = Engine.Infer<Gaussian>().ForEach(difficulty);
var DiscriminationPosterior = Engine.Infer<Gamma>().ForEach(discrimination);
}

```

- Replicated Infer.NET C# by Bachrach et al 2012
- Same statistical results, much less code, slight loss in perf

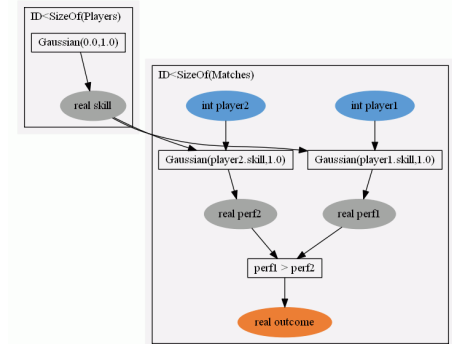
Core Syntax of Schemas

| | |
|-------------|--|
| schema | $S ::= (t_1 T_1) \dots (t_n T_n)$ |
| table model | $T ::= (c_1 ty_1 \ell_1 A_1) \dots (c_n ty_n \ell_n A_n)$ |
| column type | $ty ::= \mathbf{bool} \mid \mathbf{int} \mid \mathbf{mod}(n) \mid \mathbf{real} \mid T[n]$ |

| | |
|------------|---|
| level | $\ell ::= \mathbf{static} \mid \epsilon$ |
| annotation | $A ::= \mathbf{input} \mid \mathbf{latent}(E) \mid \mathbf{output}(E)$ |
| expression | $E ::= c \mid E.c \mid f(E_1, \dots, E_n) \mid D(E_1, \dots, E_n) \mid \dots$ |

- Attribute marked **static** is single parameter, else whole column
- **input** attribute conditions the model, but cannot be predicted
- **latent**(E) attribute defines hidden variable by E
- **output**(E) attribute conditions the model, predicted by E

Semantics of Schemas



Players

| Name | string | input | |
|-------|--------|--------|------------------|
| Skill | real | latent | Gaussian(25,100) |

For each row r of Players,
 $Skill \sim \text{Gaussian}(25, 100)$

Matches

| Player1 | link(Players) | input | |
|---------|---------------|--------|------------------------------|
| Player2 | link(Players) | input | |
| Perf1 | real | latent | Gaussian (Player1.Skill,100) |
| Perf2 | real | latent | Gaussian (Player2.Skill,100) |
| Win1 | bool | output | Perf1 > Perf2 |

For each row r of Matches,
 Player1 := r .Player1
 Player2 := r .Player2
 Perf1 \sim Gaussian(Players[Player1].Skill,100)
 Perf2 \sim Gaussian(Players[Player2].Skill,100)
 Win1 := (Perf1 > Perf2)
 if (r .Win1 not null)
observe (Win1 == r .Win1)

Infer and Decision Making

Players

| | | | |
|-------|------|--------|----------------------|
| Skill | real | latent | Gaussian(25.0,100.0) |
|-------|------|--------|----------------------|

Matches

| | | | |
|---------|---------------|--------|---|
| Player1 | link(Players) | input | |
| Player2 | link(Players) | input | |
| Win1 | bool | output | Gaussian(Player1.Skill,100.0) > Gaussian(Player2.Skill,100.0) |

Bets

| | | | |
|----------|---------------|--------|------------------------------------|
| Match | link(Matches) | input | |
| Odds1 | real | input | |
| p | real | latent | infer.Bernoulli[].Bias(Match.Win1) |
| EU | real | latent | $p * Odds1 - (1.0 - p)$ |
| PlaceBet | bool | latent | $EU > 0.0$ |

Matches

| Match | Player1 | Player2 | Win1 |
|-------|---------|---------|-------|
| 0 | Alice | Bob | FALSE |
| 1 | Bob | Cynthia | FALSE |
| 2 | Alice | Cynthia | |

Bets

| Match | Odds1 |
|-------|-------|
| 2 | 4 |
| 2 | 2 |



Bets

| Bet | Match | Odds1 | p | EU | PlaceBet |
|-----|-------|-------|-------------|--------------|----------|
| 0 | 2 | 4 | 0.309234026 | 0.546170128 | TRUE |
| 1 | 2 | 2 | 0.309234026 | -0.072297923 | FALSE |

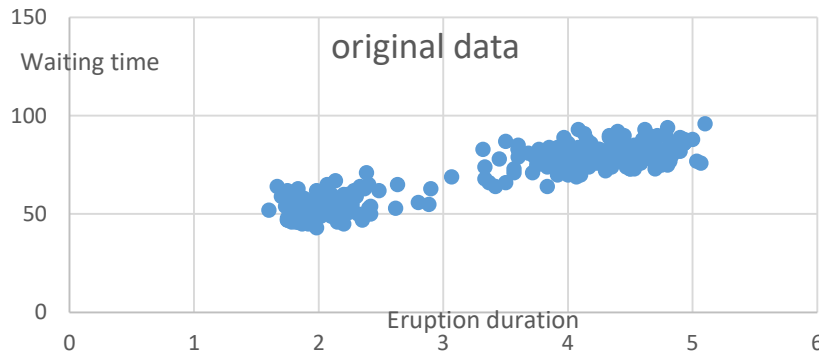
Beyond the Core Syntax

| | |
|-------------|--|
| schema | $S ::= (t_1 T_1) \dots (t_n T_n)$ |
| table model | $T ::= (c_1 ty_1 \ell_1 A_1) \dots (c_n ty_n \ell_n A_n)$ |
| column type | $ty ::= \mathbf{bool} \mid \mathbf{int} \mid \mathbf{mod}(n) \mid \mathbf{real} \mid T[n]$ |

| | |
|------------------|---|
| level | $\ell ::= \mathbf{static} \mid \epsilon$ |
| annotation | $A ::= \mathbf{input} \mid \mathbf{latent}(M) \mid \mathbf{output}(M)$ |
| model expression | $M ::= E \mid t(h_1=E_1, \dots, h_n=E_n) \mid M[E_{index} < E_{size}]$ |
| expression | $E ::= c \mid E.c \mid f(E_1, \dots, E_n) \mid D(E_1, \dots, E_n) \mid \dots$ |

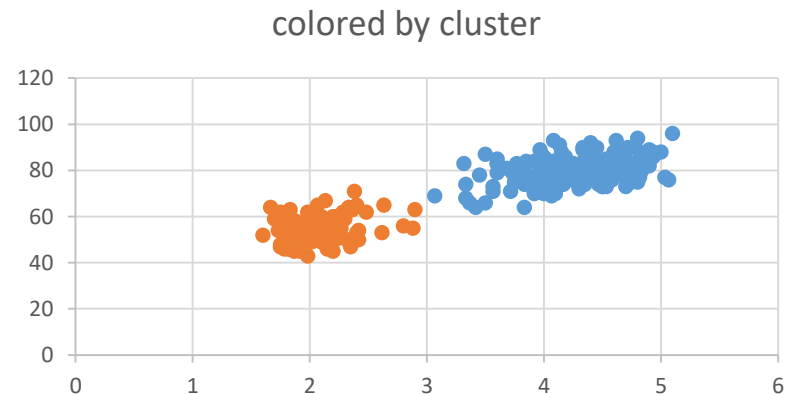
- An extended syntax (that compiles to the core):
 - Functions defined by table models T
 - Application $t(h_1=E_1, \dots, h_n=E_n)$ defined by substitution semantics
 - Indexed models $M[E_{index} < E_{size}]$ capture structure of many models

Functions and Indexing, by example



faithful

| cluster | mod(2) | latent | Discrete(2,[0.5, 0.5]) |
|-------------------|--------|--------|---------------------------------|
| eruption_duration | real | output | CG()[cluster] |
| waiting_time | real | output | CG(MeanOfMean=60.0)[cluster] |
| assignment | mod(2) | latent | infer.Discrete[2].Mode(cluster) |



Functional Programming

function:CG

| | | | |
|------------|------|--------------|-----------------------------------|
| MeanOfMean | real | static input | 0.0 |
| PrecOfMean | real | static input | 1.0 |
| Mean | real | param | Gaussian(MeanOfMean,1/PrecOfMean) |
| Prec | real | param | Gamma(1.0,1.0) |
| CG | real | output | Gaussian(Mean,1/Prec) |

faithful

| | | | |
|-------------------|------|--------|------|
| eruption_duration | real | output | CG() |
|-------------------|------|--------|------|

faithful

| | | | |
|-------------------|------|--------|-----------------------|
| Mean | real | param | Gaussian(0.0,1.0) |
| Prec | real | param | Gamma(1.0,1.0) |
| eruption_duration | real | output | Gaussian(Mean,1/Prec) |

Indexed Models

- An **indexed model** $M[E_{index} < E_{size}]$ is a model with an array of E_{size} copies of the parameters of M ; the parameter to produce each output is selected by E_{index} .

faithful

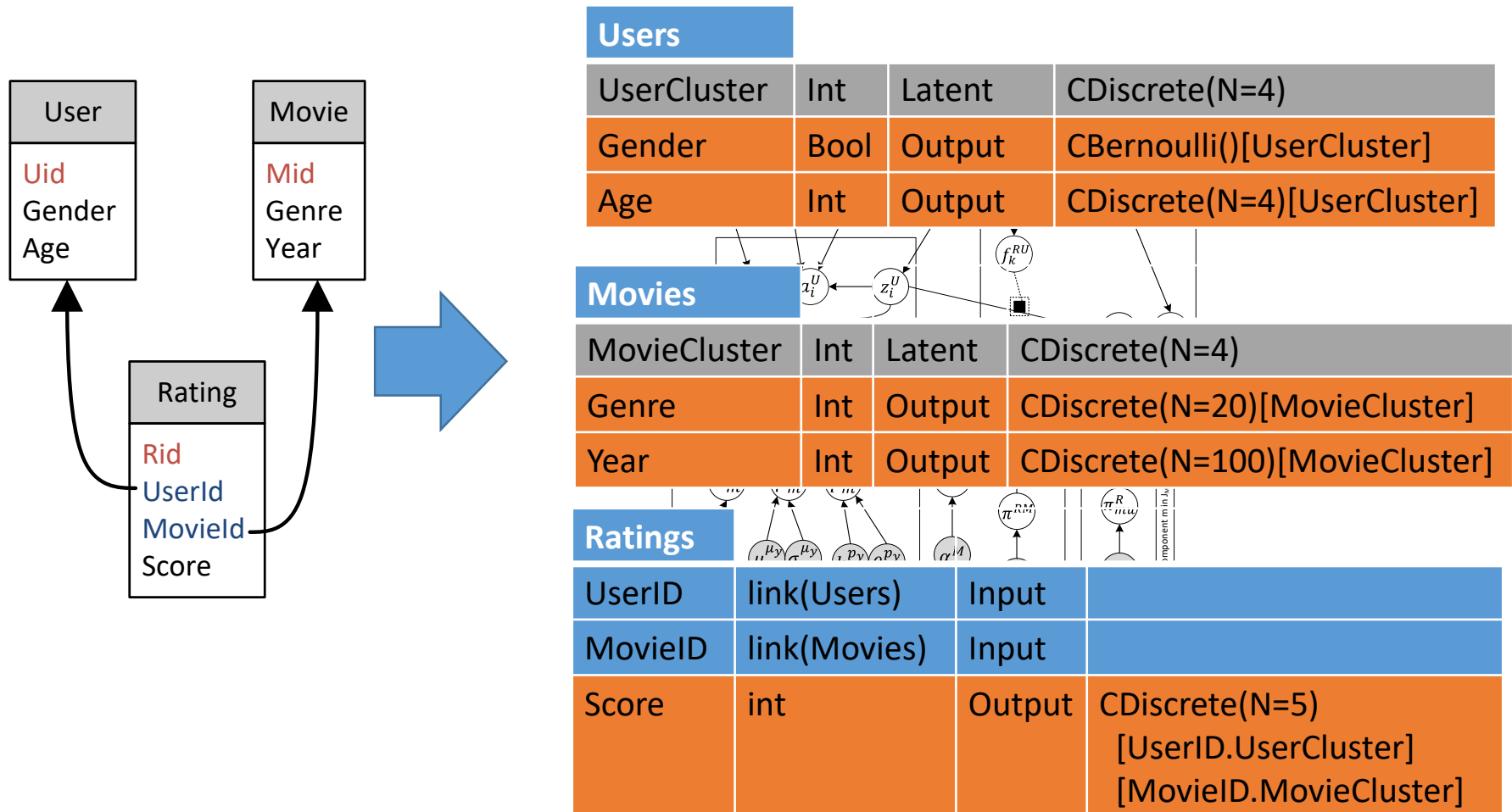
| | | | |
|-------------------|--------|--------|------------------------|
| cluster | mod(2) | latent | Discrete(2,[0.5, 0.5]) |
| eruption_duration | real | output | CG()[cluster < 2] |



faithful

| | | | |
|-------------------|---------|--------|---|
| cluster | mod(2) | latent | Discrete(2,[0.5, 0.5]) |
| Mean | real[2] | param | [for i < 2 -> Gaussian(0.0,1.0)] |
| Prec | real[2] | param | [for i < 2 -> Gamma(1.0,1.0)] |
| eruption_duration | real | output | Gaussian(Mean.[cluster],1/Prec.[cluster]) |

Automatic Model Suggestion



Model expressions allow us to recast early work on model suggestion as **probabilistic metaprogramming**

Key Features of Tabular

- Schema-driven probabilistic programming
- Unique table-based syntax for embedding in spreadsheets
- Query by missing value and latent column
- Core interpreted as factor graph structured as a Bayesian model: parameters and outputs
- Functions and indexed models capture common structure
- Post-processing for Bayesian decision-making
- Sized-types for arrays and distributions
- Type-based information-flow analysis usefully distinguishes the stochastic and deterministic parts of a probabilistic program
- R-style regression calculus for hierarchical regressions
- Papers with metatheory: POPL'14, ESOP'15, POPL'16

Skype Abuse Detection with Probabilistic Bayesian Modeling

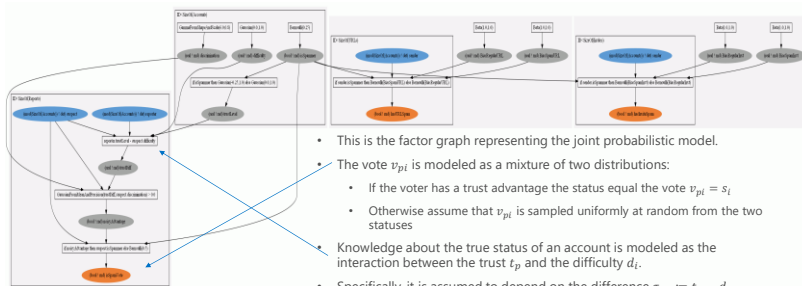
Natalia Larios
Applied Researcher
Membership – Safety Platform



Bayesian Model Proposal

- We propose a framework that uses the information provided by other accounts, combined with additional sources that detect abuse in URLs and invites to infer:
 - The spammer nature of an account,
 - the trustworthiness of the information from each account,
 - and the difficulty to evaluate them.

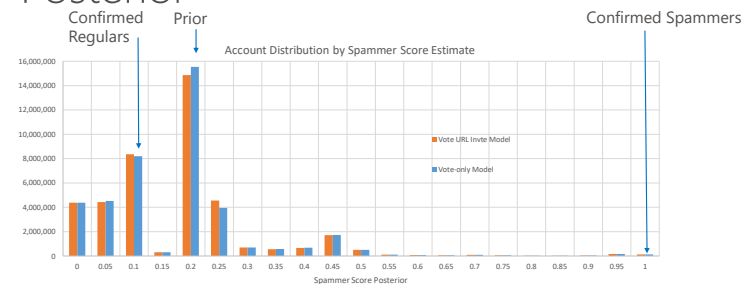
Probabilistic Model for Skype Abuse Detection



- This is the factor graph representing the joint probabilistic model.
- The vote v_{pi} is modeled as a mixture of two distributions:
 - If the voter has a trust advantage the status equal the vote $v_{pi} = s_i$
 - Otherwise assume that v_{pi} is sampled uniformly at random from the two statuses
- Knowledge about the true status of an account is modeled as the interaction between the trust t_p and the difficulty d_i .
- Specifically, it is assumed to depend on the difference $\tau_{pi} := t_p - d_i$

31

Account Distribution by Spammer Posterior



34

Internal evaluation of a Tabular model for spam detection on a social network

Download and Enjoy



Basic -
Coins



Cheatsheet



Classificatio
n - BPM



Classificatio
n - Dogs



Classificatio
n -
LogisticReg
ression



Classificatio
n -
Mammogr
aphy



Classificatio
n - Naive
Bayes



Classificatio
n -
VectorBPM



Clustering -
Faithful



Clustering -
LDA small



Clustering -
Multivariate
Faithful



NaiveBayes
Query



PCA



Ranking -
Recommen
der



Ranking -
TrueSkill
example



Ranking -
TrueSkill
large



Ranking -
TrueSkill
Raw Data



Ranking -
TrueSkillBet
s



Ranking -
TrueSkillBet
s-large



Regression
-
Hierarchical
LinearReg...



Regression
-
LinearRegre
ssion



Regression
-
MultiLinear
Regression

<https://aka.ms/tabular>

Fabular: Regression Formulas as Probabilistic Programming

Johannes Borgström, Andrew D. Gordon, Long Ouyang,
Claudio Russo, Adam Ścibior, Marcin Szymczak
Uppsala, MSR, Edinburgh, Stanford, Cambridge,

<https://aka.ms/tabular>

Probabilistic programming:

```
let  $\alpha$  = Gaussian( $0, s_{large}^2$ ) in
let  $\beta$  = Gaussian( $0, s_{large}^2$ ) in
let  $\pi$  = Gamma( $1, \lambda_{small}$ ) in
(( $\alpha, \beta$ ), [for  $z < \underline{students}$   $\rightarrow \alpha + x[z] \times \beta + \text{Gaussian}(0, 1/\pi)$ ])
coefficients                                predicted values of the y variable
```

In another galaxy, R's lm package:

$$y \sim 1 + x$$

$$y \sim x$$

Our goal: embrace and extend R's formulas to
get more succinct probabilistic programs

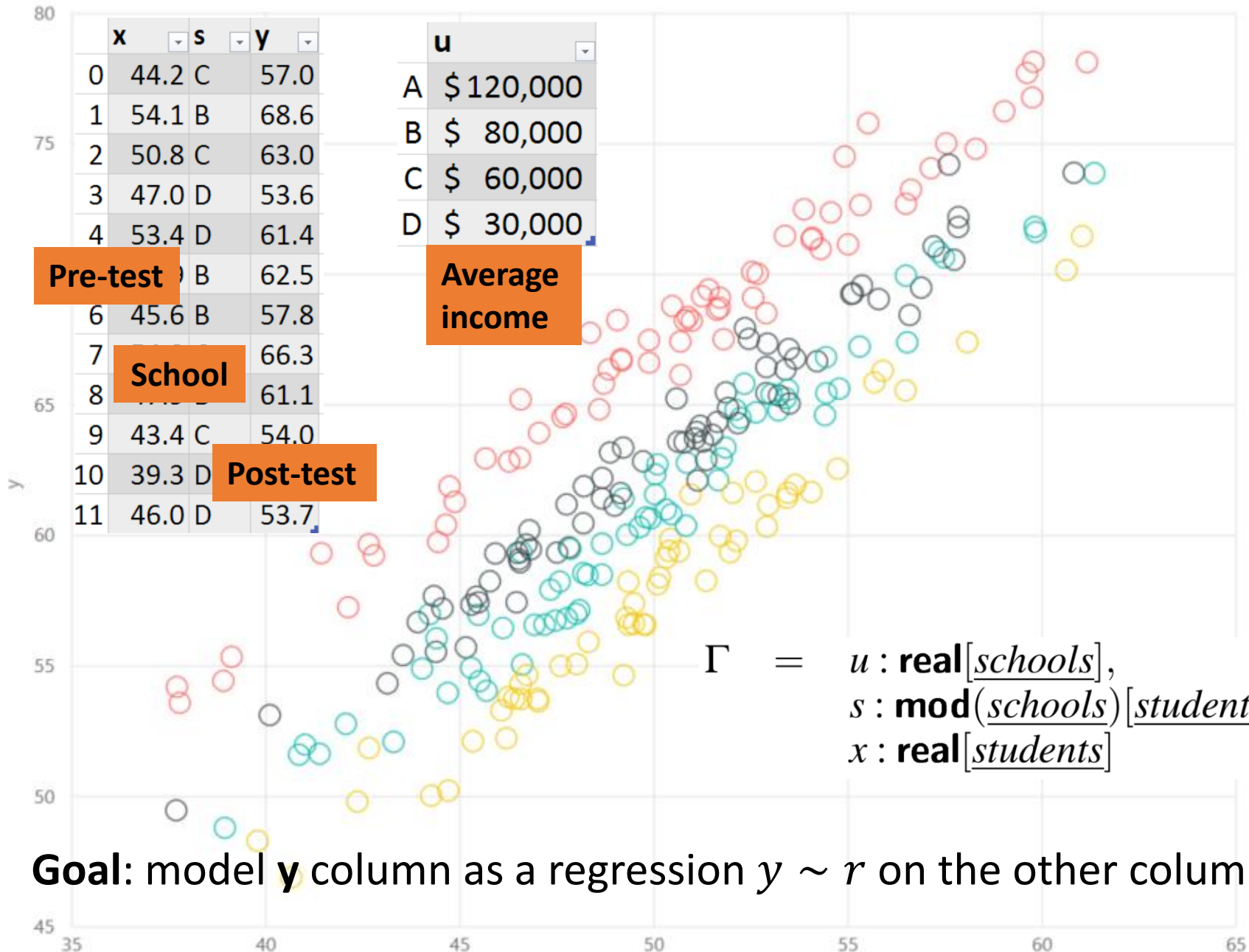
A regression calculus $y \sim r$

| $r ::=$ | regression |
|----------------------|--|
| $D(v_1, \dots, v_n)$ | noise with distribution D |
| $v\{\alpha \sim r\}$ | predictor with coefficient (new wrt R) |
| $r + r'$ | sum |
| $r \mid v$ | grouping |
| $(v\alpha)r$ | restriction (scope of α is r) (new wrt R) |

| $u, v ::=$ | predictor |
|------------|--------------------------------------|
| s | scalar (common cases are 1 and 0) |
| x | variable (categorical or continuous) |
| $u : v$ | interaction (multiplication) |

x and y by id and s

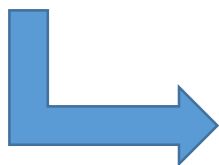
s ● A ● B ● C ● D



Goal: model y column as a regression $y \sim r$ on the other columns

(1) Pure Intercept

$$y \sim 1\{\alpha\} \sim \text{Gaussian}(0, s_{\text{large}}^2)$$

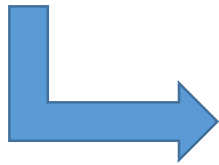


let $\alpha = \text{Gaussian}(0, s_{\text{large}}^2)$ **in**
 $(\alpha, [\text{for } z < \underline{\text{students}} \rightarrow 1 \times \alpha])$

$$\begin{aligned} v\{\alpha\} &\triangleq v\{\alpha \sim \text{Gaussian}(0, s_{\text{large}}^2)\} \\ v &\triangleq (v\alpha)v\{\alpha\} \quad \text{for } \alpha \notin \text{fv}(v) \end{aligned}$$

(2) Pure Noise

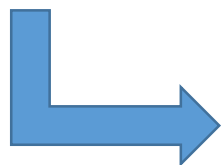
$y \sim ?$



```
let  $\pi$  = Gamma(1,  $\lambda_{small}$ ) in  
(( ), [for  $z < \underline{students}$   $\rightarrow$  Gaussian(0, 1/ $\pi$ )])
```

(3) Intercept with Noise

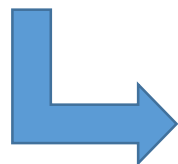
$$y \sim 1\{\alpha\} + ?$$



```
let  $\alpha$  = Gaussian(0,  $s_{large}^2$ ) in  
let  $\pi$  = Gamma(1,  $\lambda_{small}$ ) in  
(( $\alpha$ ), [for  $z < \underline{students}$   $\rightarrow \alpha + \text{Gaussian}(0, 1/\pi)$ ])
```

(4) Slope and Intercept

$$y \sim 1\{\alpha\} + x\{\beta\} + ?$$



let $\alpha = \text{Gaussian}(0, s_{\text{large}}^2)$ **in**

let $\beta = \text{Gaussian}(0, s_{\text{large}}^2)$ **in**

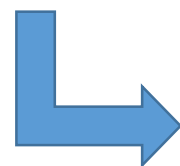
let $\pi = \text{Gamma}(1, \lambda_{\text{small}})$ **in**

$((\alpha, \beta), [\text{for } z < \underline{\text{students}} \rightarrow \alpha + x[z] \times \beta + \text{Gaussian}(0, 1/\pi)])$

$$y \sim 1 + x$$

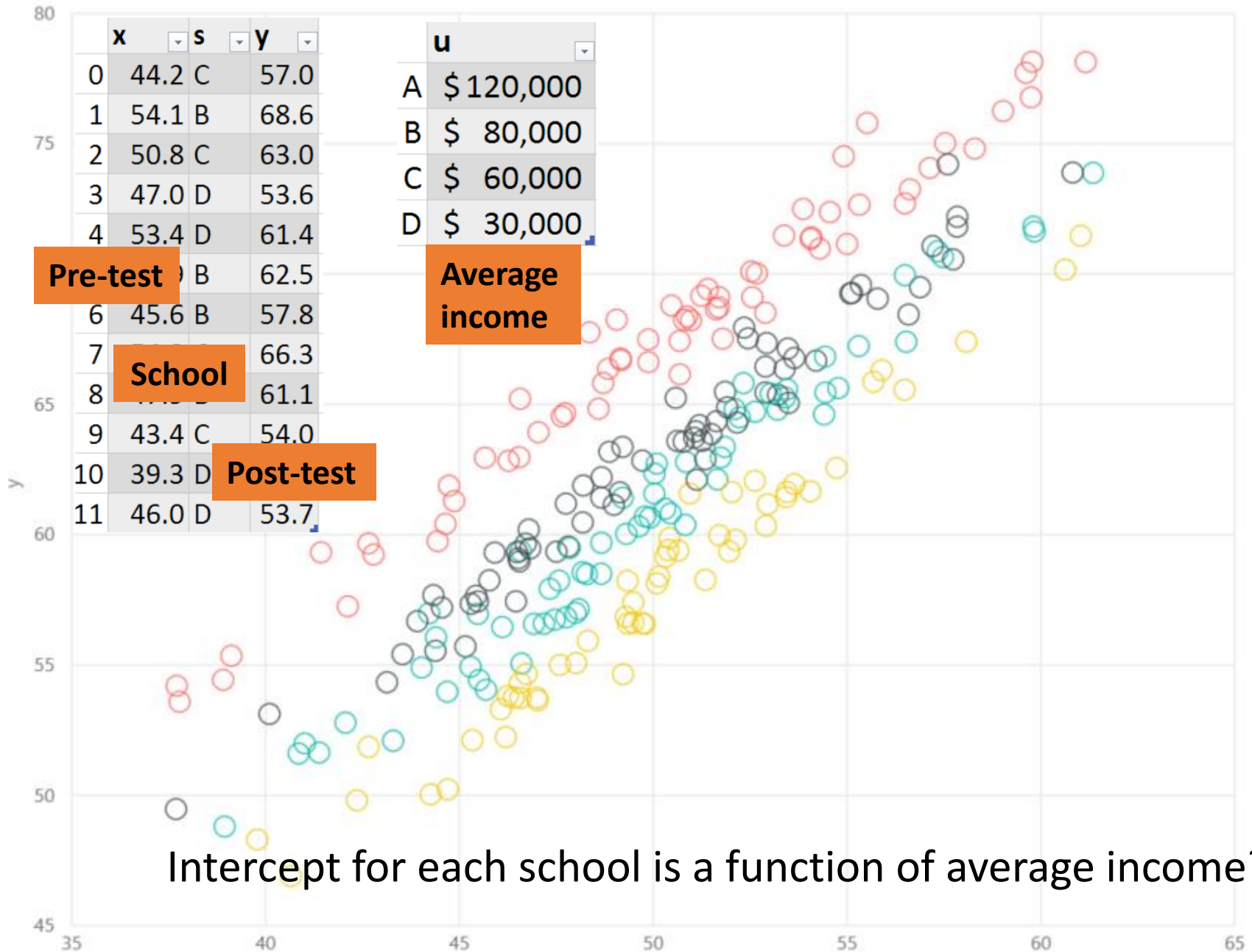
(5) Intercept per School

$$y \sim (1 \{ \alpha \} \mid s)$$

 **let** $\alpha = [\text{for } z < \underline{schools} \rightarrow \text{Gaussian}(0, s_{large}^2)]$ **in**
let $\pi = \text{Gamma}(1, \lambda_{small})$ **in**
 $((\alpha), [\text{for } z < \underline{students} \rightarrow \alpha[s[z]] + \text{Gaussian}(0, 1/\pi)])$

x and y by id and s

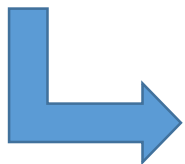
s ● A ● B ● C ● D



(6) Hierarchical Regression

$$y \sim (1\{\alpha \sim r_\alpha\} \mid s) + x\{\beta\}$$

where $r_\alpha = 1\{a\} + u\{b\}$



```
let  $a = \text{Gaussian}(0, s_{\text{large}}^2)$  in  
let  $b = \text{Gaussian}(0, s_{\text{large}}^2)$  in  
let  $\pi' = \text{Gamma}(1, \lambda_{\text{small}})$  in  
let  $\alpha = [\text{for } z < \underline{\text{schools}} \rightarrow a + u[z] \times b + \text{Gaussian}(0, 1/\pi')]$  in  
let  $\beta = \text{Gaussian}(0, s_{\text{large}}^2)$  in  
let  $\pi = \text{Gamma}(1, 1/\lambda_{\text{small}})$  in  
 $((a, b, \alpha, \beta), [\text{for } z < \underline{\text{students}} \rightarrow \alpha[s[z]] + x[z] \times \beta + \text{Gaussian}(0, 1/\pi)])$ 
```

A hierarchical model is “a regression in which the parameters – the regression coefficients – are given a probability model.” (Gelman and Hill 2007)

Summary so Far

$$\begin{array}{ll}
 r ::= & u, v ::= \\
 & D(v_1, \dots, v_n) \\
 & v\{\alpha \sim r\} \\
 & r + r' \\
 & r \mid v \\
 & (v\alpha)r \\
 & s \\
 & x \\
 & u : v
 \end{array}$$

| Model | Regression | Evidence |
|----------------------|---------------------------------------|----------|
| Pure Intercept | 1 | fails |
| Pure Noise | ? | -5577 |
| Intercept With Noise | 1 | -3320 |
| Slope and Intercept | 1 + x | -2695 |
| Intercept Per School | $(1\{\alpha\} \mid s)$ | -3174 |
| Hierarchical | $(1\{\alpha \sim 1 + u\} \mid s) + x$ | -1871 |

- No prior formalisations of R-style formulas
- Unlike R formulas, this **regression calculus** can specify priors and also hierarchical models

Fabular = Tabular + Formulas

| Points | | | |
|--------|------|-------------------|--|
| X | real | input | |
| Y | real | output $\sim X+1$ | |



| Points | | | |
|--------|------|--------|---------------------------------|
| X | real | input | |
| Y_0 | real | param | Gaussian(0.0,1e05) |
| Y_1 | real | param | Gaussian(0.0,1e05) |
| v___1 | real | param | Gamma(1.0,100000.0) |
| Y | real | output | Gaussian((X * Y_0) + Y_1,v___1) |

Fabular = Tabular + Formulas

| ID | Y | X0 | X1 | X2 | X3 | X4 | X5 | X6 | |
|----|-------|----|----|---------|---------|---------|---------|---------|---------|
| 0 | FALSE | | 1 | -0.7844 | -0.4702 | -0.5916 | -0.8596 | -0.3779 | -0.9457 |
| 1 | TRUE | | 1 | 0.5969 | 0.7905 | -0.2761 | 1.1607 | -0.3779 | 1.1842 |
| 2 | FALSE | | 1 | 0.1516 | 0.7153 | -0.1859 | 0.6025 | 1.4231 | 0.9191 |
| 3 | FALSE | | 1 | 0.5684 | 1.6885 | -0.5015 | 0.5738 | -0.3779 | 0.8125 |

Z_0 real param Gaussian(0.0,1e-05)

Z_1 real param Gaussian(0.0,1e-05)

Z_2 real param Gaussian(0.0,1e-05)

Z_3 real param Gaussian(0.0,1e-05)

Z_4 real param Gaussian(0.0,1e-05)

Z_5 real param Gaussian(0.0,1e-05)

Z_6 real param Gaussian(0.0,1e-05)

v__2 real param Gamma(1.0,100000.0)

Z real latent (((((((X1 * Z_0) + (X2 * Z_1)) + (X3 * Z_2)) + (X4 * Z_3)) + (X5 * Z_4)) + (X6 * Z_5)) + Z_6) + Gaussian(0.0,v__2)

Y boo output Z > 0.0

| Log Evidence | Data | |
|--------------|------|---|
| -439.410806 | Name | Value |
| | Z_0 | Gaussian(8.575, 2.977) |
| | Z_1 | Gaussian(-24.13, 22.58) |
| | Z_2 | Gaussian(-42.48, 47.41) |
| | Z_3 | Gaussian(29.61, 3.99) |
| | Z_4 | Gaussian(22.18, 1.996) |
| | Z_5 | Gaussian(-0.2548, 5.846) |
| | Z_6 | Gaussian(-155.8, 7.302) |
| | v__2 | Gamma(301.6, 1.033e-06)[mean=0.0003115] |

Da

X1

X2

X3

X4

X5

X6

Z

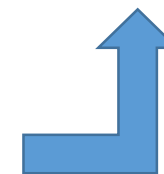
Y

real input

real input

real latent $\sim X1 + X2 + X3 + X4 + X5 + X6$

bool output Z > 0.0



A **link function** maps from a **continuous** regression to a **discrete** output

Less Boilerplate for Stan?



```
model {  
  real p;  
  q ~ 1 + female + black + female:black  
    + age + edu + age:edu +  
    (1{(1|region) + v_prev} | state) ;  
  p <- fmax(0, fmin(1, q));  
  y ~ bernoulli(p);  
}
```

```
p[i] ~ fmax(0, fmin(1, inv_logit(v_0  
  + b_female*female[i] + b_black*black[i]  
  + b_female_black*female[i]*black[i]  
  + b_age[age[i]] + b_edu[edu[i]]  
  + b_age_edu[age[i],edu[i]] + b_hat[state[i]])));  
  
y ~ bernoulli(p);  
}
```

What else is in the paper?

- Type system and a type-preserving formal semantics.
- Nested models may be eliminated – every regression normalizes to the form: $(\vec{v}\alpha)(P + N)$

Classes of Terms: N, P

| | |
|---|-------------------------|
| $N ::= \sum_{i=1}^n D_i(u_{i1}, \dots, u_{i D_i })$ | noise |
| $P ::= \sum_{i=1}^n v_i \{ \alpha_i \sim N_i \} \mid \vec{w}_i$ | single-level regression |

- We explain the essence of the popular formula notation in R's **lm** and **lmer** packages by converting formulas to the regression calculus.

Dimensions and Cube-Expressions:

Let a *dimension*, \vec{e} or \vec{f} , be a finite list of natural numbers.

Let a *cube-expression with dimension* $\vec{e} = [e_1; \dots; e_n]$ be a phrase that denotes a multi-dimensional array of some type $T[e_n] \dots [e_1]$.

An *index* for \vec{e} is a list $[i_1; \dots; i_n]$ with $0 \leq i_j < e_j$ for each j .

$\Gamma; \vec{e} \vdash_{\rightarrow} v : T$ predictor v yields an \vec{e} -cube of T

Typing Rules for Predictors:

| | | |
|--|---|--|
| (SCALAR) | (VAR) | (INTERACT) |
| $\frac{\Gamma \vdash \diamond \quad s \in \mathbb{R}}{\Gamma; \vec{e} \vdash s : \mathbf{real}}$ | $\frac{\Gamma \vdash x : T[\vec{e}]}{\Gamma; \vec{e} \vdash x : T}$ | $\frac{\Gamma; \vec{e} \vdash u : \mathbf{real} \quad \Gamma; \vec{e} \vdash v : \mathbf{real}}{\Gamma; \vec{e} \vdash (u : v) : \mathbf{real}}$ |

(PATH)

$$\frac{\Gamma; \vec{e} \vdash u_i : \mathbf{mod}(f_i) \quad \forall i \in 1..n \quad \Gamma; \vec{f} \vdash v : T}{\Gamma; \vec{e} \vdash (u_1, \dots, u_n).v : T}$$

$\Gamma; \vec{e}; \vec{f} \vdash r ! \Pi$ regression r yields \vec{e} -cube with parameter \vec{f} -cubes

Typing Rules for Regressions:

(NOISE)

$$\frac{D : (x_1 : U_1, \dots, x_n : U_n) \rightarrow \mathbf{real} \quad \Gamma; \vec{e} \vdash u_j : U_j \quad \forall j \in 1..n}{\Gamma; \vec{e}; \vec{f} \vdash D(u_1, \dots, u_n) ! \emptyset}$$

(SUM)

$$\Gamma; \vec{e}; \vec{f} \vdash r ! \Pi$$
$$\frac{(\Gamma, \Pi); \vec{e}; \vec{f} \vdash r' ! \Pi'}{\Gamma; \vec{e}; \vec{f} \vdash r + r' ! (\Pi, \Pi')}$$

(GROUP)

$$\Gamma; \vec{e} \vdash v : \mathbf{mod}(f)$$
$$\frac{\Gamma; \vec{e}; (f :: \vec{f}) \vdash r ! \Pi}{\Gamma; \vec{e}; \vec{f} \vdash r \mid v ! \Pi}$$

(COEFF)

$$\frac{\Gamma; \vec{e} \vdash v : \mathbf{real} \quad \Gamma; \vec{f}; [] \vdash r ! \Pi \quad \alpha \notin \text{dom}(\Gamma, \Pi)}{\Gamma; \vec{e}; \vec{f} \vdash v\{\alpha \sim r\} ! (\Pi, \alpha : \mathbf{real}[\vec{f}])}$$

Translation of Predictors to Fun: $\llbracket v \rrbracket \vec{E} = E$

$$\llbracket s \rrbracket \vec{E} \triangleq s$$

$$\llbracket x \rrbracket \vec{E} \triangleq x[\vec{E}]$$

$$\llbracket u : v \rrbracket \vec{E} \triangleq \llbracket u \rrbracket \vec{E} \times \llbracket v \rrbracket \vec{E}$$

$$\llbracket (u_1, \dots, u_n).v \rrbracket \vec{E} \triangleq \llbracket v \rrbracket [\llbracket u_1 \rrbracket \vec{E}] \dots [\llbracket u_n \rrbracket \vec{E}]$$

Lemma 2. *If $\Gamma; (e_i)^{i \in I} \vdash v : T$ and $\Gamma \vdash E_i : \mathbf{mod}(e_i)$ for all $i \in I$ then $\Gamma \vdash \llbracket v \rrbracket (E_i)^{i \in I} : T$.*

Translation of Regressions to Fun: $\llbracket r \rrbracket \vec{e} \vec{f} \vec{F} = E$

$$\begin{aligned}
\llbracket D(u_1, \dots, u_n) \rrbracket \vec{e} \vec{f} \vec{F} &\triangleq ((), [\mathbf{for} \vec{z} < \vec{e} \rightarrow D(\llbracket u_1 \rrbracket \vec{z}, \dots, \llbracket u_n \rrbracket \vec{z})]) \\
\llbracket v\{\alpha \sim r\} \rrbracket \vec{e} \vec{f} \vec{F} &\triangleq \mathbf{let} (\text{dom}(r), \alpha) = \llbracket r \rrbracket \vec{f} [] [] \mathbf{in} \\
&\quad (\text{dom}(r) @ [\alpha], [\mathbf{for} \vec{z} < \vec{e} \rightarrow \llbracket v \rrbracket \vec{z} \times \alpha[\vec{F}[\vec{z}]]]) \\
\llbracket r + r' \rrbracket \vec{e} \vec{f} \vec{F} &\triangleq \mathbf{let} (\text{dom}(r), y) = \llbracket r \rrbracket \vec{e} \vec{f} \vec{F} \mathbf{in} \\
&\quad \mathbf{let} (\text{dom}(r'), y') = \llbracket r' \rrbracket \vec{e} \vec{f} \vec{F} \mathbf{in} \\
&\quad (\text{dom}(r) @ \text{dom}(r'), [\mathbf{for} \vec{z} < \vec{e} \rightarrow y[\vec{z}] + y'[\vec{z}]]) \\
\llbracket r \mid v \rrbracket \vec{e} \vec{f} \vec{F} &\triangleq \llbracket r \rrbracket \vec{e} (f :: \vec{f}) (F :: \vec{F}) \quad \text{where} \\
&\quad \Gamma; \vec{e} \vdash v : \mathbf{mod}(f) \text{ and } F = [\mathbf{for} \vec{z} < \vec{e} \rightarrow \llbracket v \rrbracket \vec{z}] \\
\llbracket (v\alpha)r \rrbracket \vec{e} \vec{f} \vec{F} &\triangleq \mathbf{let} (\text{dom}(r), y) = \llbracket r \rrbracket \vec{e} \vec{f} \vec{F} \mathbf{in} (\text{dom}(r) \setminus \alpha, y)
\end{aligned}$$

Theorem 1 (Type Preservation). *If $\Gamma; \vec{e}; (f_j)^{j \in J} \vdash r ! \Pi$ and $\Gamma \vdash F_j : \mathbf{mod}(f_j)[\vec{e}]$ for all $j \in J$, and $E = \llbracket r \rrbracket \vec{e} (f_j)^{j \in J} (F_j)^{j \in J}$, then we have $\Gamma \vdash E : \text{tuple}(\Pi) \times \mathbf{real}[\vec{e}]$.*

Summary of Fabular

- The **regression calculus** is the first formal calculus of regressions, with a unique recursive syntax for hierarchical models (written $v\{\alpha \sim r\}$), going beyond the formulas of R.
- Regressions are boilerplate components of many models.
- **Fabular** mixes potentially vectorised regressions with latent variables and link functions, scrapping some boilerplate.
- We envisage that boilerplate in other languages, such as Stan, can also be eliminated.
- Synthesis of probabilistic programs in Fabular?

[**https://aka.ms/tabular**](https://aka.ms/tabular)

BDA3

Gelman
Carlin
Stern
Dunson
Vehtari
Rubin



The BUGS Book

Lunn, Jackson, Best,
Thomas, and Spiegelhalter



APPLIED REGRESSION MODELING

SECOND
EDITION



Machine Learning

Murphy



BAYESIAN REASONING and MACHINE LEARNING

CAMBRIDGE

Monte Carlo Statistical Methods

Second
Edition



Statistical Rethinking

McElreath



Bayesian Methods

Gill



53

APPLICATIONS
OF MATHEMATICS

Glasserman
Monte Carlo

JAYNES

Probability
Theory

PROBABILISTIC GRAPHICAL

PEARL

CAUSAL

Mackay

Information
Theory,
Information

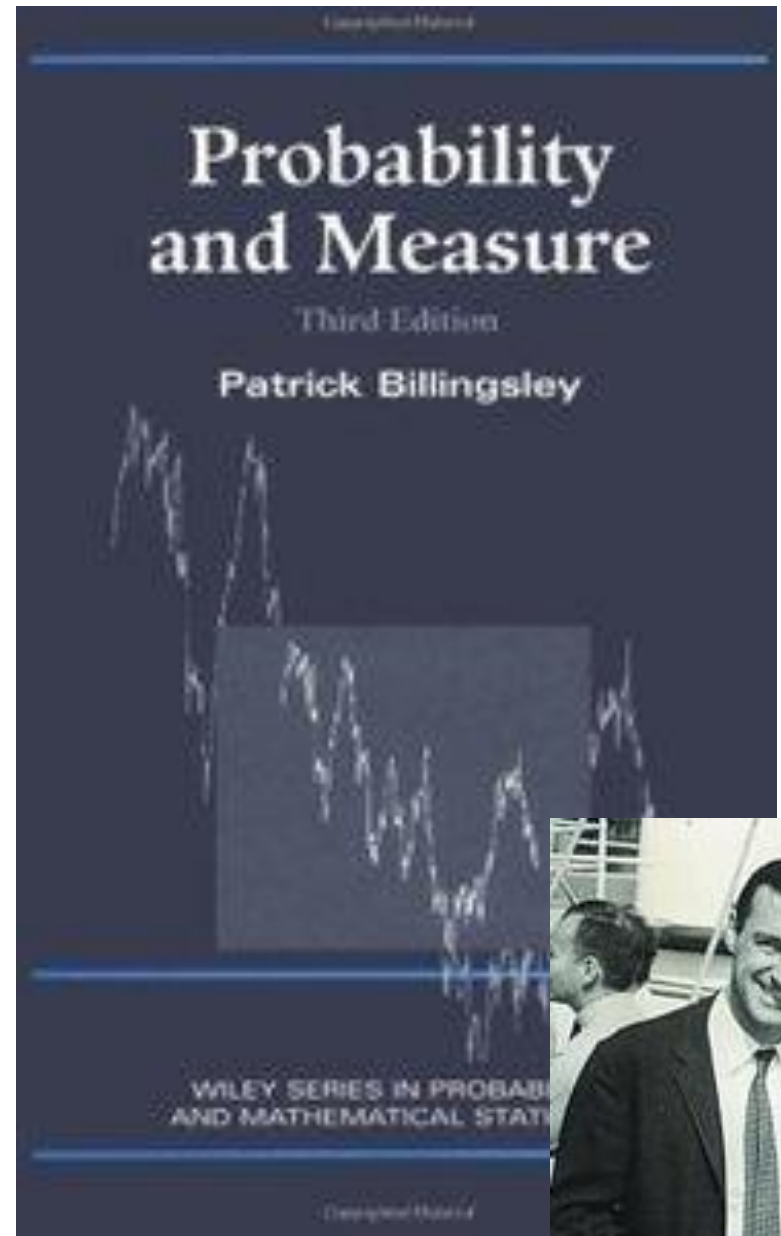
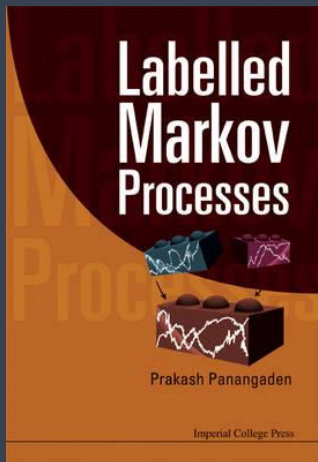
BISHOP



PATTERN
AND MACHINE

5

Mathematical Statistics



4

Probability Theory

The Logic of Science

E. T. JAYNES

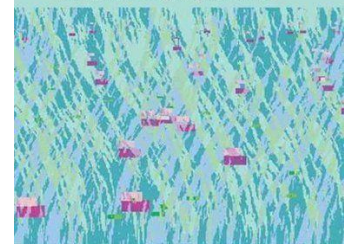


CAMBRIDGE

Statistical Physics

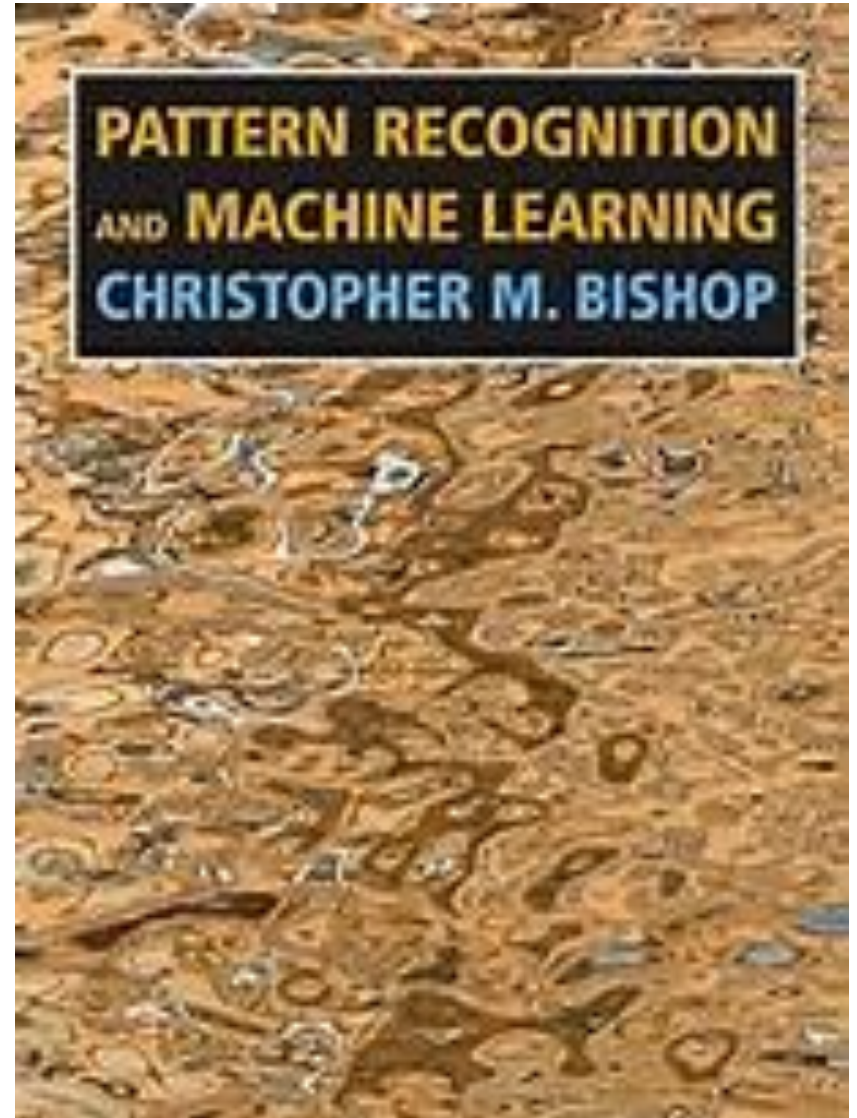
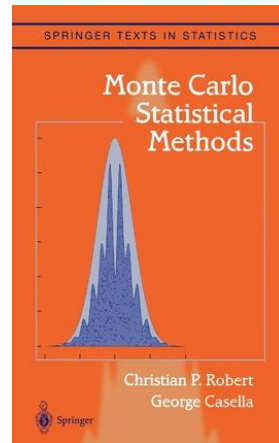
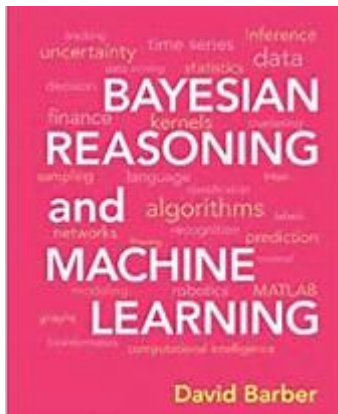
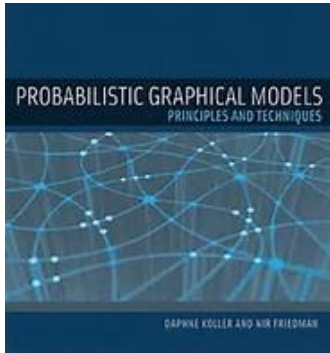
David J. C. Mackay

**Information Theory, Inference,
and Learning Algorithms**



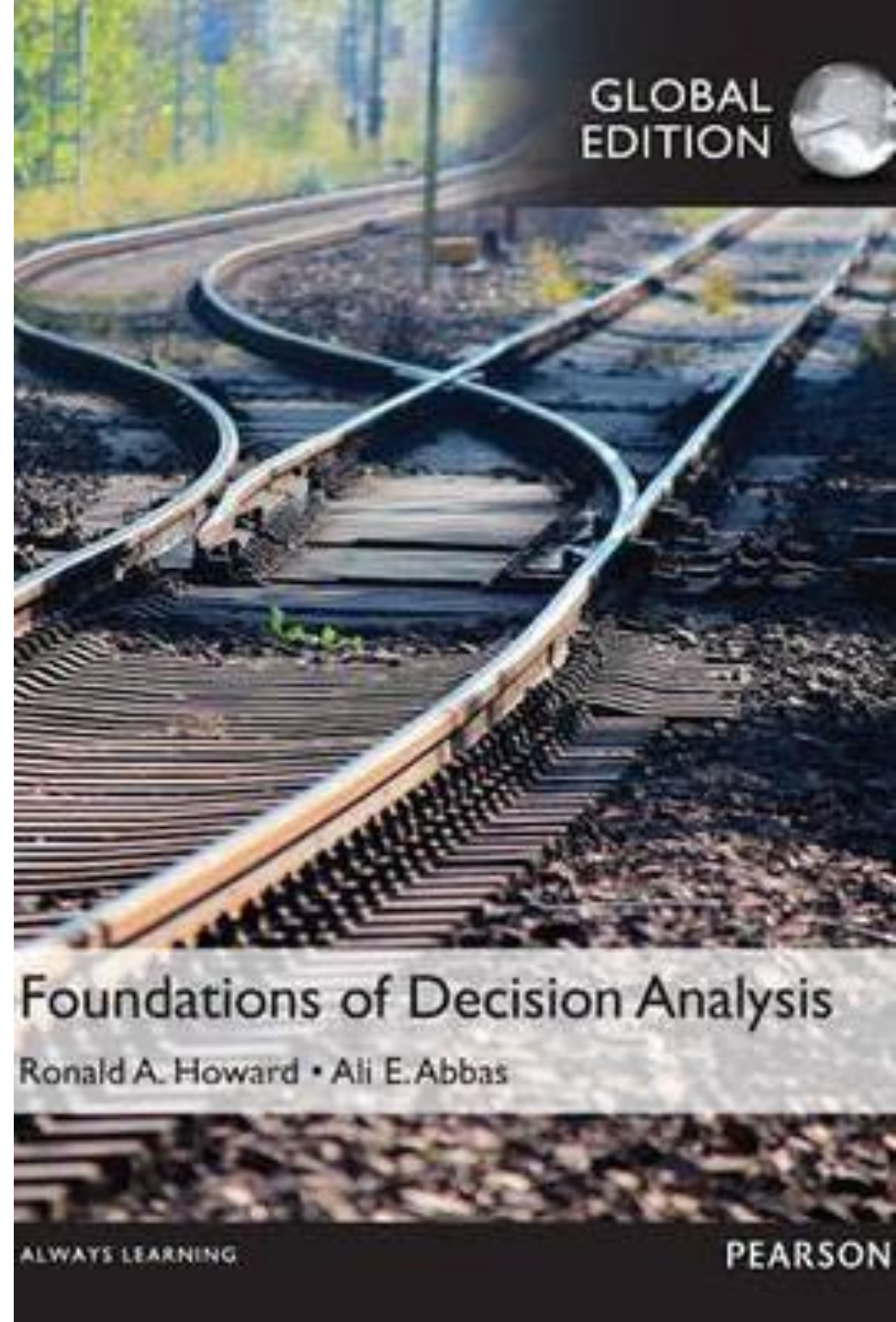
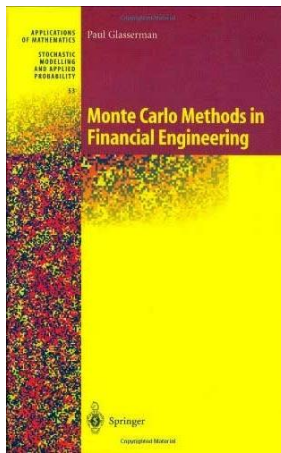
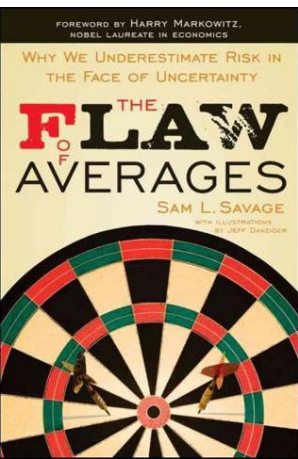
CAMBRIDGE

3 Machine Learning



2

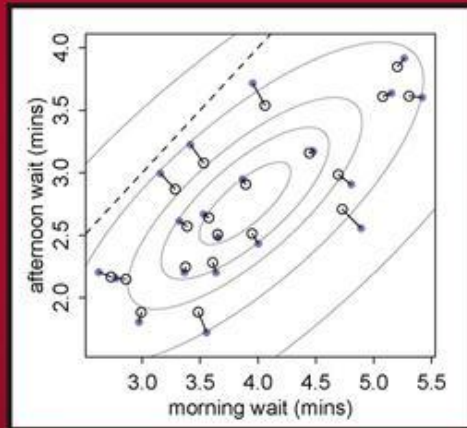
Business and Finance



Texts in Statistical Science

Statistical Rethinking

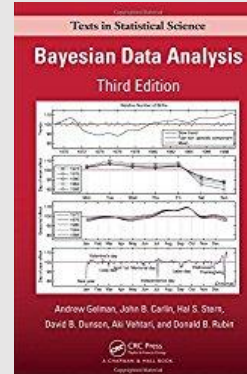
A Bayesian Course with Examples in R and Stan



Richard McElreath

 **CRC Press**
Taylor & Francis Group
A CHAPMAN & HALL BOOK

Bayesian Statistics



Language-Based Statistical Thinking

- "Statistical thinking will one day be as necessary for efficient citizenship as the ability to read and write." H.G. Wells
- Tabular seeks to empower today's spreadsheet users to be efficient citizens, by casting statistical thinking as a PL question
- Probabilistic programming as a field empowers developers to more easily build custom machine-learning solutions
- Many research challenges:
 - Better model suggestion – automatically discover suitable statistical model for dataset
 - Better model criticism – how to convey uncertainty?
 - Better inference – exciting recent progress using Deep Neural Nets to speed up Monte Carlo inference for probabilistic programs

<https://aka.ms/tabular>