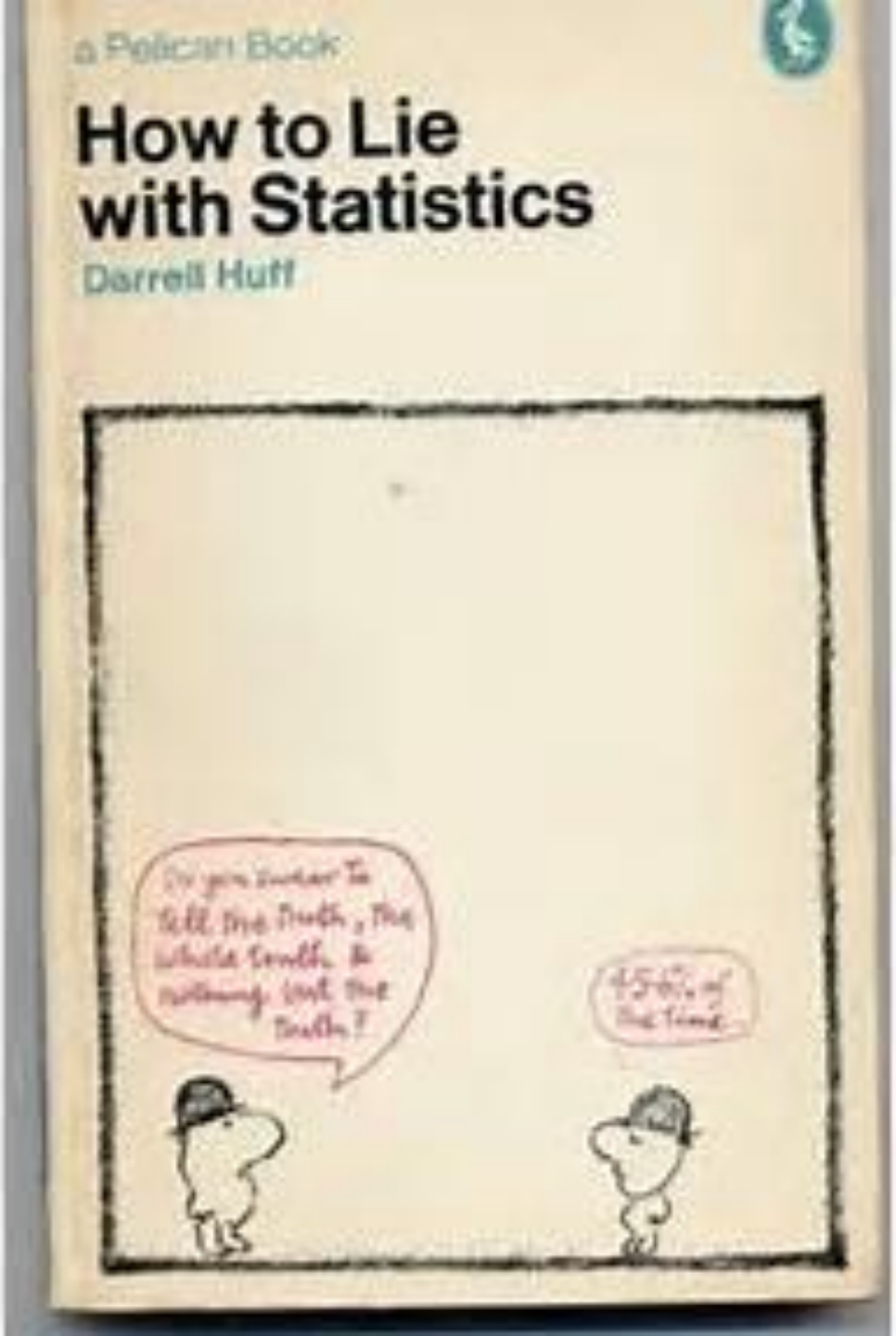


"Statistical thinking will one day be as necessary for efficient citizenship as the ability to read and write."
H.G. Wells

Empowering Spreadsheet Users with Probabilistic Programming

Andrew D. Gordon
(MSR and Edinburgh)

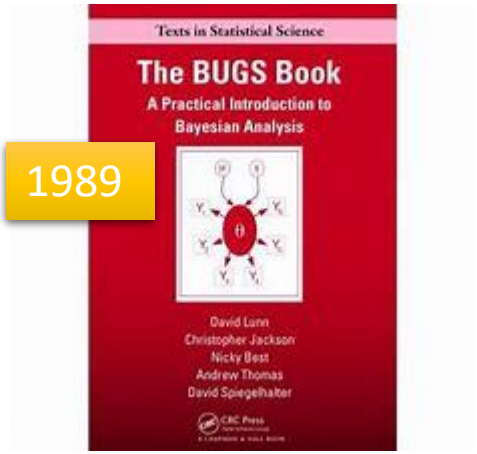
#OPLSS, Eugene, Oregon

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)

@AndrewDGordon
aka.ms/CalcIntel
aka.ms/popl

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, Gen, ...
- Academic prizes (eg in computer vision), and large-scale adoption in some commercial services (eg Microsoft Office Clutter, TrueSkill, etc)
- Semantics pioneered by Kozen, Giry, Jones/Plotkin, Panangaden et al, Ramsey/Pfeffer



1989

infer.net

2004

Church: a language for generative models

Noah D. Goodman, Vikash K. Mansinghka,
Daniel M. Roy, Keith Bonawitz & Joshua B. Tenenbaum
MIT BCS/CSAIL
Cambridge, MA 02139

2007

Stan

2010?

Probabilistic Programming Languages since 1989

Plan

- Monday:
  Probabilistic Programming in Infer.NET and Tabular
- Tuesday:
  Probabilistic Programming in Fabular and Stan
- Wednesday:
  End-User Probabilistic Programming

- Purpose:
  - Why we are doing this? Who are we doing it for?
  - An intuition of how we are doing this
  - Some challenges for the future

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?

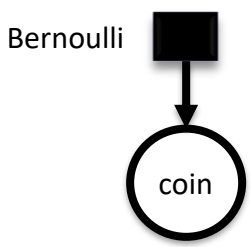
How do we specify the model?

- 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

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

(Bernoulli distribution)



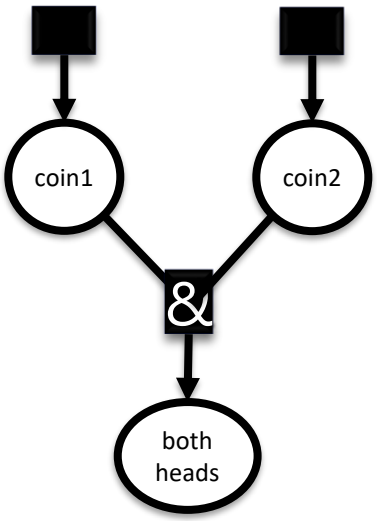

```
bool coin  = random()<0.5;
```

(Factor graph)

(Simulator)

Model for two coins problem

$$p(coin_1)p(coin_2)p(bothHeads|coin_1, coin_2)$$



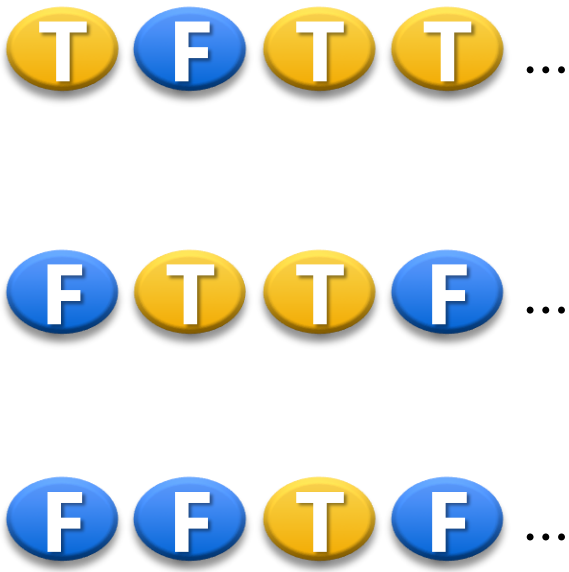

```
bool coin1  = random()<0.5;
bool coin2 = random()<0.5;
bool bothHeads  = coin1 & coin2;
```

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;

bool coin2 =
  random()<0.5;

bool bothHeads =
  coin1 & coin2;
```

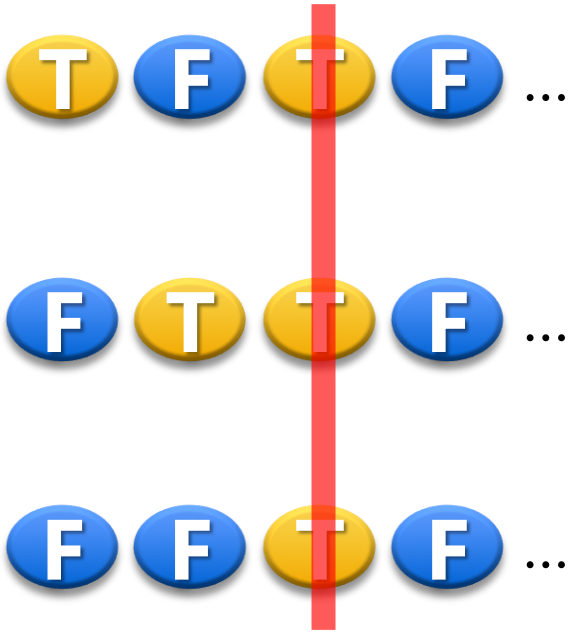
T ~50%
F ~50%

T ~50%
F ~50%

T ~25%
F ~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;

bool coin2 =
  random()<0.5;

bool bothHeads =
  coin1 & coin2;
```

T ~33%
F ~67%

T ~33%
F ~67%

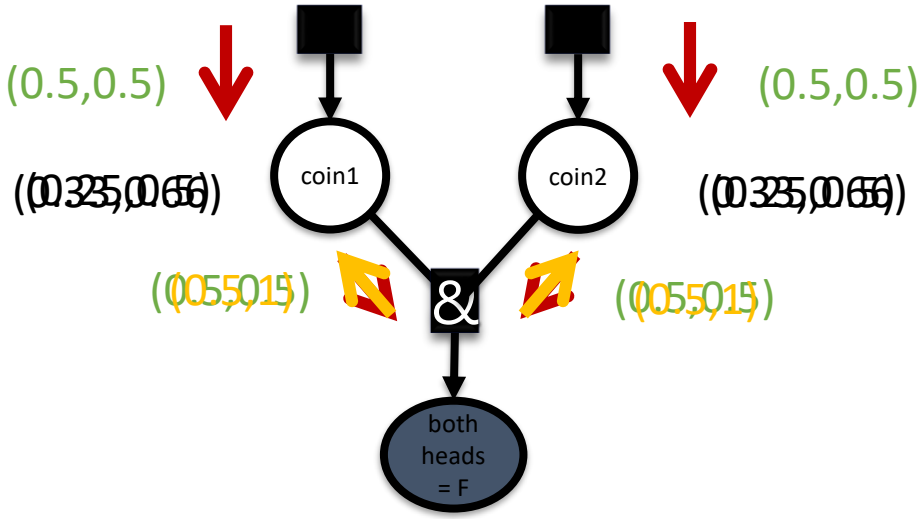
T ~0%
F ~100%

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

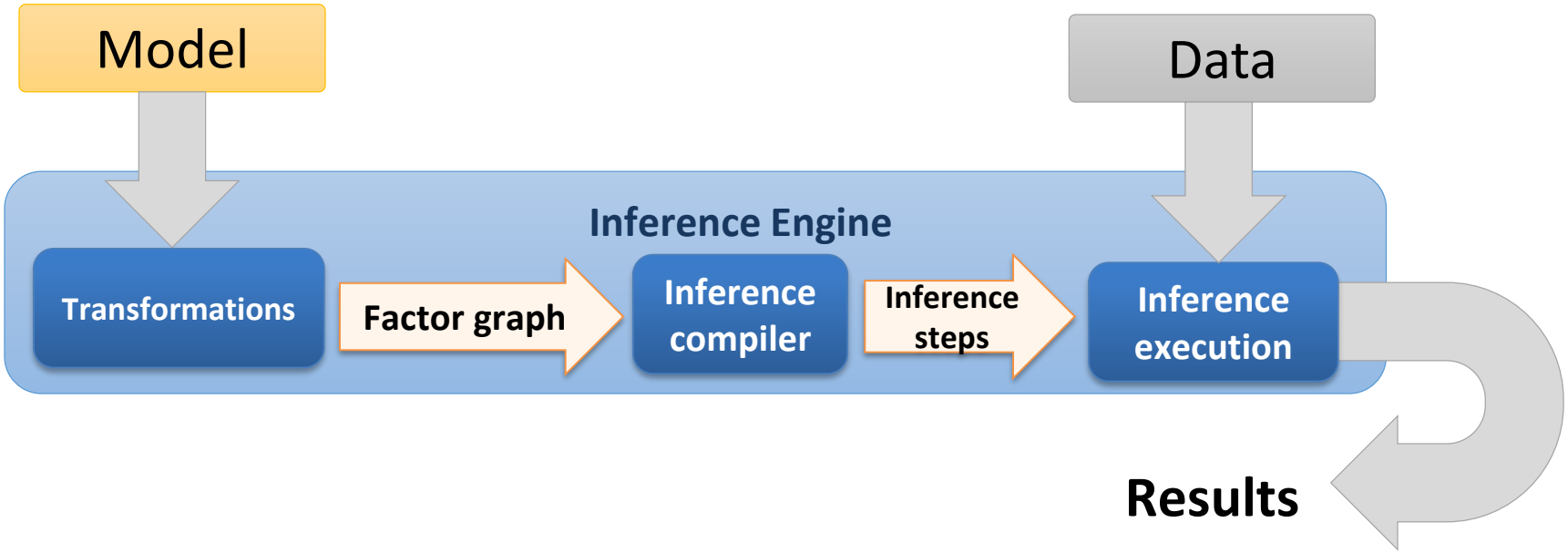
infer.net

https://dotnet.github.io/infer/

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

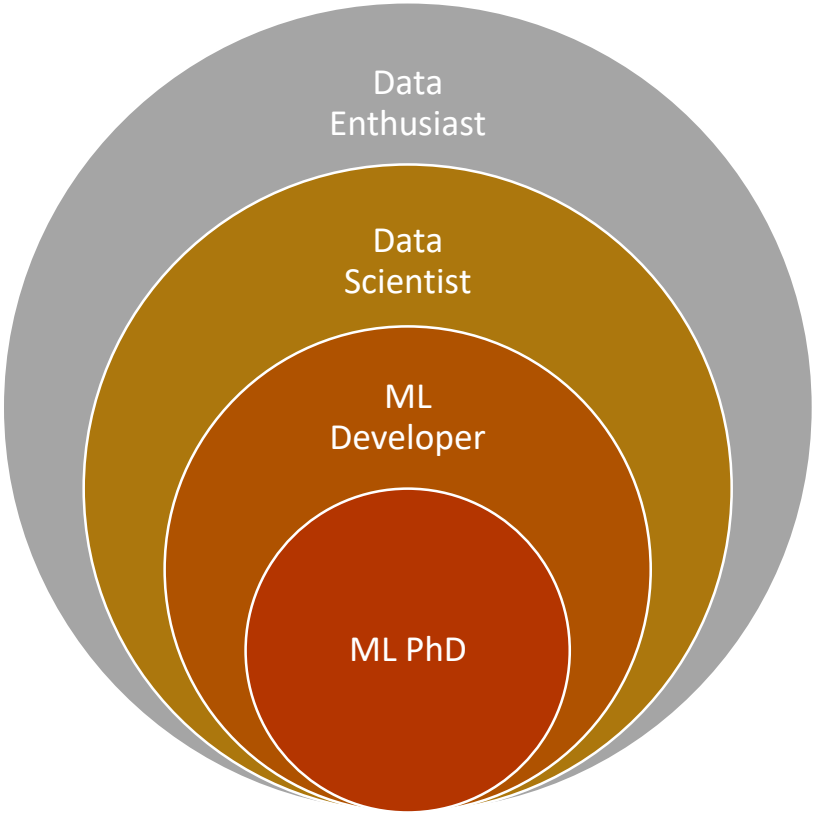
We *observe* that bothHeads is F

```
bothHeads.ObservedValue = false;

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

// ‘dist1’ is Bernoulli(0.333…)
```

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

Tabular

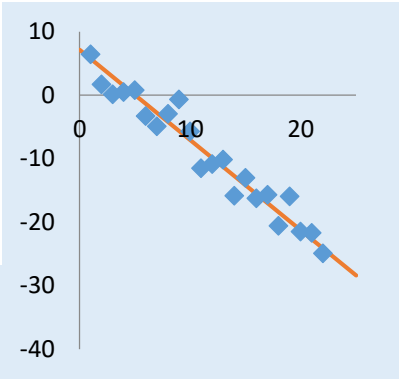


- **DSL to get insight from data with probabilistic programs**

- 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

Papers at POPL'14, ESOP'15, POPL'16

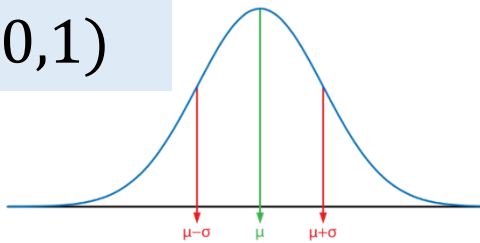
Structure of a Model



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)
B	real	static latent	Gaussian(0,1)
x	real	input	
y	real	output	Gaussian(A*x+B,4)

Parameter distribution $p(A, B)$

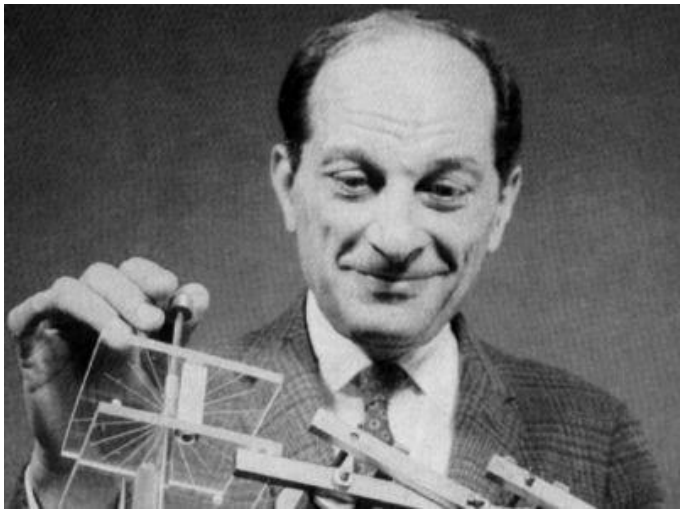
Sampling distribution $p(\vec{y} \mid A, B, \vec{x})$ (aka the **likelihood**)

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

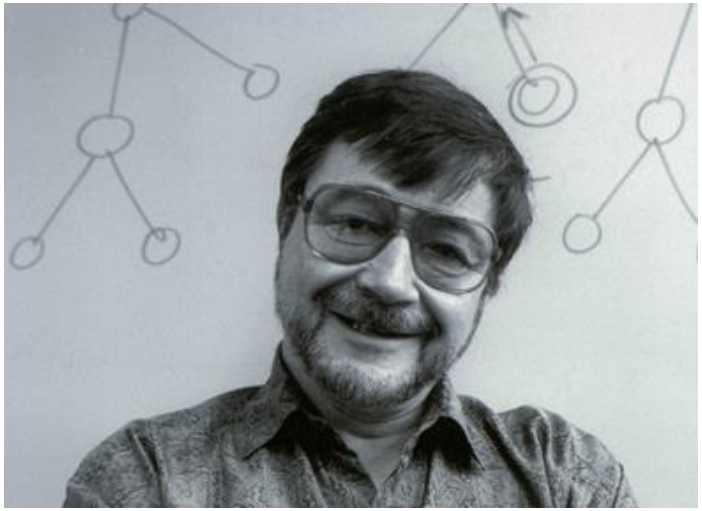
- Monte Carlo



Stanisław Ulam

- Set of samples for joint
- Slow but accurate

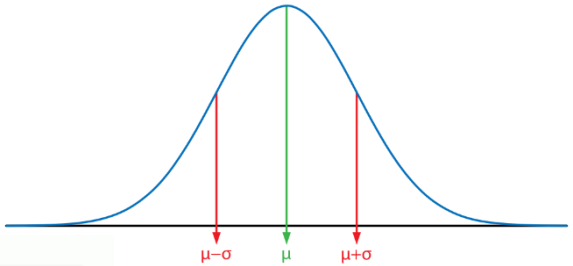
- Message Passing

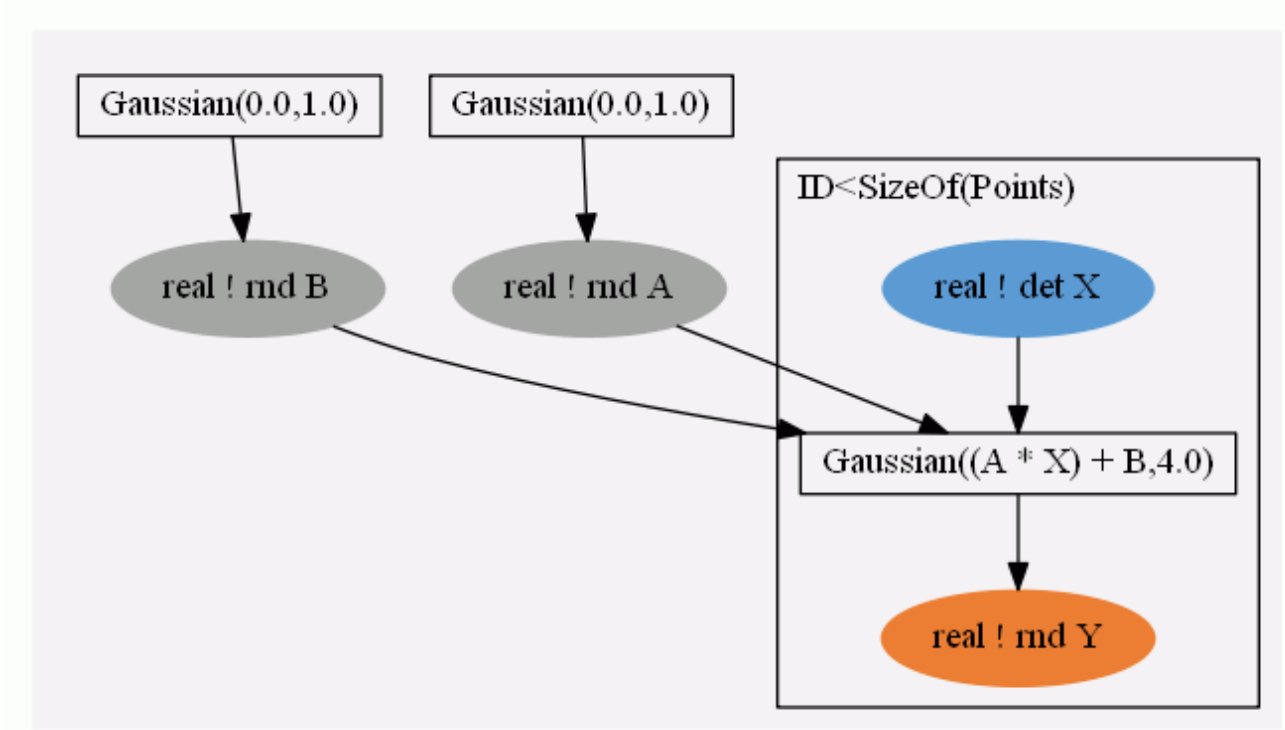


Judea Pearl

- Parameters for marginals
- Fast but approximate

Factor Graphs





Sample

Infer

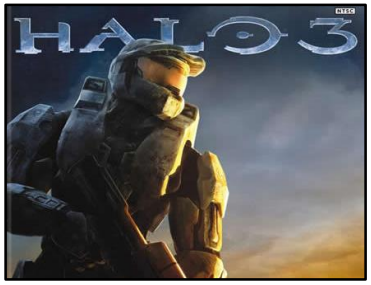
$$p(A, B, \vec{x}, \vec{y}) \triangleq pdf\big(A, Gaussian(0,1)\big) \times pdf\big(B, Gaussian(0,1)\big) \times \prod_i pdf\big(y_i, Gaussian(Ax_i + B, 4)\big)$$

The Tabular Recipe

1. Start with the schema
2. Add latent columns
3. Write models for latent and output columns
4. Learn latent columns and table parameters, and/or
5. Predict missing values in output columns

- We focus on this part of the cycle of learning from data, leaving gathering, preprocessing, visualizing to other tools

TrueSkill – Data

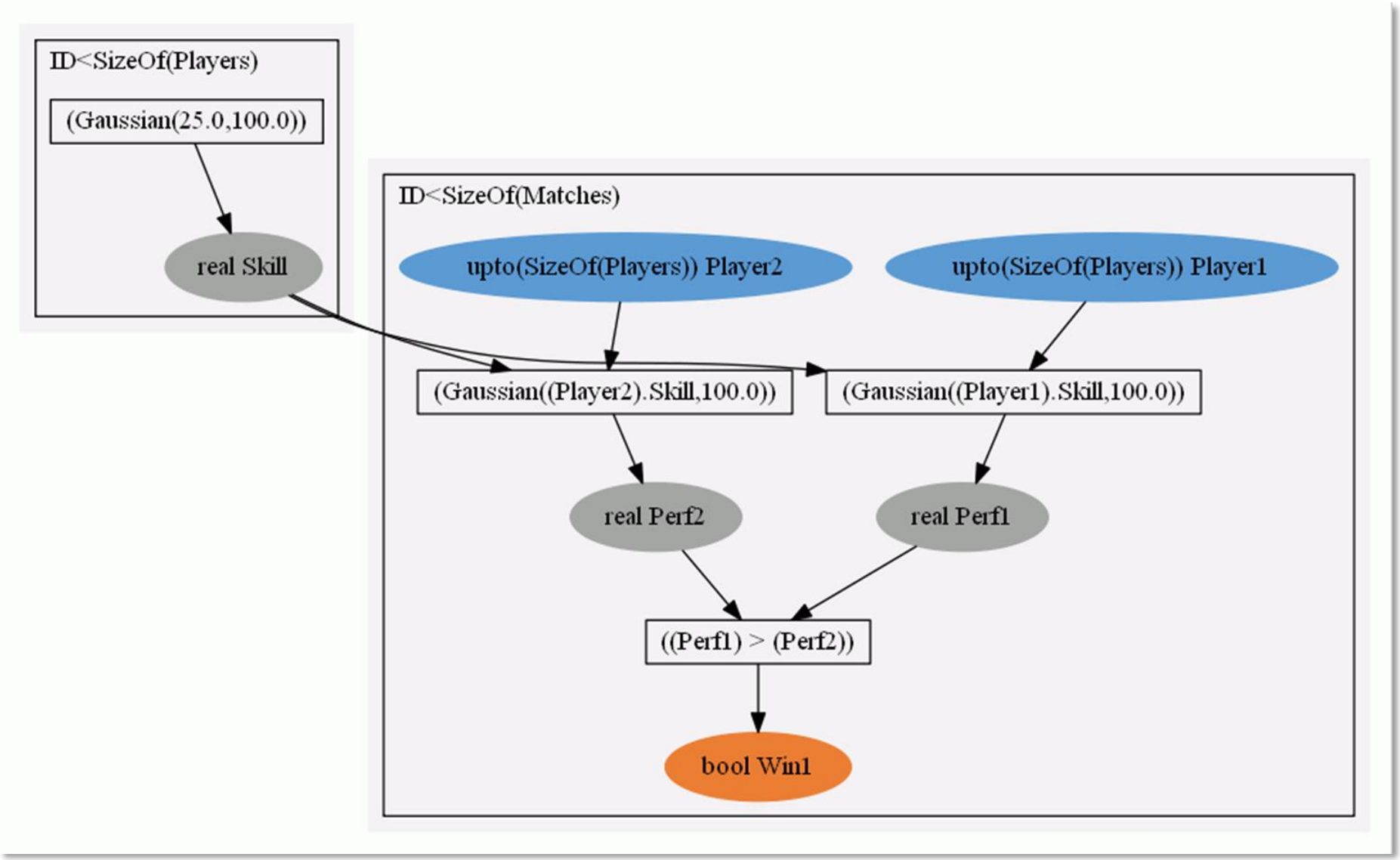


Players	
Name	String

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


ID<SizeOf(Players)
(Gaussian(25.0,100.0))
real Skill
ID<SizeOf(Matches)
upto(SizeOf(Players)) Player2
upto(SizeOf(Players)) Player1
(Gaussian((Player2).Skill,100.0))
(Gaussian((Player1).Skill,100.0))
real Perf2
real Perf1
((Perf1) > (Perf2))
bool Win1

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	

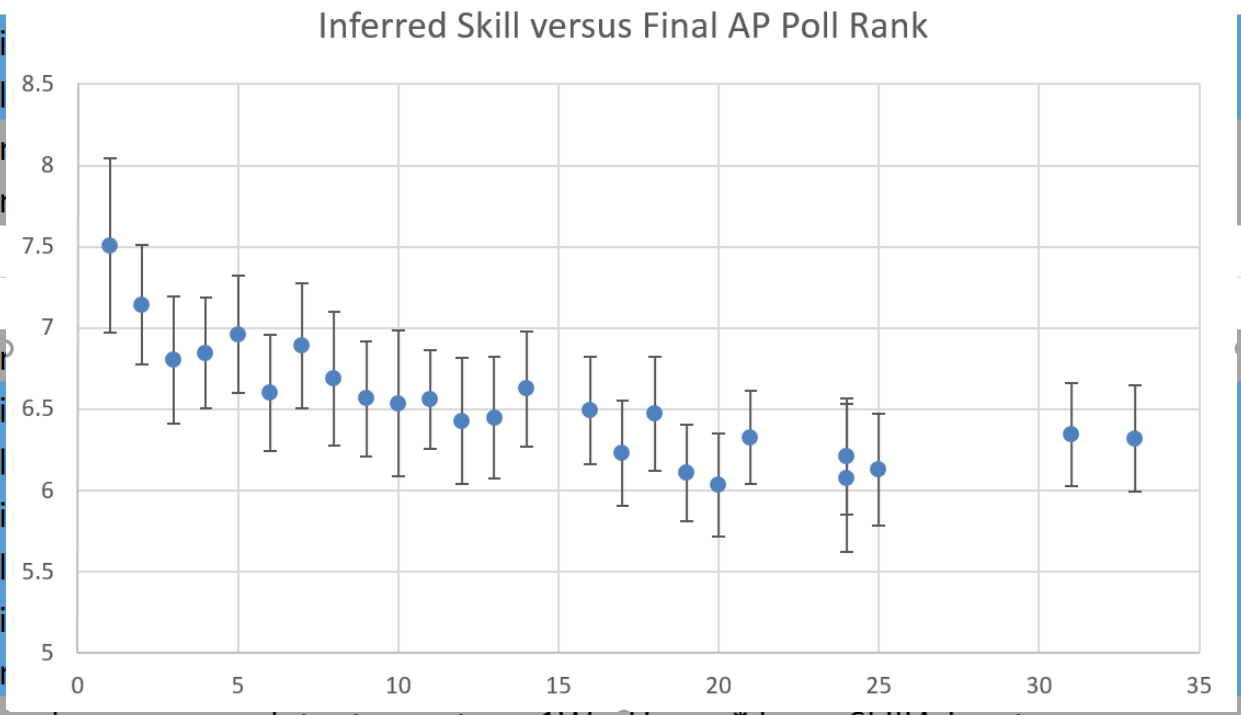
	Player1	Player2	Win1
0	0	1	False
1	1	2	False
2	0	2	Bernoulli(0.31)

NCAAF.xlsx - Excel

File Home Insert Page Layout Formulas Data Review View Tabular Tell me what you want to do... Andy Gordon (RESEARCH) Share

A30

	A	B	C	D
1	**Leagues**			
2	leagueId	int	input	
3	skillModifier	real	latent	Gaussian(0.0, 1.0)
4				
5	**Teams**			
6	teamId	i		
7	leagueId	I		
8	individualSkill	r		
9	skill	r		
10				
11	**Matches**			
12	**homeSkillAdvantage**	r		
13	gameId	i		
14	team1Id	I		
15	team1Score	i		
16	team2Id	I		
17	team2Score	i		
18	team1WasHome	r		
19	team1HomeAdvantage	real	latent	team1WasHome * homeSkillAdvantage
20	team1Perf	real	latent	Gaussian(team1Id.skill + team1HomeAdvantage, 0.25)
21	team2Perf	real	latent	Gaussian(team2Id.skill, 0.25)
22	team1Won	bool	output	team1Perf > team2Perf



... teams games Tabular_NCAAF_TrueSkill Team Visualization (L+H) Comp ...

Ex: Cold Boot Attacks

This paper was released February 21, 2008 and published in *Proc. 2008 USENIX Security Symposium*. For the most recent revision, related source code, and videos of demonstration attacks, visit http://citp.princeton.edu/memory.

Lest We Remember: Cold Boot Attacks on Encryption Keys

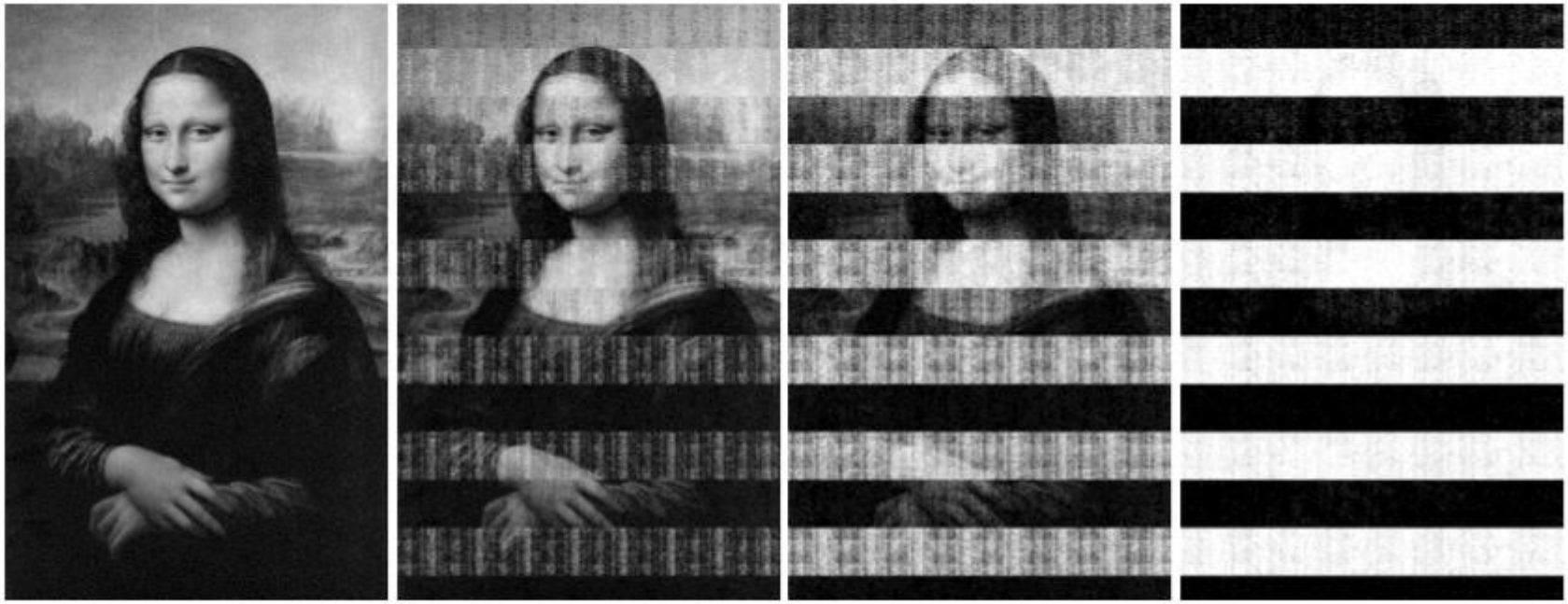
J. Alex Halderman*, Seth D. Schoen†, Nadia Heninger*, William Clarkson*, William Paul‡, Joseph A. Calandrino*, Ariel J. Feldman*, Jacob Appelbaum, and Edward W. Felten*

*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

- 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.


Esc
F1
F2
Tab
F5
6:48 29
GO Walkin

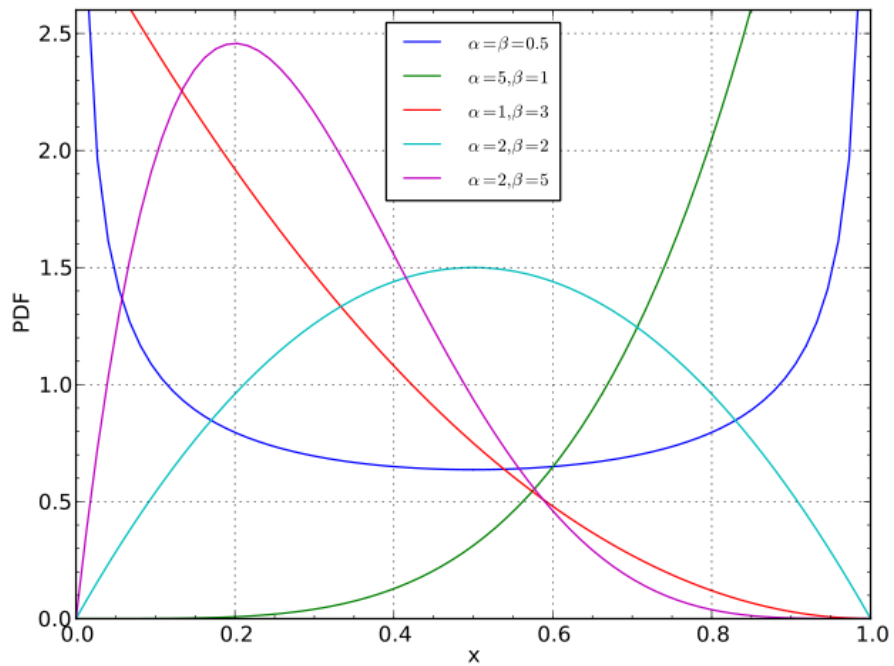
Recovering DES Keys

- 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 Decay11
  - the probability of a 0 flipping to a 1 is some fixed Decay01.
- 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.



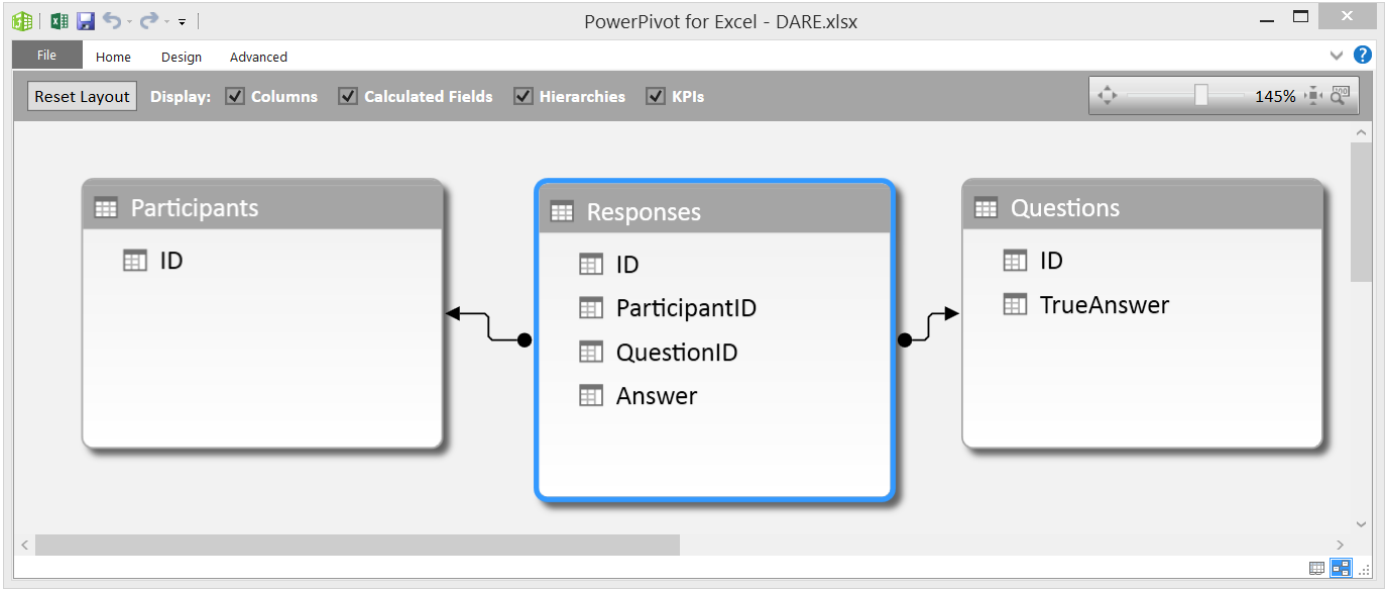
Source: Wikipedia

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)

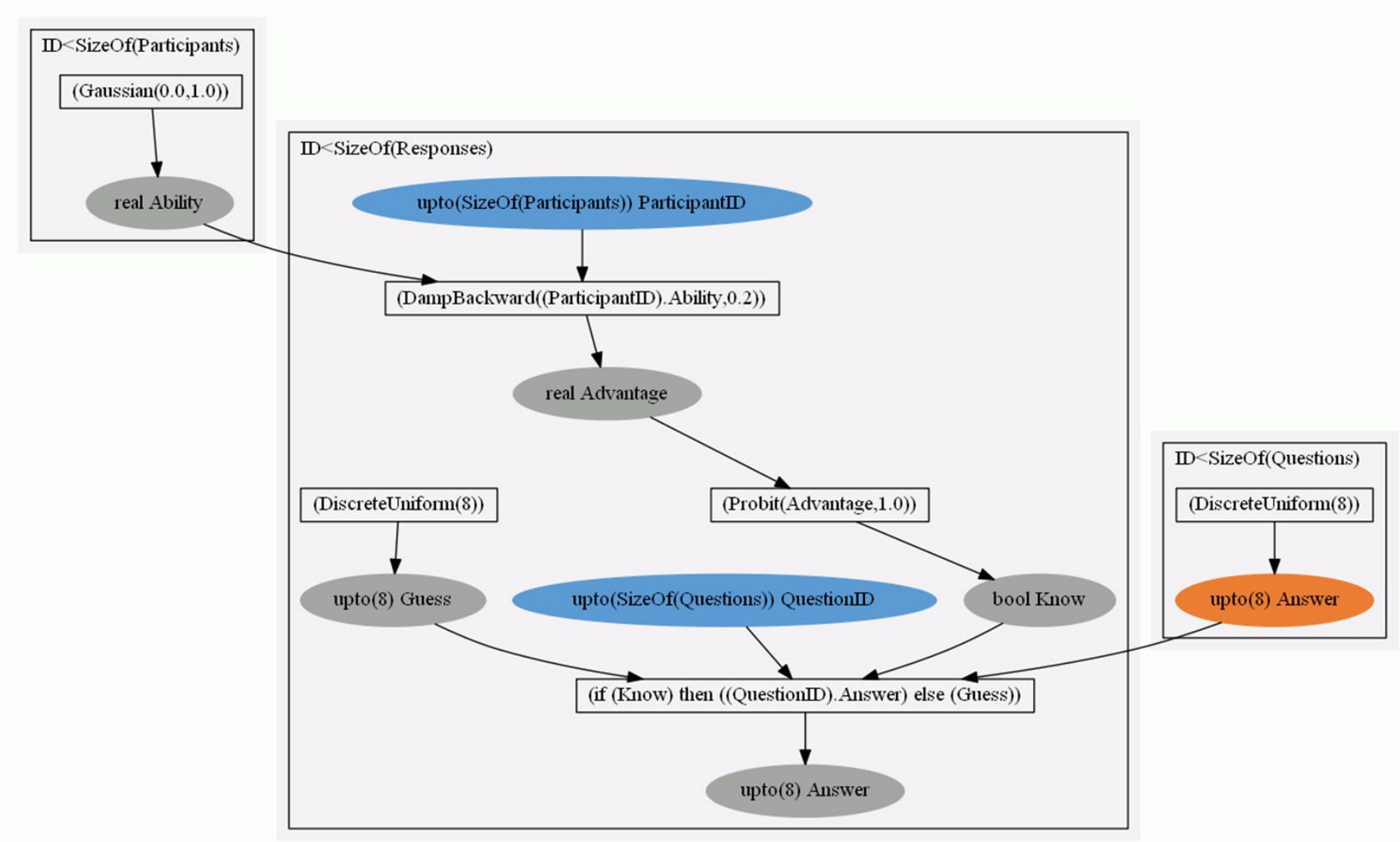


- Data drawn from multiple choice intelligence test
- 60 questions with 8 possible answers (one correct)
- 121 participants
- Responses relate ParticipantID and QuestionID to Participant's answers.

Log evidence -7499.92

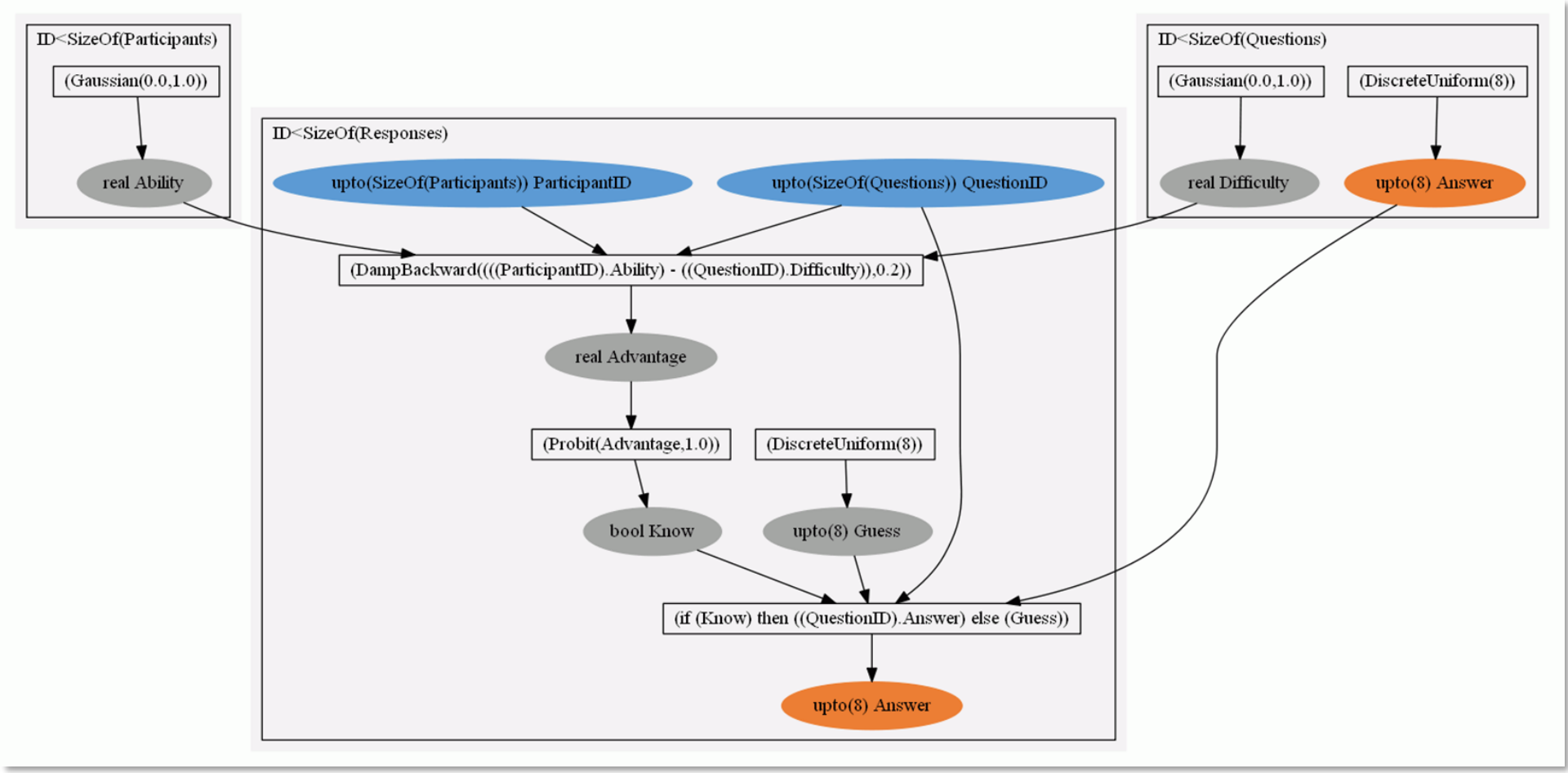
Ability

Probit(a,b) = Gaussian(a,b)>0.0



Log evidence -5932.34

Difficulty/Ability



Case study: Psychometrics

```
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");
```

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)
Guess	Int	Latent	DiscreteUniform(8)
Response	Int	Output	if Know then QuestionID.Answer else Guess

```
    NTrainingQuestions.ObservedValue = nTrainingQuestions;
    ParticipantOfResponse.ObservedValue = participantOfResponse;
    QuestionOfResponse.ObservedValue = questionOfResponse;
    TrainingResponseIndices.ObservedValue = trainingResponseIndices;
    ObservedResponseAnswer.ObservedValue = Util.ArrayInit(nTrainingResponses, i =>
                        response[trainingResponseIndices[i]]);
    TrainingQuestionIndices.ObservedValue = trainingQuestionIndices;
    ObservedQuestionAnswer.ObservedValue = Util.ArrayInit(nTrainingQuestions, i =>
                        answer[trainingQuestionIndices[i]]);
    var Engine = new InferenceEngine()
    { ShowTimings = true,ShowWarnings = false,ShowProgress = false,NumberOfIterations = 10 };
    var AnswerOfResponsePosterior = Engine.Infer<Discrete[]>(AnswerOfResponse);
    var AnswerOfQuestionPosterior = Engine.Infer<Discrete[]>(AnswerOfQuestion);
    var LogEvidence = Engine.Infer<Bernoulli>(Evidence).LogOdds;
    var AbilityPosterior = Engine.Infer<Gaussian[]>(Ability);
    var DifficultyPosterior = Engine.Infer<Gaussian[]>(Difficulty);
    var DiscriminationPosterior = Engine.Infer<Gamma[]>(Discrimination);
}
```

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

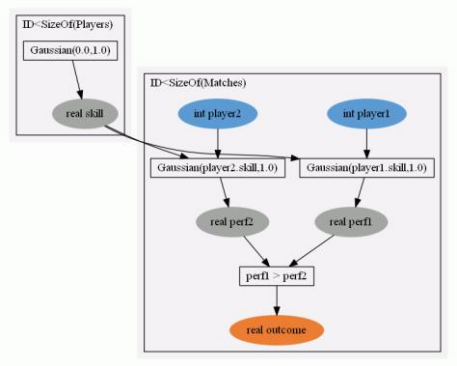
Core Syntax of Schemas

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

level	ℓ ::= **static** | ϵ
annotation	A ::= **input** | **latent**(E) | **output**(E)
expression	E ::= c | $E.c$ | $\mathrm{f}(E_1,\ldots,E_n)$ | $\mathrm{D}(E_1,\ldots,E_n)$| …

- 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 ~ 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 ~ Gaussian(Players[Player1].Skill,100)
Perf2 ~ 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

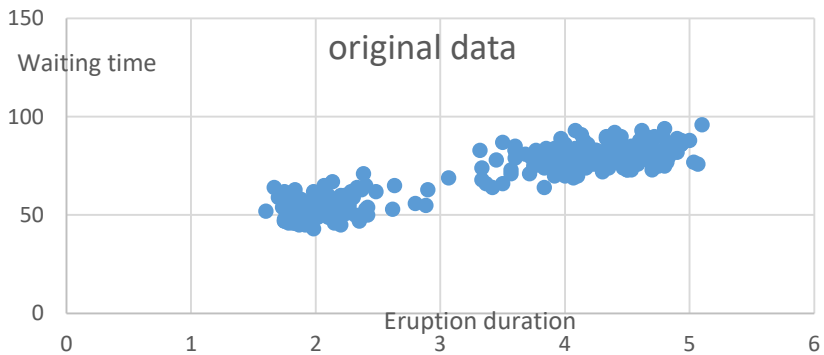
Beyond the Core Syntax

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

level	ℓ ::= **static** | ϵ
annotation	A ::= **input** | **latent**(M) | **output**(M)
model expression	M ::= E | $t(h_1{=}E_1,\ldots,h_n{=}E_n)$ | $M[E_{index}{<}E_{size}]$
expression	E ::= c | $E.c$ | $\mathrm{f}(E_1,\ldots,E_n)$ | $\mathrm{D}(E_1,\ldots,E_n)$| …

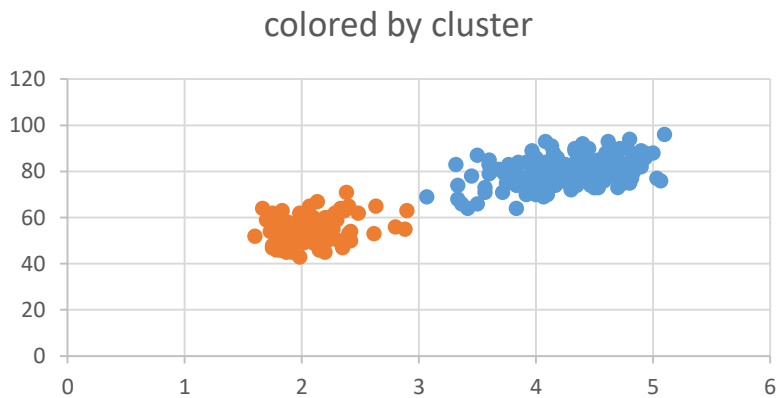
- An extended syntax (that compiles to the core):
  - Functions defined by table models T
  - Application $t(h_1{=}E_1,\ldots,h_n{=}E_n)$ defined by substitution semantics
  - Indexed models $M[E_{index}{<}E_{size}]$ capture structure of many models

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}.

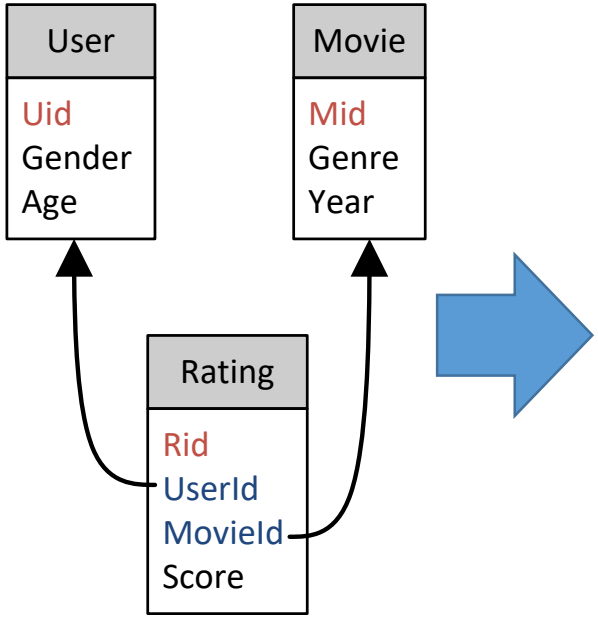
faithful

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

faithful

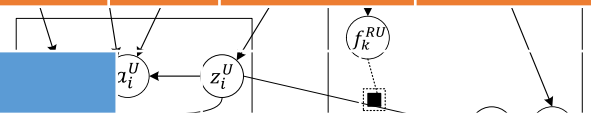
cluster	mod(2)	latent	Discrete(2,[0.5, 0.5])
Mean	real[2]	param	[for i < 2 -> Gaussian(0.0,1.0)]
Prec	real[2]	param	[for i < 2 -> Gamma(1.0,1.0)]
eruption_duration	real	output	Gaussian(Mean.[cluster],1/Prec.[cluster])

Automatic Model Suggestion



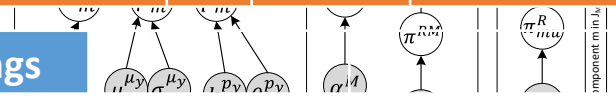
Users

UserCluster	Int	Latent	CDiscrete(N=4)
Gender	Bool	Output	CBernoulli()[UserCluster]
Age	Int	Output	CDiscrete(N=4)[UserCluster]



Movies

MovieCluster	Int	Latent	CDiscrete(N=4)
Genre	Int	Output	CDiscrete(N=20)[MovieCluster]
Year	Int	Output	CDiscrete(N=100)[MovieCluster]



Ratings

UserID	link(Users)	Input	
MovieID	link(Movies)	Input	
Score	int	Output	CDiscrete(N=5) [UserID.UserCluster] [MovieID.MovieCluster]

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

Language-Based Statistical Thinking

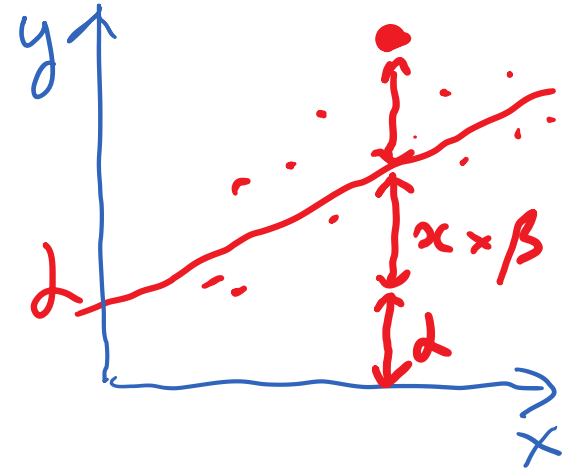
- "Statistical thinking will one day be as necessary for efficient citizenship as the ability to read and write." H.G. Wells
- Tabular seeks to empower today's spreadsheet users to be efficient citizens, by casting statistical thinking as a PL question
- Probabilistic programming as a field empowers 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://github.com/TabularLang/CoreTabular

Fabular: Regression Formulas as Probabilistic Programming

Johannes Borgström, Andrew D. Gordon, Long Ouyang, Claudio Russo, Adam Ścibior, Marcin Szymczak

Uppsala, MSR, Edinburgh, Stanford, Cambridge,



Probabilistic programming:

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

coefficients

predicted values of the y variable

In another galaxy, R's lm package:

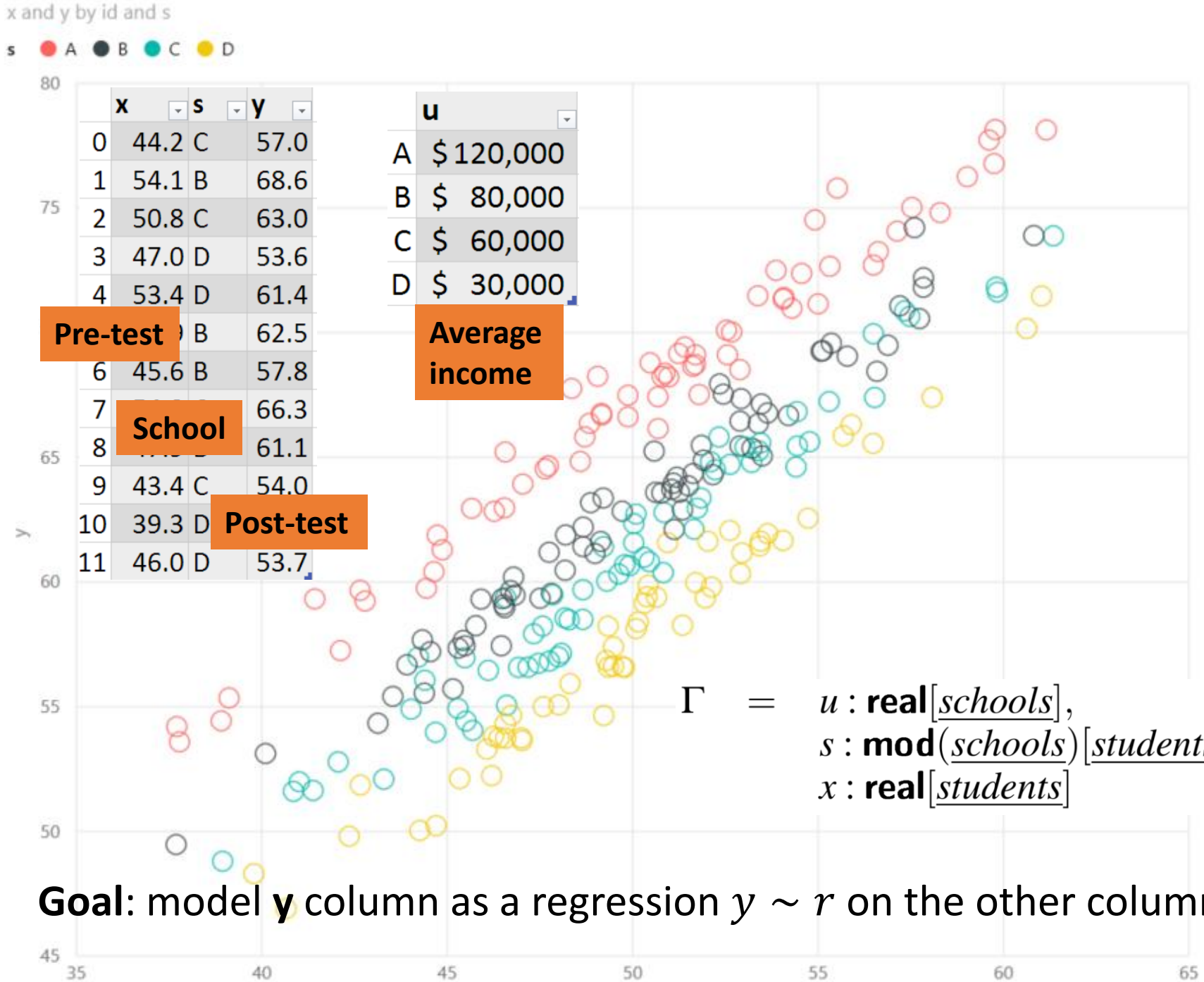
$$\cancel{y \sim 1 + x} \qquad y \sim x$$

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

A regression calculus $y \sim r$

$r ::=$	regression
$D(v_1, \ldots, v_n)$	noise with distribution D
$v\{\alpha \sim r\}$	predictor with coefficient **(new wrt R)**
$r + r'$	sum
$r \mid v$	grouping
$(\nu\alpha)r$	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	s	y
0	44.2	C	57.0
1	54.1	B	68.6
2	50.8	C	63.0
3	47.0	D	53.6
4	53.4	D	61.4
5	[illegible]	B	62.5
6	45.6	B	57.8
7	[illegible]	[illegible]	66.3
8	[illegible]	[illegible]	61.1
9	43.4	C	54.0
10	39.3	D	[illegible]
11	46.0	D	53.7

Pre-test

School

Post-test

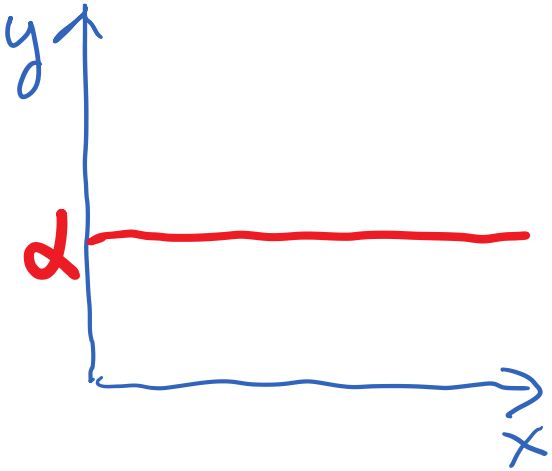
	u
A	$ 120,000
B	$ 80,000
C	$ 60,000
D	$ 30,000

Average income

$$\Gamma \quad = \quad \begin{array}{l} u : \textbf{real}[\underline{schools}], \\ s : \textbf{mod}(\underline{schools})[\underline{students}], \\ x : \textbf{real}[\underline{students}] \end{array}$$

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

(1) Pure Intercept

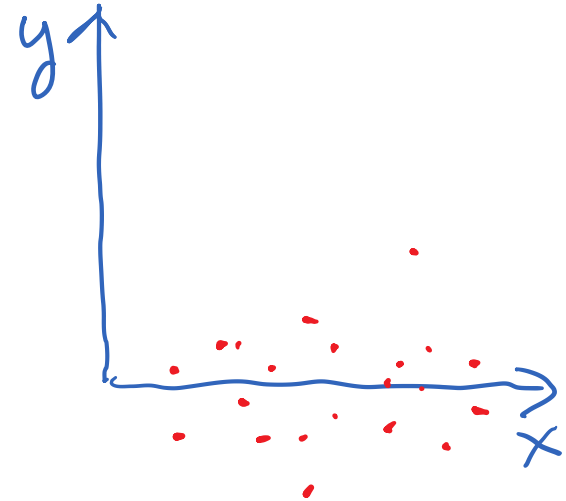


$$y \sim 1\{\alpha\} \sim \text{Gaussian}(0, s_{large}{}^2)\}$$

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

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

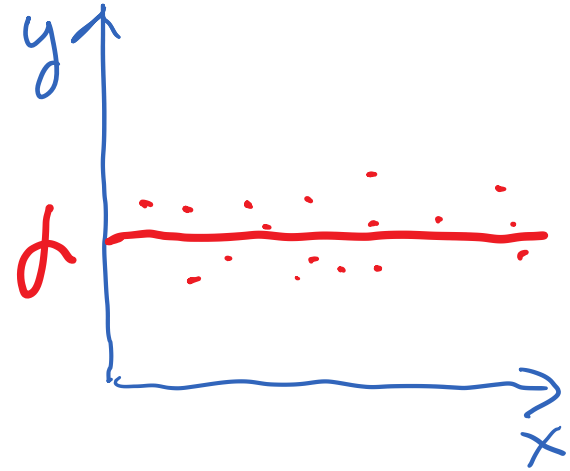
(2) Pure Noise



$$y \sim ?$$

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

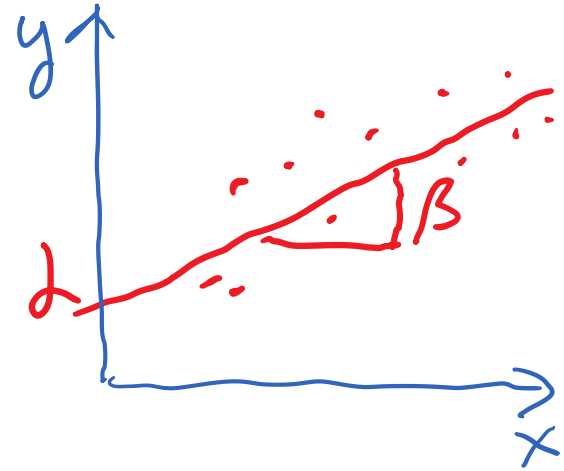
(3) Intercept with Noise



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

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

(4) Slope and Intercept

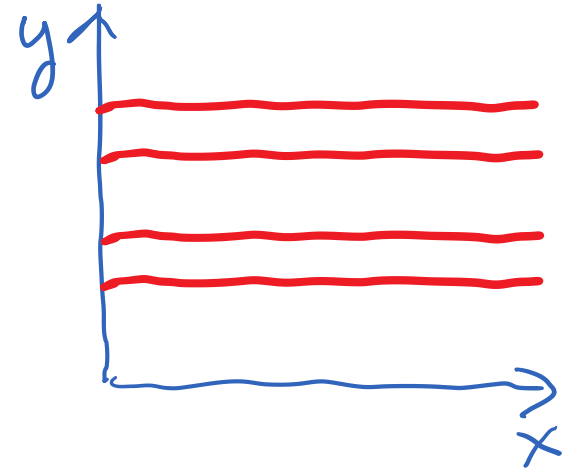


$$y \sim 1\{\alpha\} + x\{\beta\} + ?$$

$$\begin{aligned}
&\textbf{let } \alpha = \text{Gaussian}(0, {s_{large}}^2) \textbf{ in} \\
&\textbf{let } \beta = \text{Gaussian}(0, {s_{large}}^2) \textbf{ in} \\
&\textbf{let } \pi = \text{Gamma}(1, \lambda_{small}) \textbf{ in} \\
&((\alpha, \beta), [\textbf{for } z < \underline{students} \rightarrow \alpha + x[z] \times \beta + \text{Gaussian}(0, 1/\pi)])
\end{aligned}$$

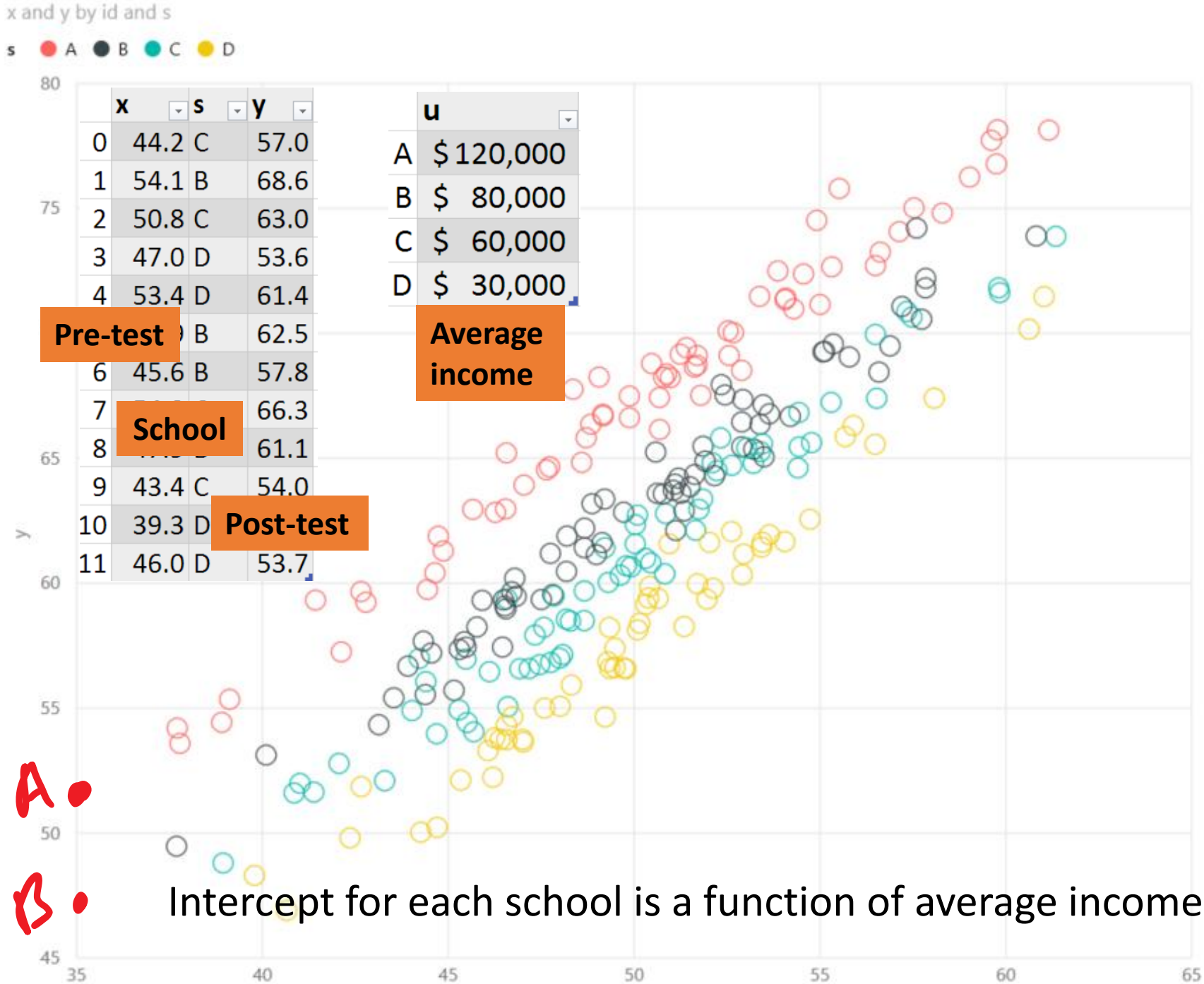
$$y \sim 1 + x$$

(5) Intercept per School



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

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



	x	s	y
0	44.2	C	57.0
1	54.1	B	68.6
2	50.8	C	63.0
3	47.0	D	53.6
4	53.4	D	61.4
5	[illegible]	B	62.5
6	45.6	B	57.8
7	[illegible]	[illegible]	66.3
8	[illegible]	[illegible]	61.1
9	43.4	C	54.0
10	39.3	D	[illegible]
11	46.0	D	53.7

Pre-test

School

Post-test

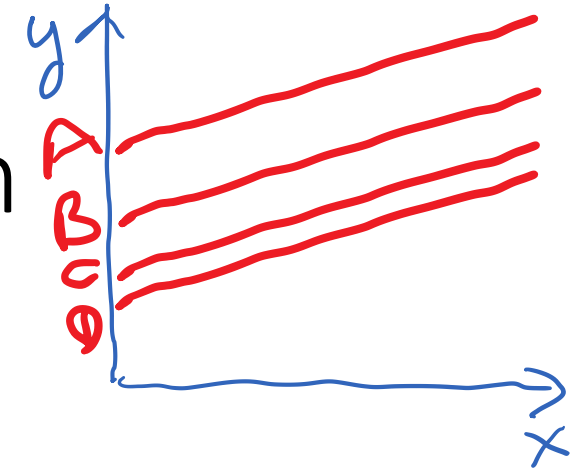
	u
A	$ 120,000
B	$ 80,000
C	$ 60,000
D	$ 30,000

Average income

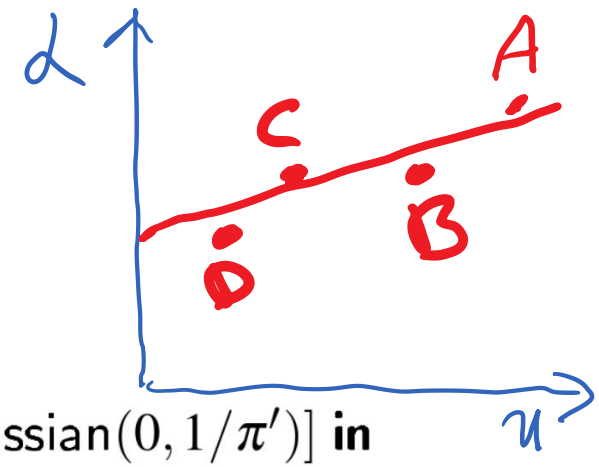
A.

B. Intercept for each school is a function of average income?

(6) Hierarchical Regression



$$y \sim (1\{\alpha \sim r_\alpha\} \mid s) + x\{\beta\}$$
$$\text{where } r_\alpha = 1\{a\} + u\{b\}$$



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

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

$$
\begin{aligned}
r ::= \quad & D(v_1, \ldots, v_n) \\
& v\{\alpha \sim r\} \\
& r + r' \\
& r \mid v \\
& (\nu\alpha) r
\end{aligned}
\qquad
\begin{aligned}
u, v ::= \quad & s \\
& x \\
& u : v
\end{aligned}
$$

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}|s)	-3174
Hierarchical	(1{alpha ~ 1 + u} |s) + x	-1871

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

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* $\vec{e} = [e_1; \ldots; e_n]$ be a phrase that denotes a multi-dimensional array of some type $T[e_n] \ldots [e_1]$.
An *index* for $\vec{e}$ is a list $[i_1; \ldots; i_n]$ with $0 \leq i_j < e_j$ for each j.

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

Typing Rules for Predictors:

(SCALAR)
$$\frac{\Gamma \vdash \diamond \quad s \in \mathbb{R}}{\Gamma; \vec{e} \vdash s : \textbf{real}}$$

(VAR)
$$\frac{\Gamma \vdash x : T[\vec{e}]}{\Gamma; \vec{e} \vdash x : T}$$

(INTERACT)
$$\frac{\Gamma; \vec{e} \vdash u : \textbf{real} \quad \Gamma; \vec{e} \vdash v : \textbf{real}}{\Gamma; \vec{e} \vdash (u : v) : \textbf{real}}$$

Translation of Predictors to Fun: $[\![v]\!]\ \vec{E} = E$

$$[\![s]\!]\ \vec{E} \triangleq s$$
$$[\![x]\!]\ \vec{E} \triangleq x[\vec{E}]$$
$$[\![u : v]\!]\ \vec{E} \triangleq [\![u]\!]\ \vec{E} \times [\![v]\!]\ \vec{E}$$

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

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

Typing Rules for Regressions:

(NOISE)

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

(SUM)

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

(GROUP)

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

(COEFF)

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

Translation of Regressions to Fun: $[\![r]\!]\ \vec{e}\ \vec{f}\ \vec{F} = E$

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

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

Fabular = Tabular + Formulas

Points			
X	real	input	
Y	real	output	~ 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	real	input	
X6	real	input	
Z	real	latent	~ X1 +X2 + X3 + X4 + X5 + X6
Y	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(b_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?

- Nested models may be eliminated – every regression normalizes to the form: $(\vec{v}\alpha)(P + N)$

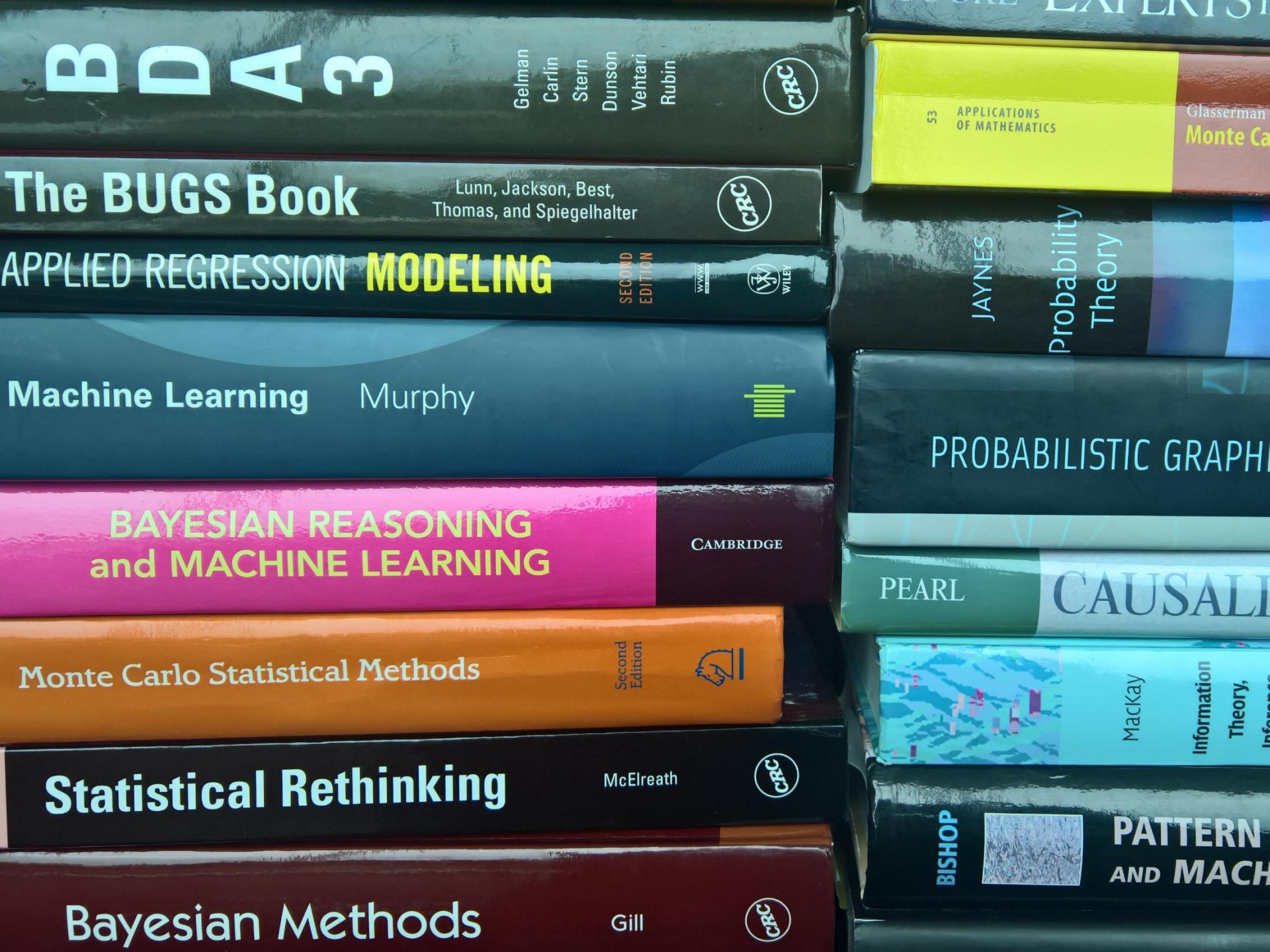
Classes of Terms: N, P

$N ::= \Sigma_{i=1}^{n} D_i(u_{i1}, \ldots, u_{i\|D_i\|})$	noise
$P ::= \Sigma_{i=1}^{n} v_i\{\alpha_i \sim N_i\} \mid \vec{w}_i$	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.

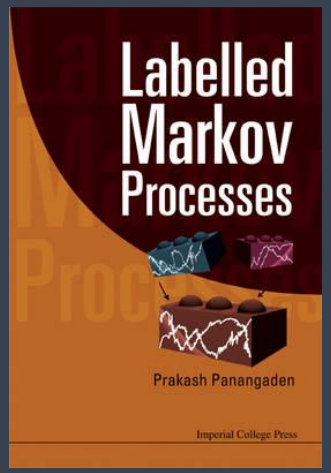
Summary of Fabular

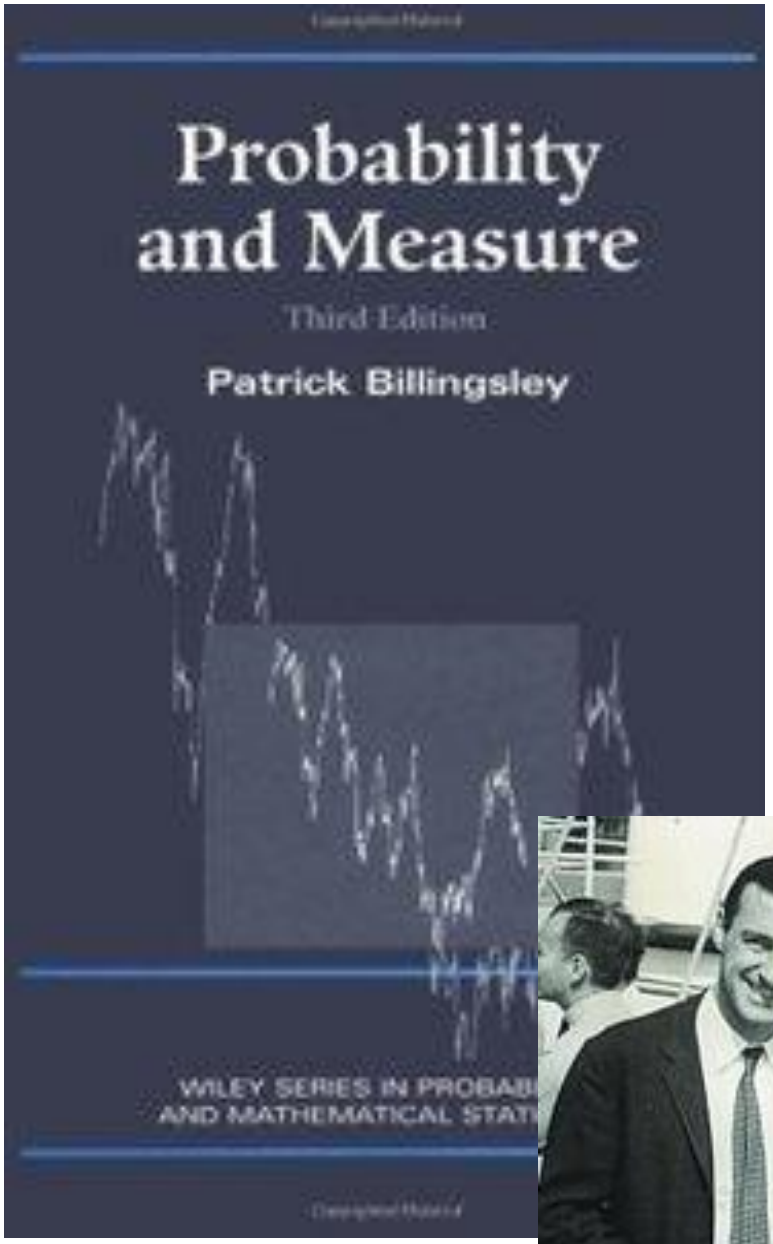
- The **regression calculus** is the first formal calculus of regressions, with a unique recursive syntax for hierarchical models (written $v\{\alpha \sim r\}$), going beyond the formulas of R.
- Regressions are boilerplate components of many models.
- **Fabular** mixes potentially vectorised regressions 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?

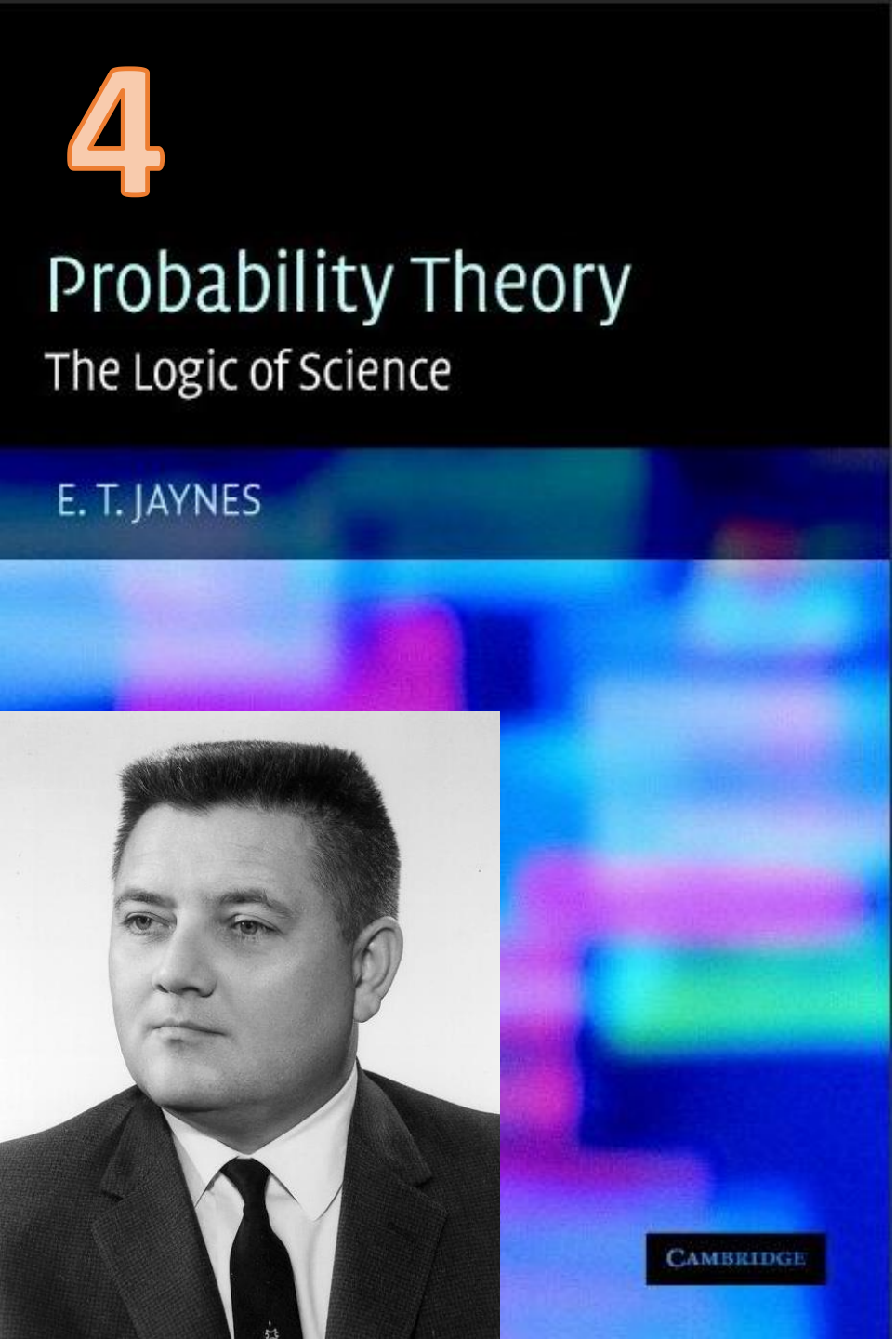

BDA3
Gelman
Carlin
Stern
Dunson
Vehtari
Rubin
CRC
The BUGS Book
Lunn, Jackson, Best,
Thomas, and Spiegelhalter
CRC
APPLIED REGRESSION MODELING
SECOND EDITION
WILEY
Machine Learning
Murphy
BAYESIAN REASONING
and MACHINE LEARNING
CAMBRIDGE
Monte Carlo Statistical Methods
Second Edition
Statistical Rethinking
McElreath
CRC
Bayesian Methods
Gill
CRC
53
APPLICATIONS
OF MATHEMATICS
Glasserman
JAYNES
Probability
Theory
PROBABILISTIC GRAPH
PEARL
CAUSALI
MacKay
Information
Theory,
BISHOP
PATTERN
AND MACH

5

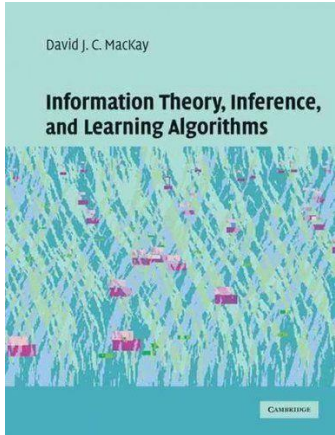
Mathematical Statistics



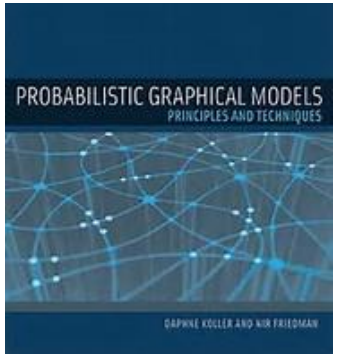


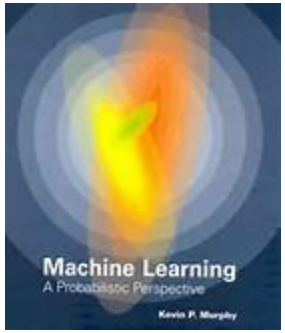


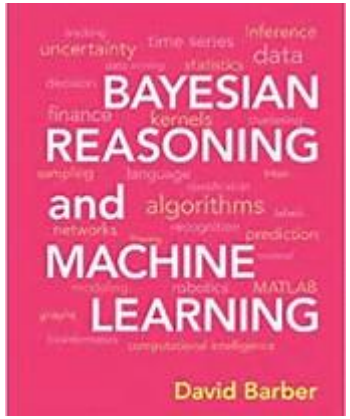
Statistical Physics

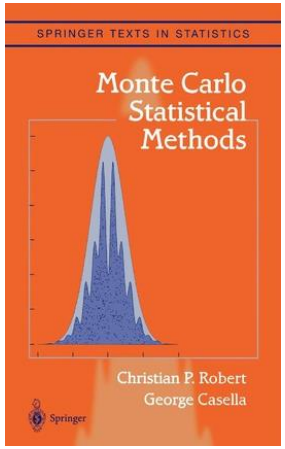


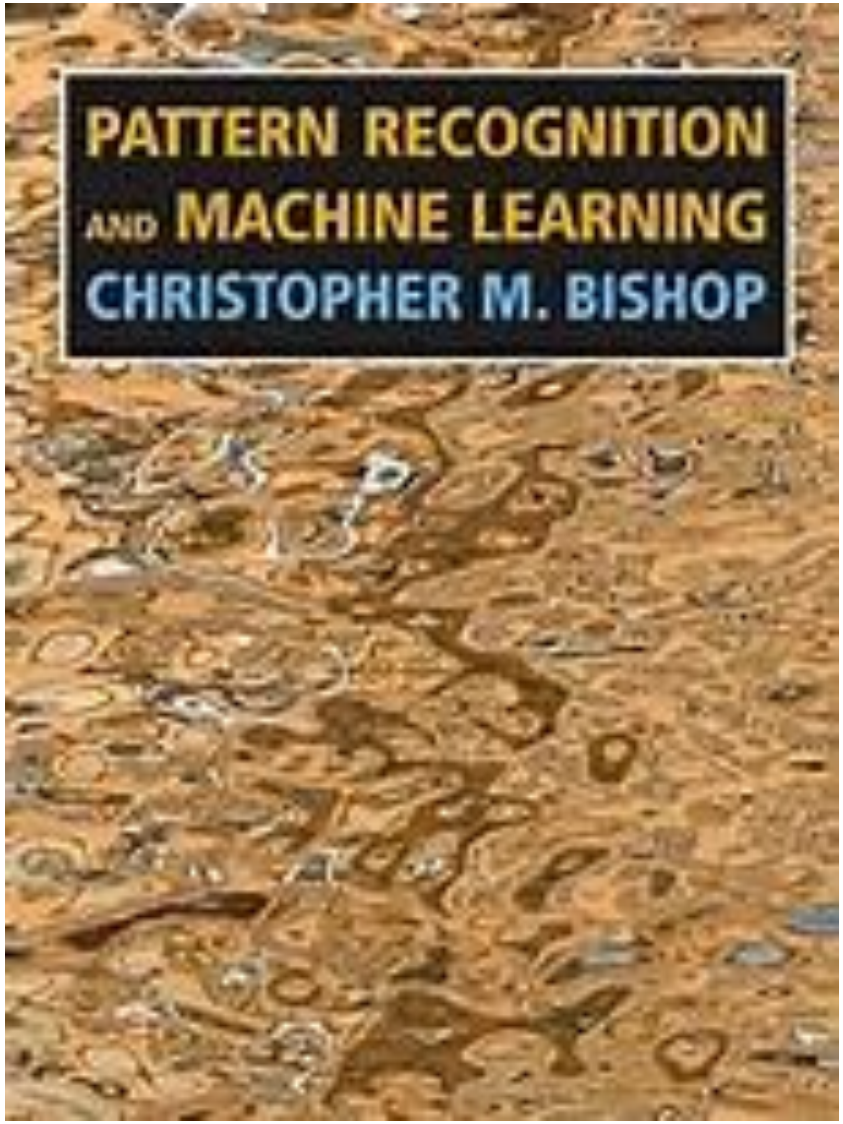
3 Machine Learning





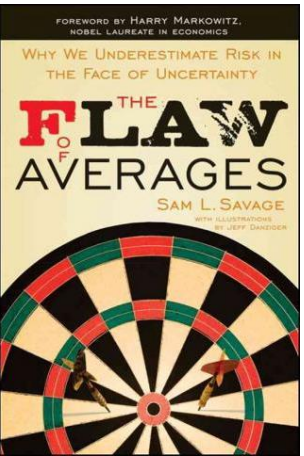


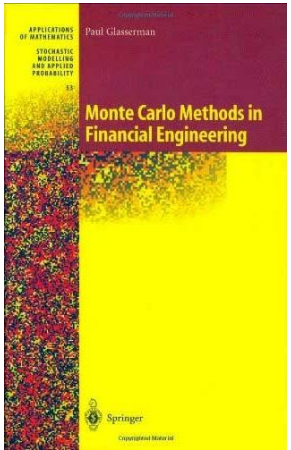


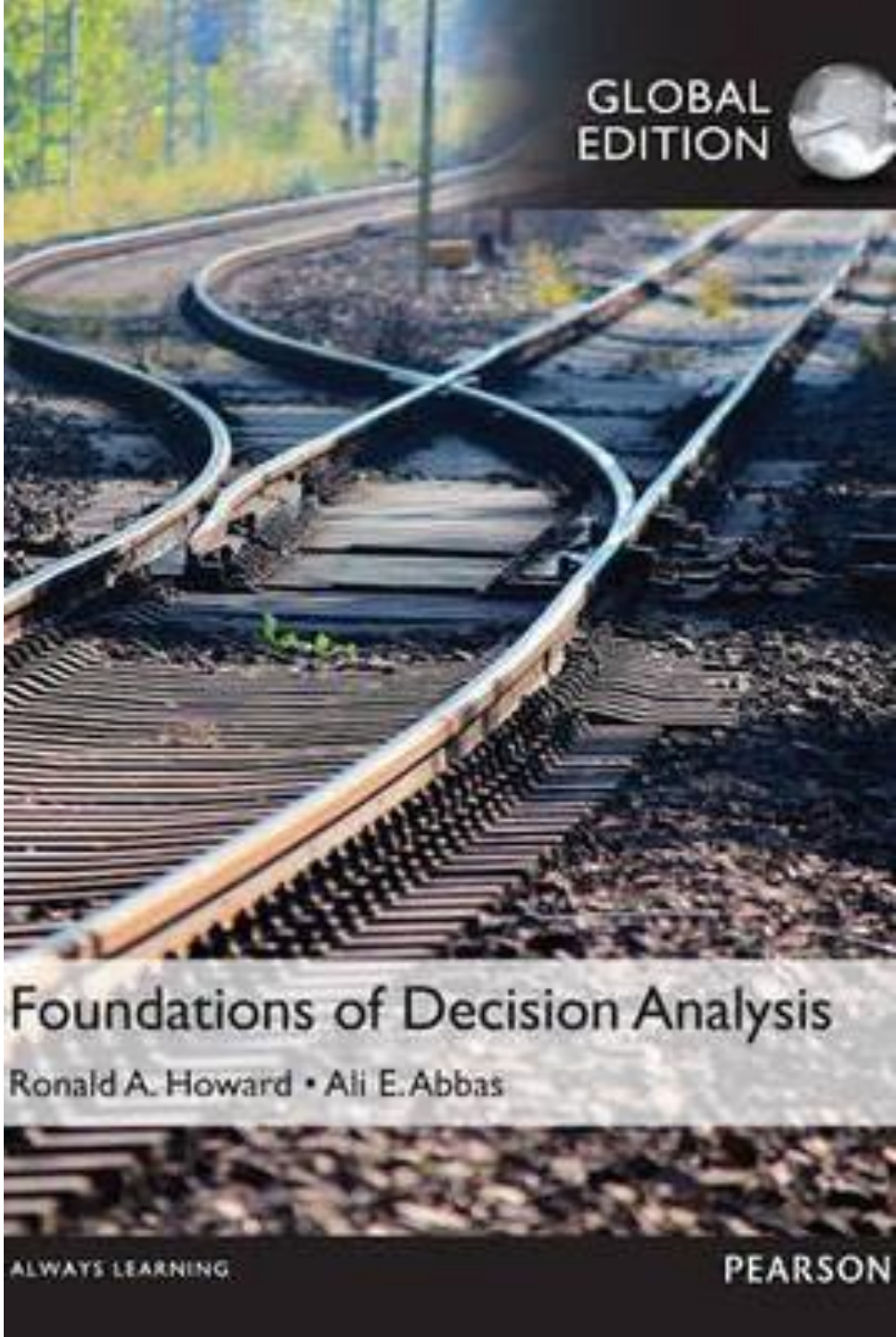


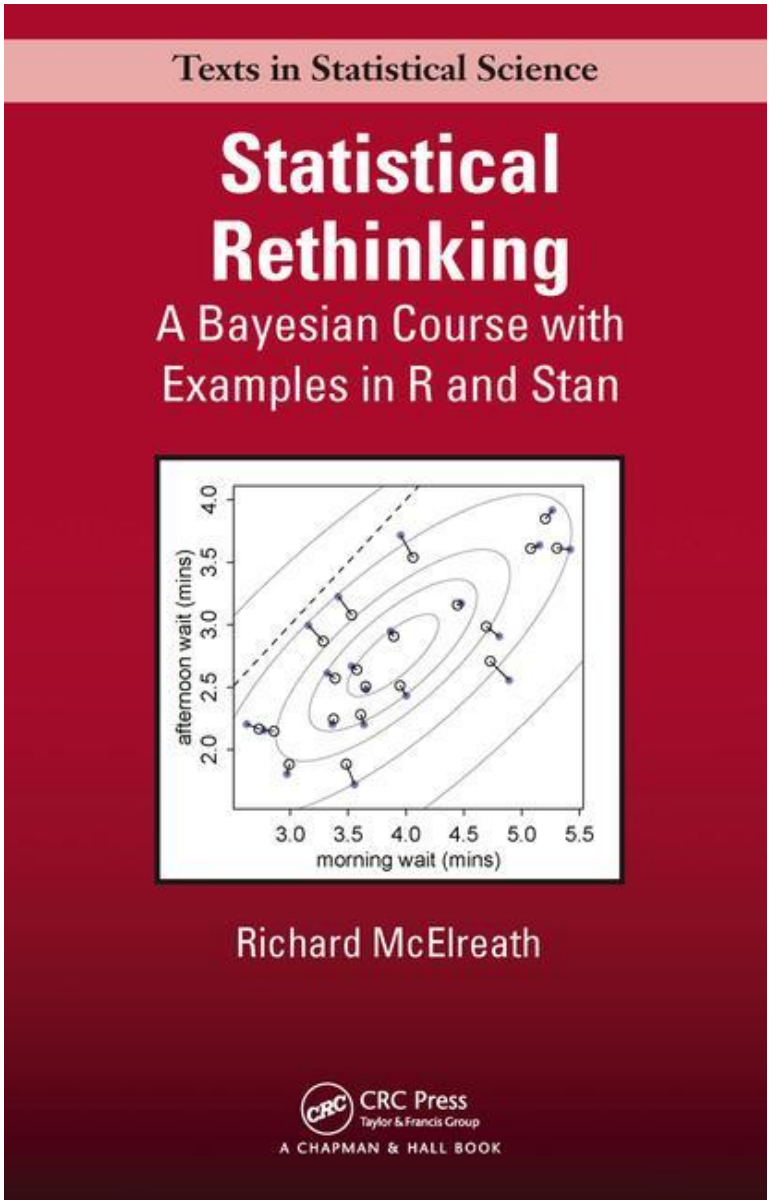
2

Business and Finance









Bayesian Statistics

1