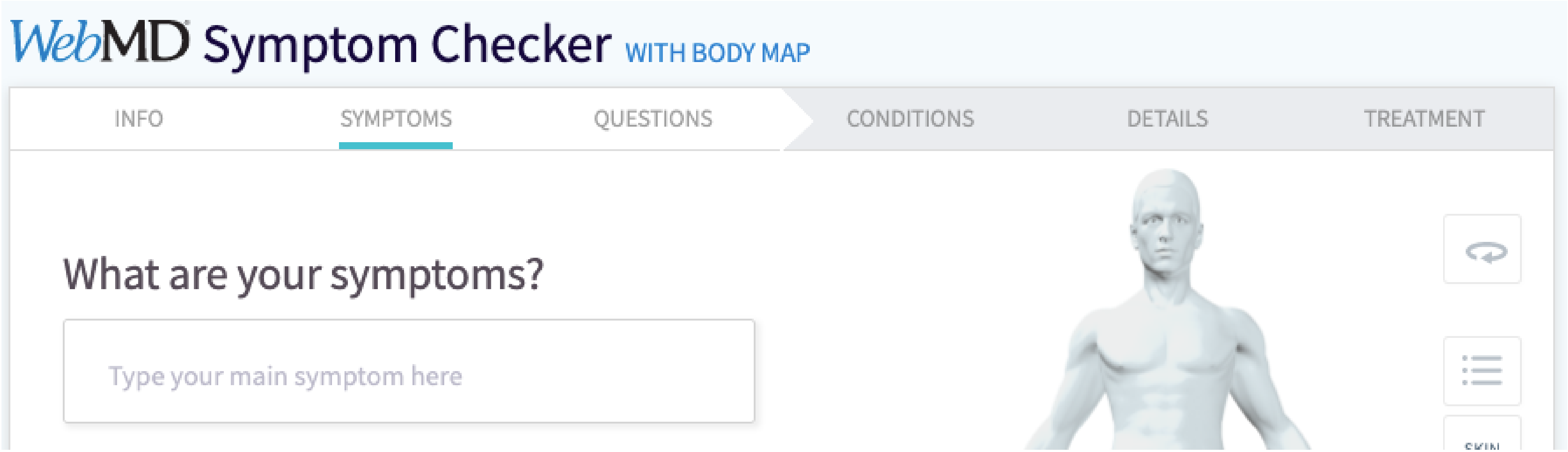
General Inference

Chris Piech
CS109, Stanford University

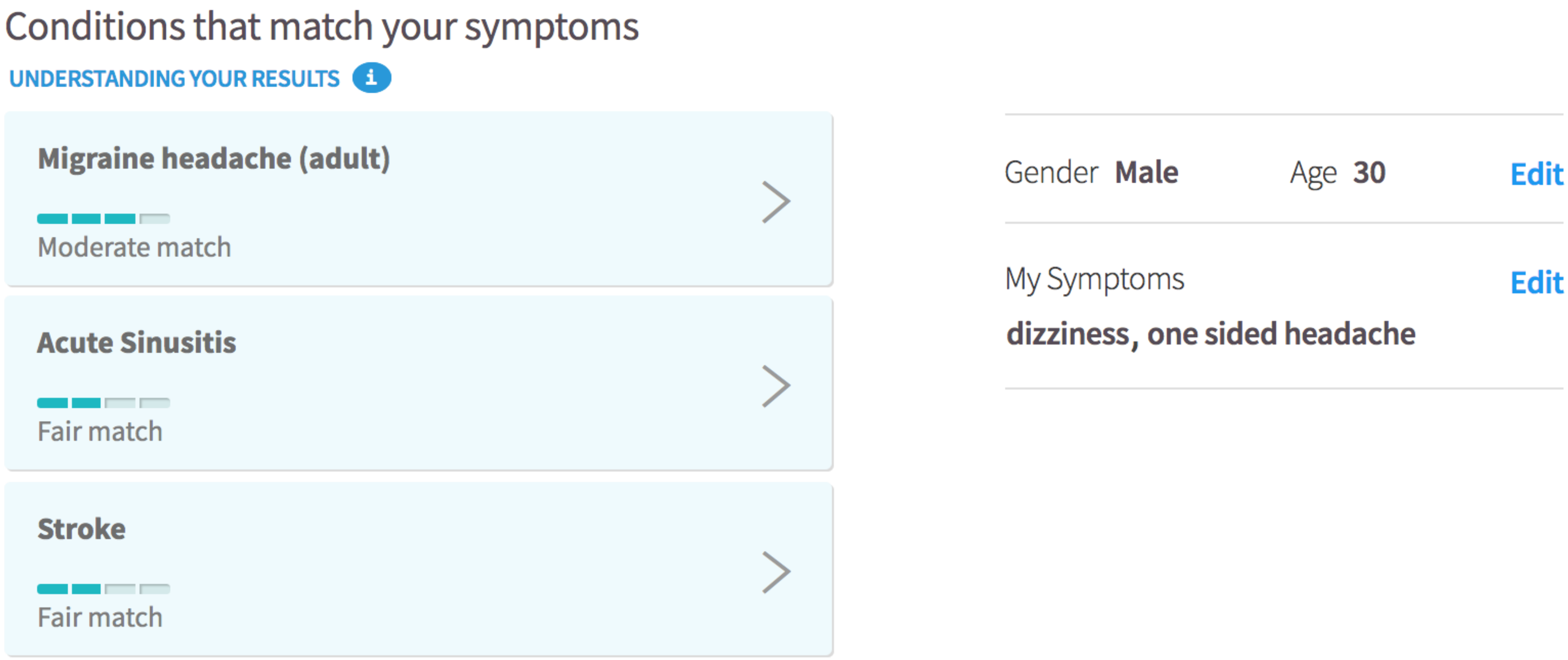
Why You Need a Model

WebMD®

Why You Need a Model



Multiple Random Variables. Start of Digital Revolution



Surprisingly Simple (if you can code)

Code

Probability

Three Guiding Questions

1. How do people actually define large models?
2. How can we do inference in large models?
3. What data can inform the design process?

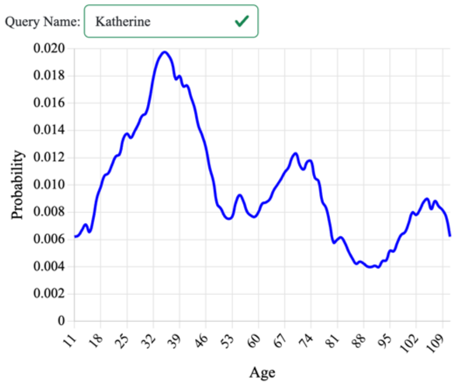
At this point you know inference with two random variables

Today: Five New Real + Exciting Problems

Age from C14

Updated Delivery Prob

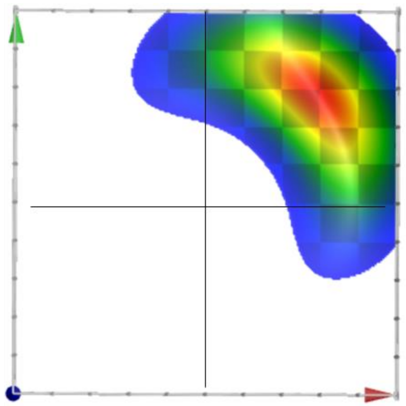
Age from Name



Hidden Chambers

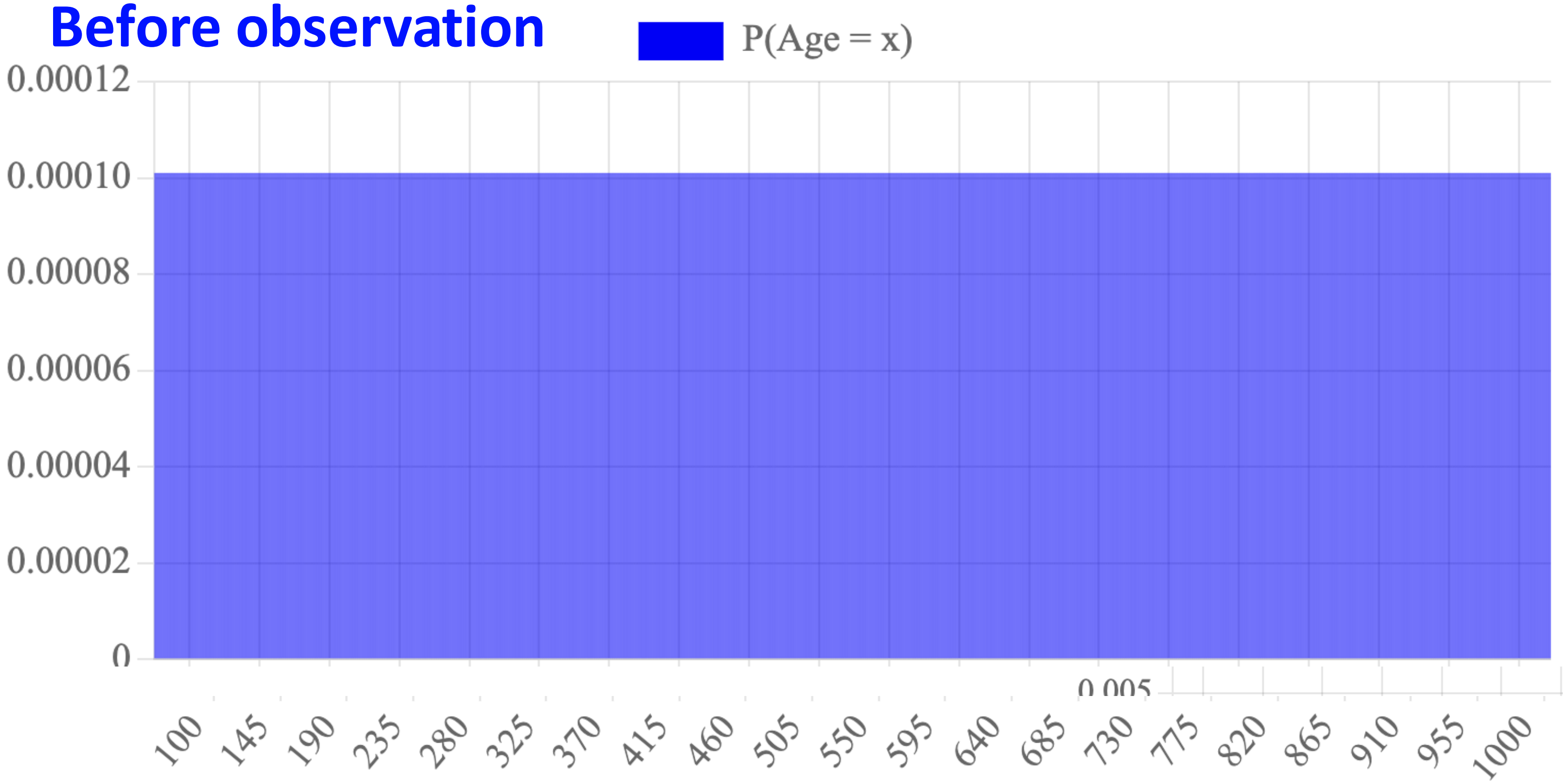
Stanford Eye Test

Cellphone Tracking Cont.



Update Belief PMF

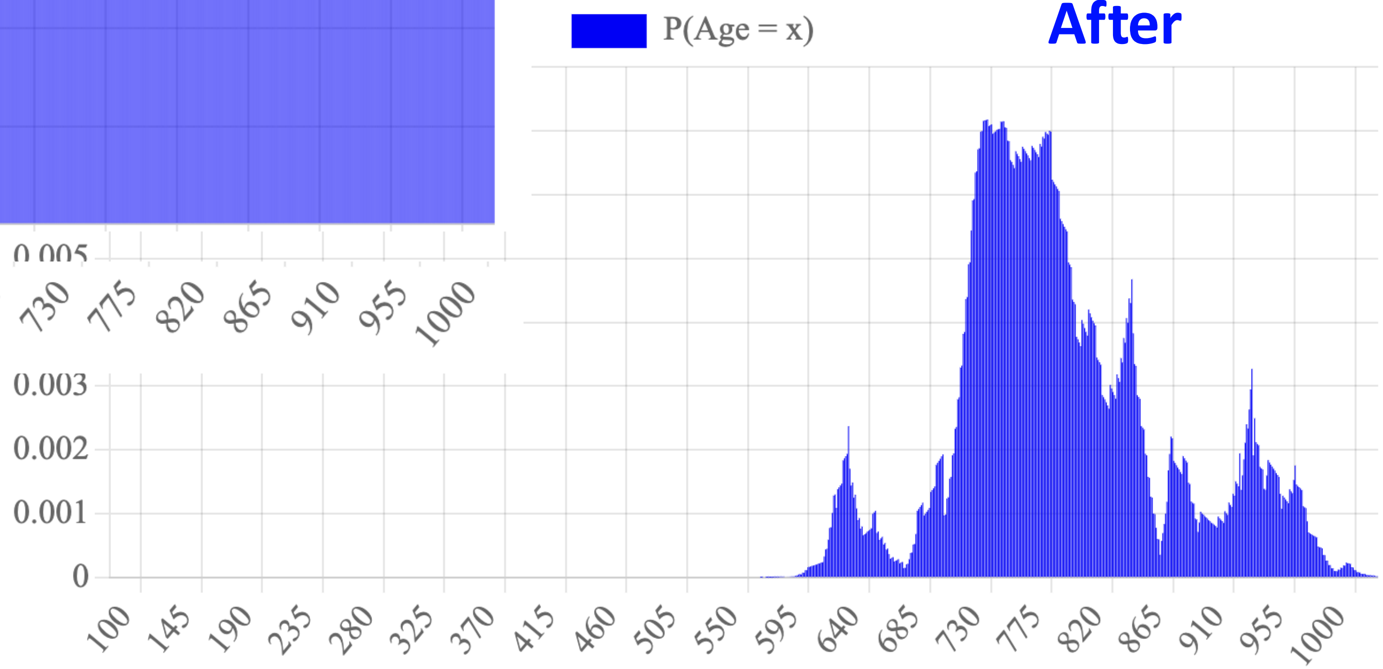
Before observation



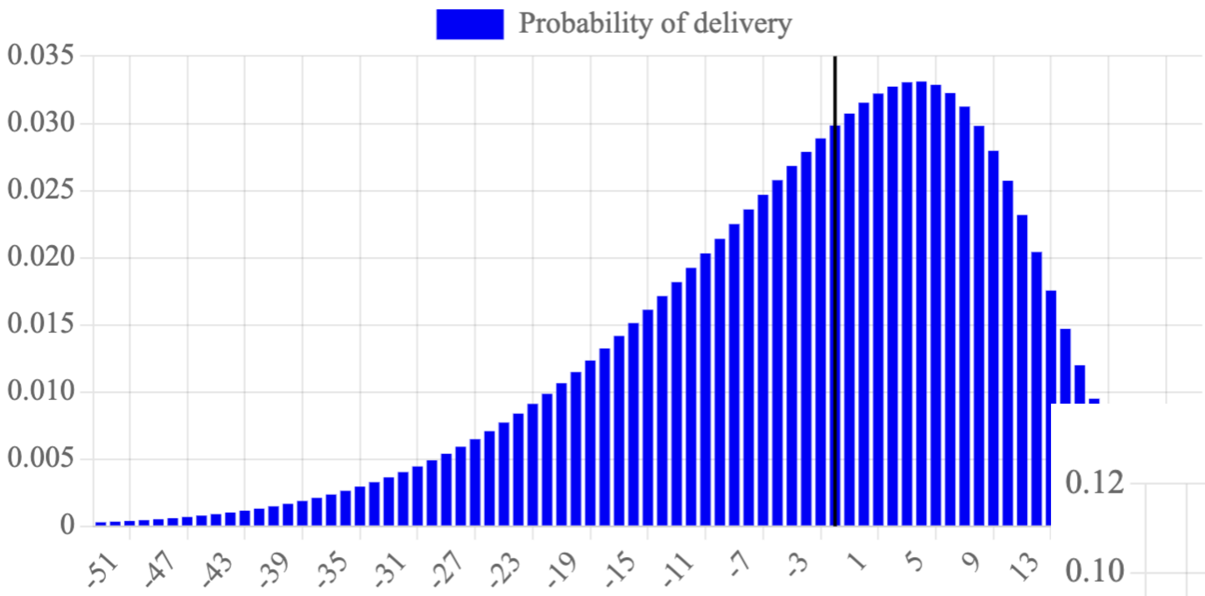
Observation

Remaining C14: 900

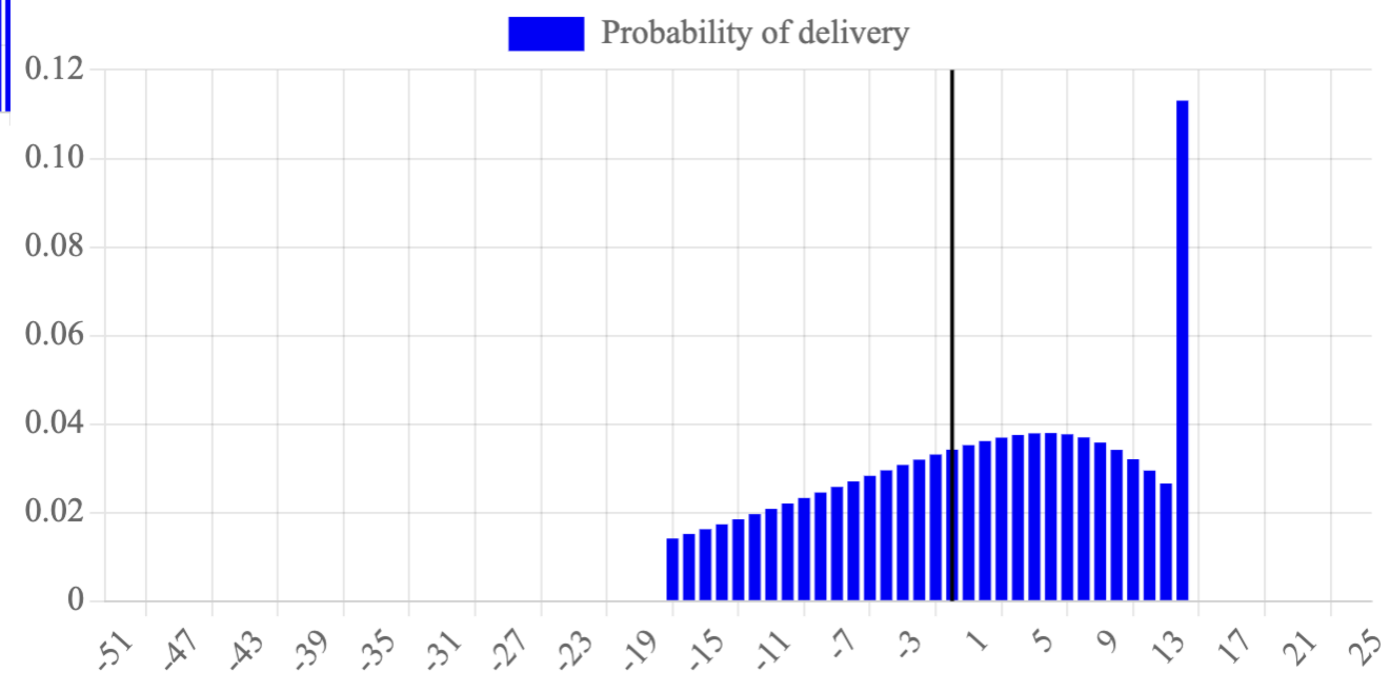
After



Baby delivery



Its 19 days until the due date and no baby



For each value *d:*

$$P(D = d | \text{no child so far})$$

$$= \frac{P(\text{no child so far} | D = d) P(D = d)}{P(\text{no child so far})}$$

```
def update_belief_carbon_dating(m = 900):
  # pr_A[i] is P(Age = i| m = 900).
  pr_A = {}
  for i in range(100,10000+1):
    prior = 1 / n_years # P(A = i)
    likelihood = calc_likelihood(m, i) #P(M=m|A=i)
    pr_A[i] = likelihood * prior
  # implicitly computes the normalization constant
  normalize(pr_A)
  return pr_A
```

```
def update_belief_baby(prior, today = 10):
  # pr_D[i] is P(D = i| No Baby Yet).
  pr_D = {}
  for i in range(-50,25):
    # P(NoBaby | D = i)
    likelihood = 0 if i < today else 1
    pr_D[i] = likelihood * prior[i]
  # implicitly computes the LOTP
  normalize(pr_D)
  return pr_D
```

What do you notice is the same. What is different?

Normalize does so much for you!!!

Normalize: scale each value so that they would sum to 1

```
normalize({
  'cat'      : 5,
  'dog'      : 10,
  'axolotyl' : 5
})
```

```
{
  'cat'      : 0.2,
  'dog'      : 0.4,
  'axolotyl' : 0.2
}
```

Hidden Pyramid Chambers with Poisson + Bayes

Hidden Pyramid Chambers with Poisson + Bayes

Number of Muons

$$f(X = x | M = 12) = \frac{P(M = 12 | X = x) f(X = x)}{P(M = 12)}$$

Amount of limestone

Denominator option #1: Law of total probability

$$P(M = 12) = \int_0^{100} P(M = 12 | X = x) f(X = x) dx$$

Hidden Pyramid Chambers with Poisson + Bayes

Number of Muons

$$f(X = x|M = 12) = \frac{P(M = 12|X = x)f(X = x)}{P(M = 12)}$$

Amount of limestone

Denominator option #2: Solve for K

$$f(X = x|M = 12)$$
$$= \frac{P(M = 12|X = x)f(X = x)}{P(M = 12)}$$
$$= \frac{\frac{\left(100 \cdot e^{-x/40}\right)^{12} e^{-\left(100 \cdot e^{-x/40}\right)}}{12!} \cdot \frac{1}{100}}{P(M = 12)} \qquad \propto e^{-\frac{12x}{40} - 100 \cdot e^{-x/40}}$$

FunTip: Integration via Sampling

Teachers hate him!

See how you (approximate) the integral of any function with **one simple trick**.

ProTip: Integration via Sampling

6 Goodbye integral, my old friend [14 points]

Fall 2017

In this problem we are going to compute probabilities for a random variable X that can takes on values in the range $0 \leq x \leq 1$, and has probability density function:

$$f_X(x) = \frac{1}{K} \cdot g(x)$$

Where g(x) is some terribly nasty and non-integratable function. For your sanity I won't even write out g. It is that bad. In such a situation, we can turn to the power of computers to help us (you may assume that while g is impossible to integrate, you were able to code g as a function in a programming language). The key idea that we are going to use is called Monte Carlo Integration:

> Generate N values $(X_1, X_2, \ldots, X_N)$ uniformly sampled over a range (a, b). We can approximate the integral of a function h over (a, b) as:
>
> $$\int_a^b h(x)\, dx \approx \frac{(b-a)}{N} \sum_{i=1}^{N} h(X_i)$$

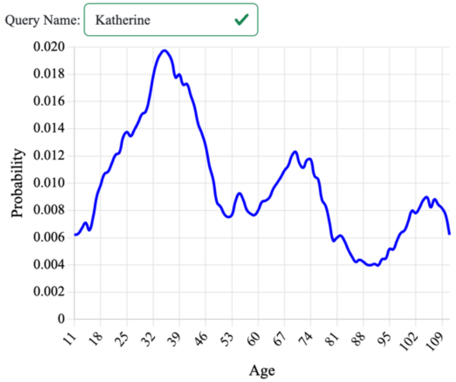
Pretty amazing! Why did we bother with integrals at all? This question requires you to write pseudo code. Such code does not have to compile, but it should be specific enough that a knowledgable programmer could implement what you have described. You may use a function `random(a, b)` which returns a sample from $X \sim \text{Uni}(a, b)$. You may also use a function `g(x)` which is the hard-to-integrate term in $f_X(x)$.

Today: Five New Real + Exciting Problems

Age from C14

Updated Delivery Prob

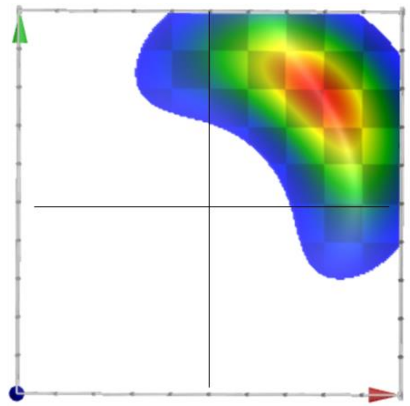
Age from Name



Hidden Chambers

Stanford Eye Test

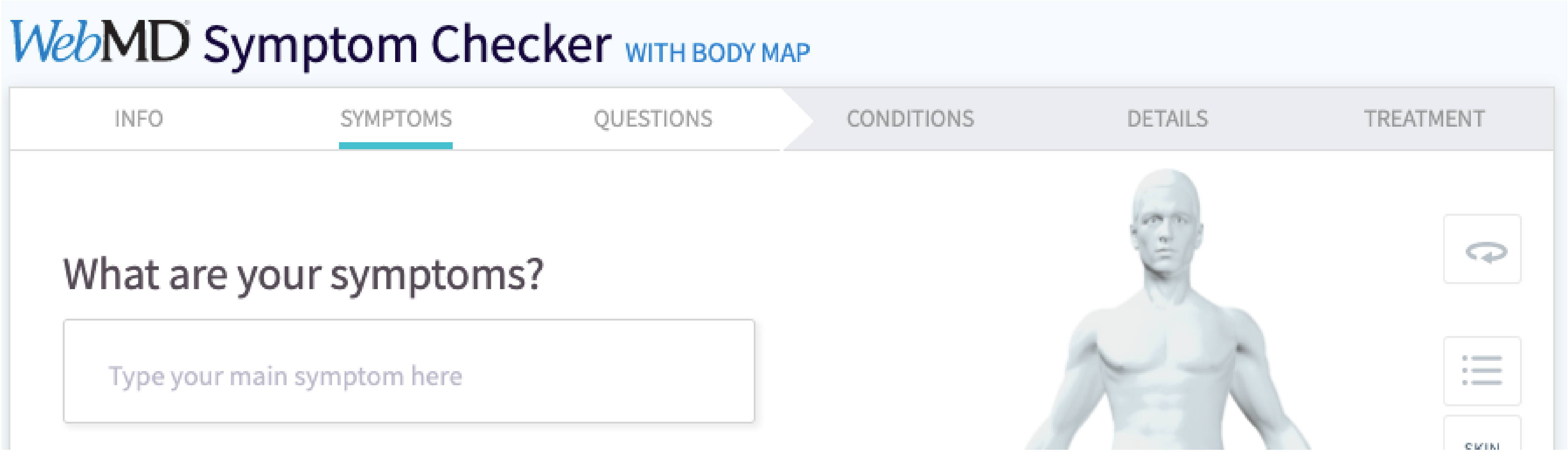
Cellphone Tracking Cont.



Many real world problems have way more than two random variables…

Why You Need a Model

Why You Need a Model



Multiple Random Variables. Start of Digital Revolution

Conditions that match your symptoms

UNDERSTANDING YOUR RESULTS

Migraine headache (adult)

Moderate match

Acute Sinusitis

Fair match

Stroke

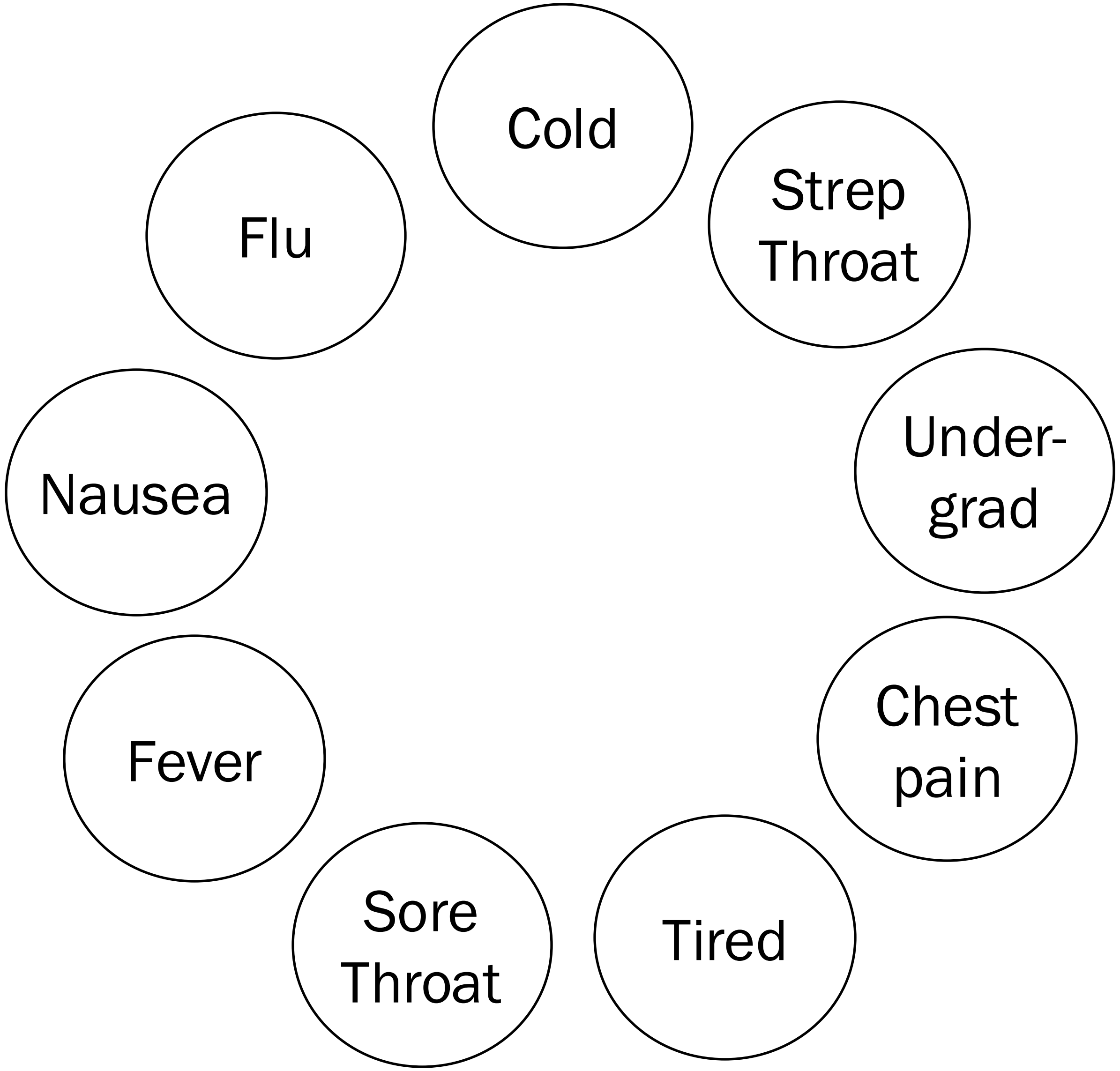
Fair match

Gender **Male** Age **30** Edit

My Symptoms Edit

dizziness, one sided headache

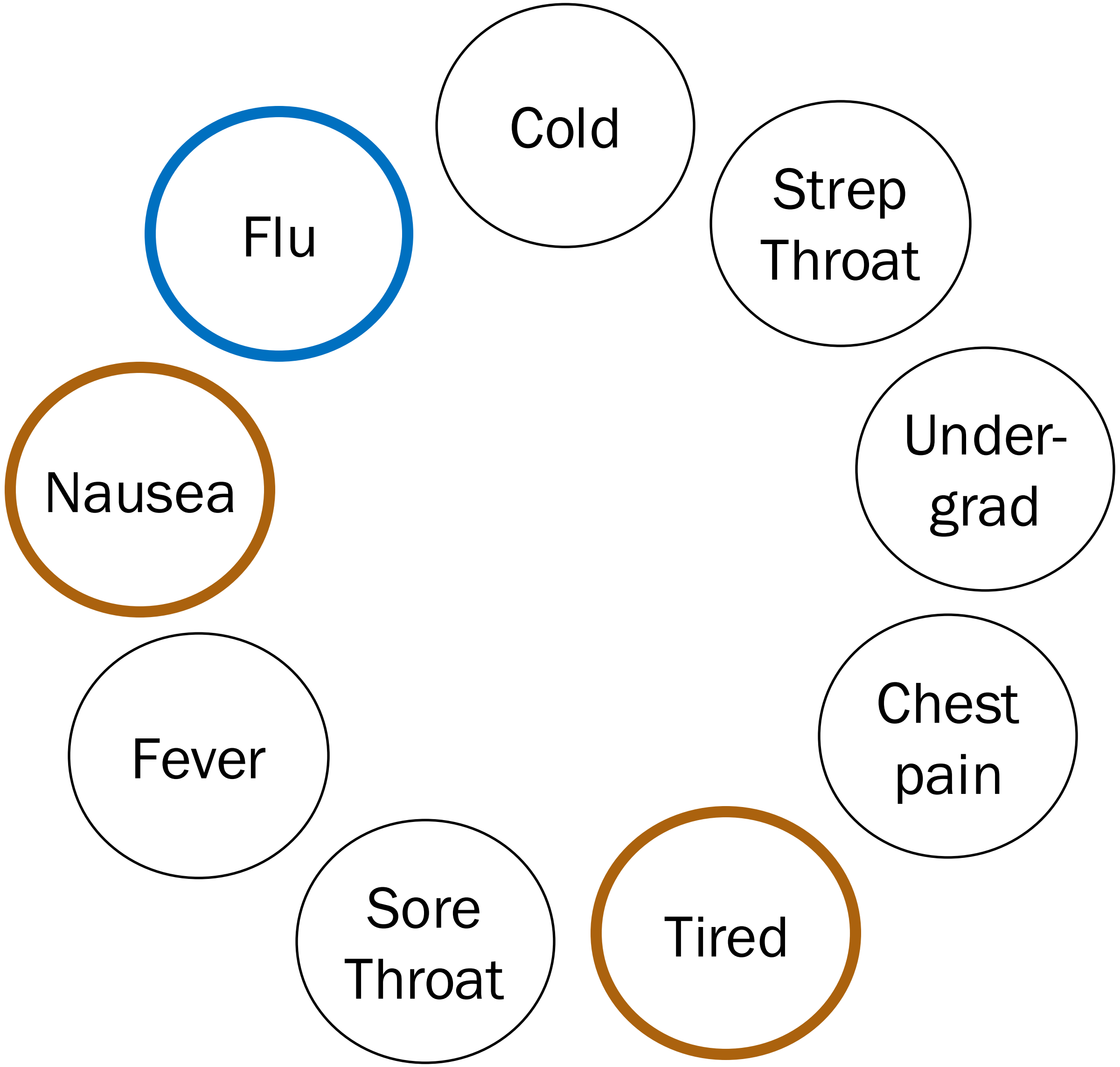
Challenge #1: Many Inference Questions



Inference question:

Given the values of some random variables, what are the conditional distributions of some other random variables?

Challenge #1: Many Inference Questions

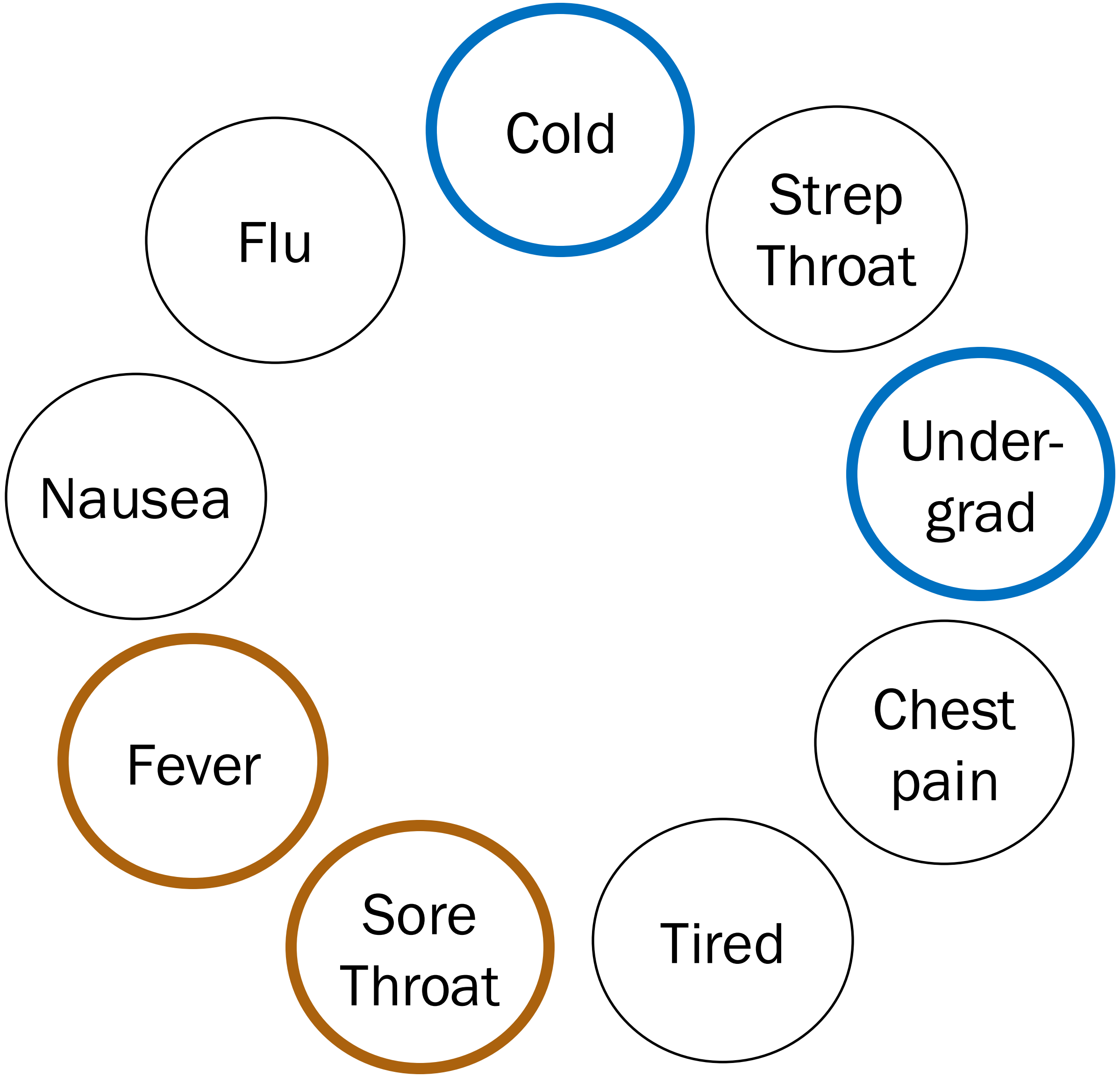


One inference question:

$$P(F = 1 | N = 1, T = 1)$$

$$= \frac{P(F = 1, N = 1, T = 1)}{P(N = 1, T = 1)}$$

Challenge #1: Many Inference Questions

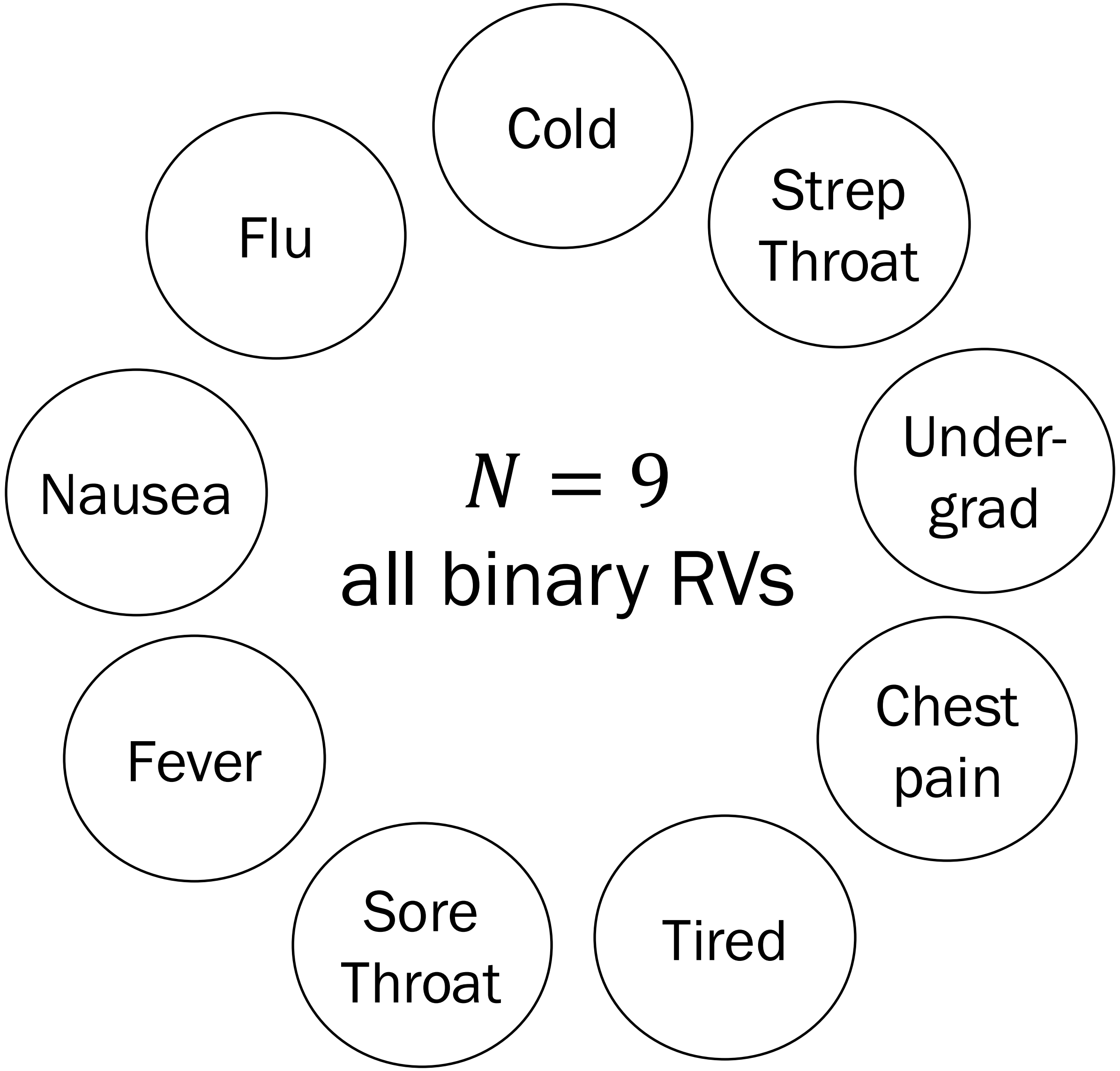


Another inference question:

$$P(C_o = 1, U = 1 | S = 0, F_e = 0)$$

$$= \frac{P(C_o = 1, U = 1, S = 0, F_e = 0)}{P(S = 0, F_e = 0)}$$

Challenge #2: Joint is Large



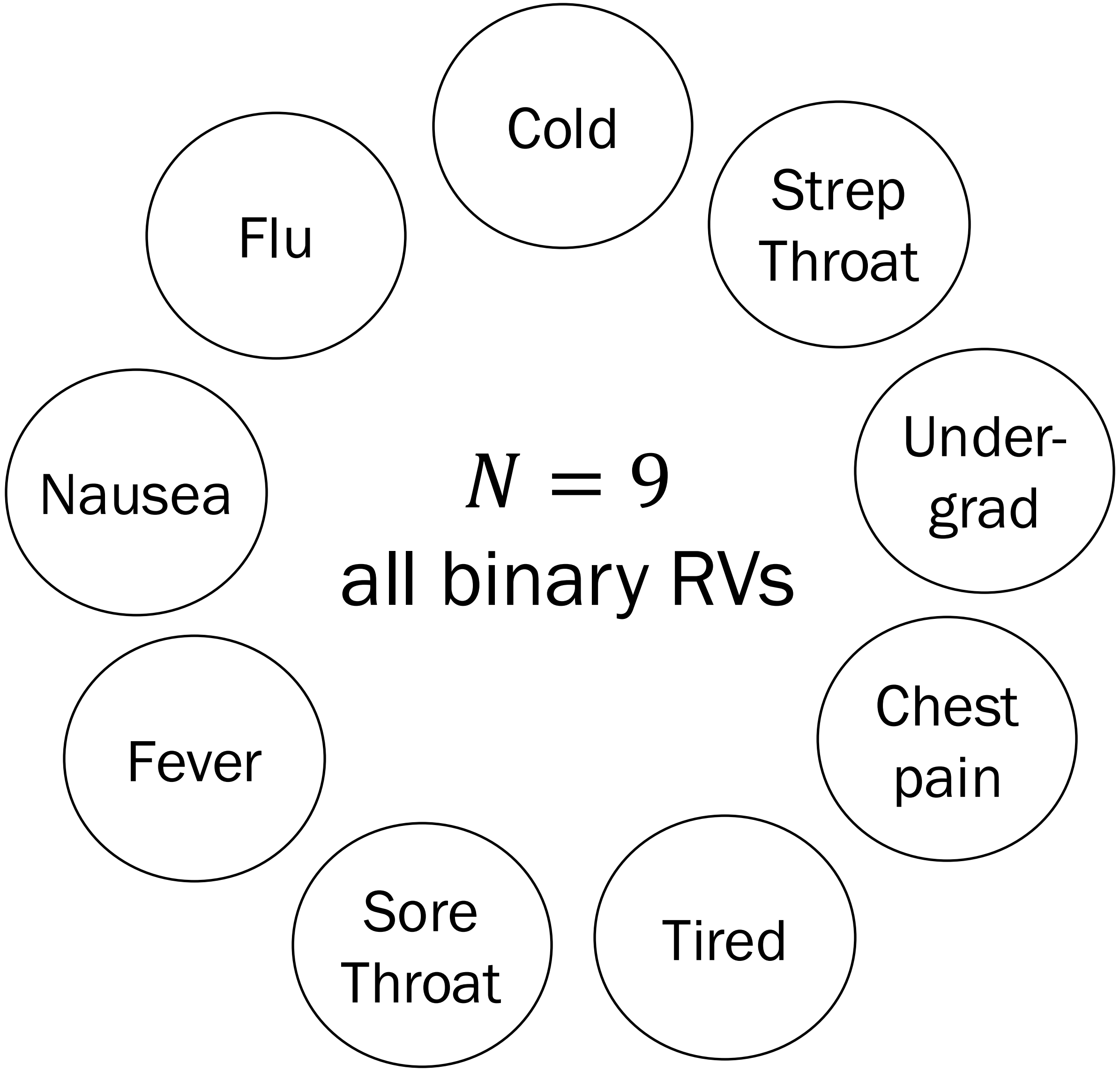
$N = 9$
all binary RVs

If we knew the **joint distribution**, we can answer all probabilistic inference questions.

What is the size of the joint probability table?

A. 2^{N-1} entries
B. N^2 entries
C. 2^N entries
D. None/other/don't know

Challenge #2: Joint is Large



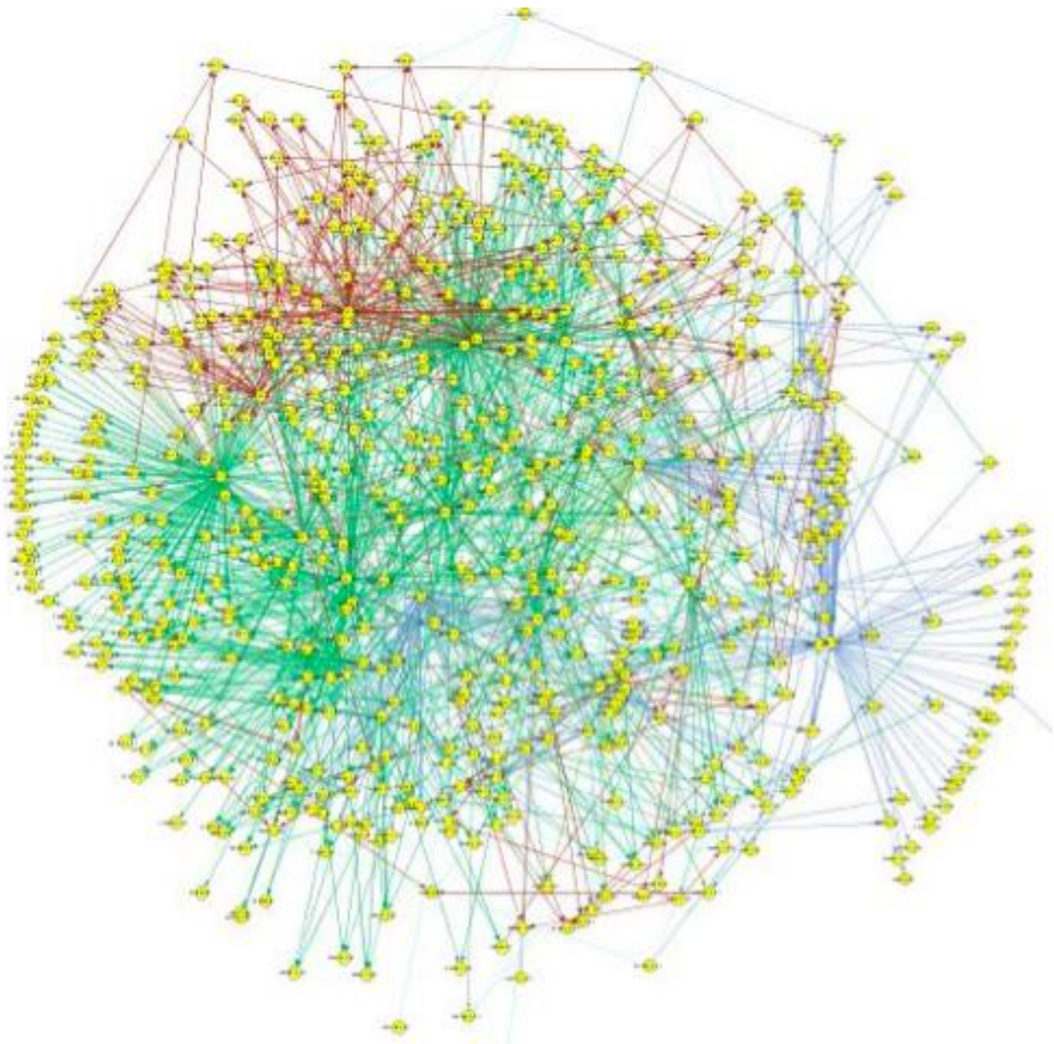
If we knew the **joint distribution**, we can answer all probabilistic inference questions.

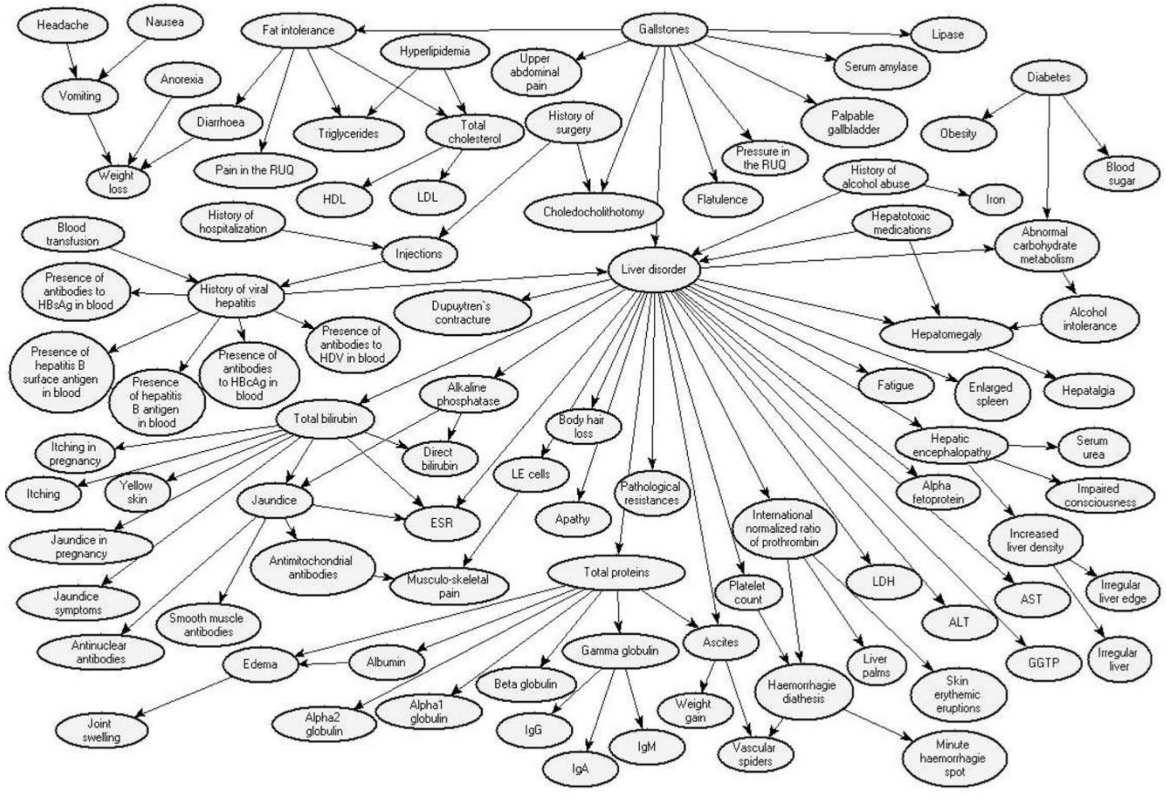
What is the size of the joint probability table?

A. 2^{N-1} entries
B. N^2 entries
C. 2^N entries
D. None/other/don't know

Naively specifying a joint distribution is, in general, intractable.

N can be large…





Three Guiding Questions

1. How do people actually define large models?

2. How can we do inference in large models?

3. What data can inform the design process?

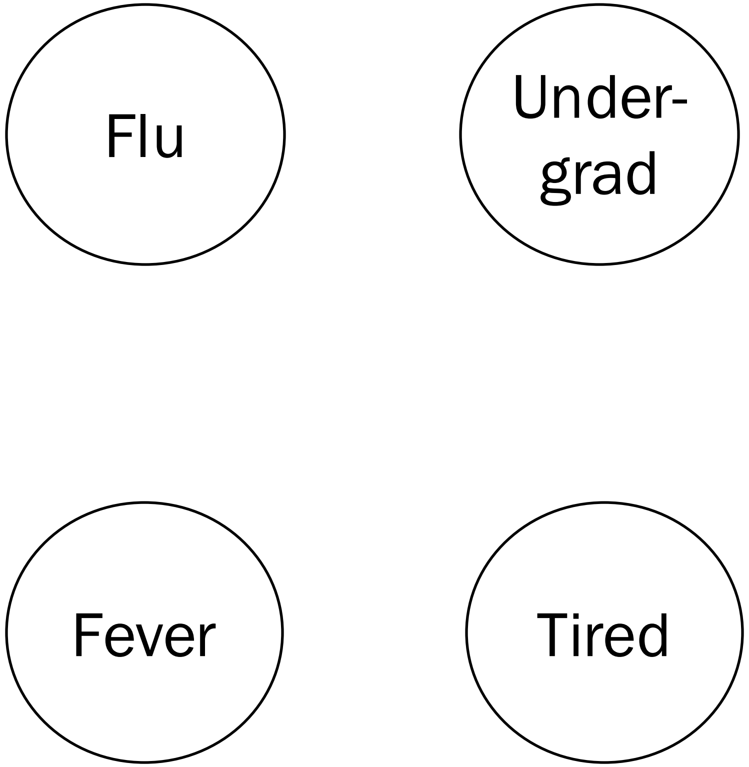
Three Guiding Questions

1. How do people actually define large models?
2. How can we do inference in large models?
3. What data can inform the design process?

Why You Need a Model

WebMD®

A simpler WebMD

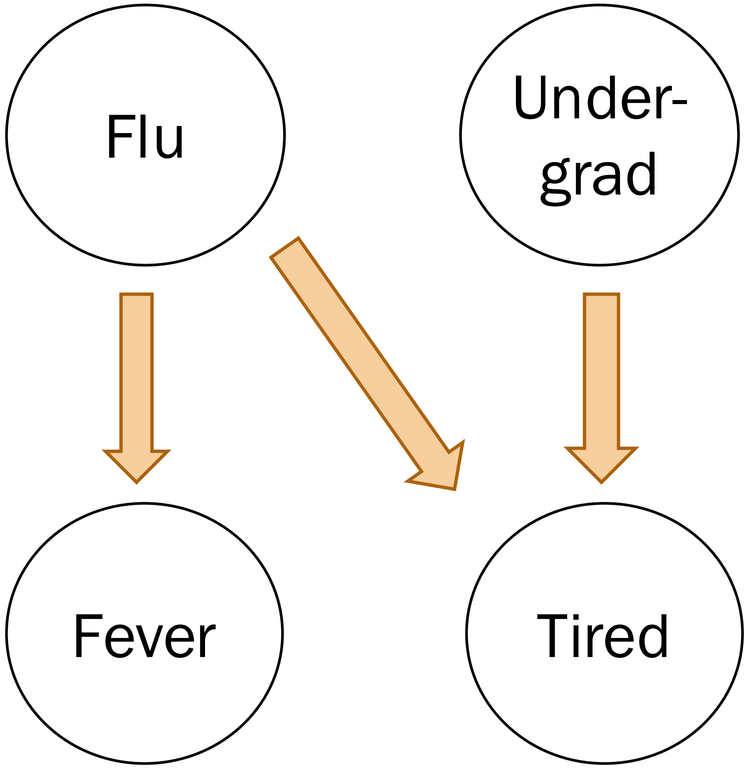


Great! Just specify $2^4 = 16$ joint probabilities…?

$$P(F_{lu} = a, F_{ev} = b, U = c, T = d)$$

We can compress the joint if we know the generative story…

Constructing a Bayesian Network



What would a Stanford flu expert do?

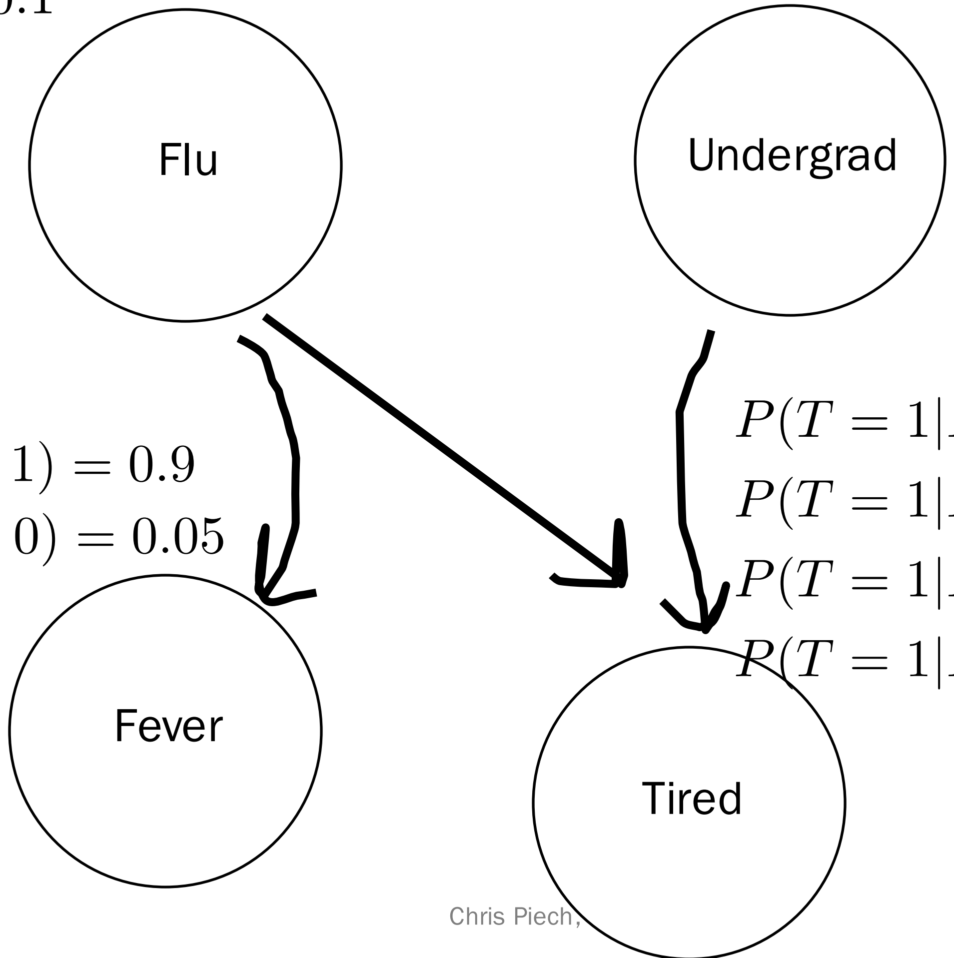
1. ✅ Describe the causality.
2. Provide $P(\text{values}|\text{causal parents})$ for each random variable
3. Implicitly assumes independences.

Recall: Probabilistic Model

✅ 2. Provide P(values|causal parents) for each random variable

$P(Fl = 1) = 0.1$

$P(U = 1) = 0.8$



$P(Fev = 1|Flu = 1) = 0.9$
$P(Fev = 1|Flu = 0) = 0.05$

$P(T = 1|Flu = 0, U = 0) = 0.1$
$P(T = 1|Flu = 0, U = 1) = 0.8$
$P(T = 1|Flu = 1, U = 0) = 0.9$
$P(T = 1|Flu = 1, U = 1) = 1.0$

Could we write a python program which makes a fake person from this joint?

To the Code

Midjourney 2023. Prompt: “a lot of excited (non human) pixar characters running off to computers”

```python
def make_sample():
  """
  Make Sample
  -------------------
  chose a single sample from the joint distribution
  """
  # prior on causal factors
  flu = bern(0.1)
  undergrad = bern(0.8)

  # choose fever based on flue
  if flu == 1: fever = bern(0.9)
  else:        fever = bern(0.05)

  # choose tired based on (undergrade and flu)
  if undergrad == 1 and flu == 1:   tired = bern(1.0)
  elif undergrad == 1 and flu == 0: tired = bern(0.8)
  elif undergrad == 0 and flu == 1: tired = bern(0.9)
  else:                             tired = bern(0.1)

  # a sample from the joint has an
  # assignment to *all* random variables
  return {
      'flu':flu,
      'undergrad':undergrad,
      'fever':fever,
      'tired':tired
    }
```

```python
def make_sample():
    """
    Make Sample
    -------------------
    chose a single sample from the joint distribution
    """
    # prior on causal factors
    flu = bern(0.1)
    undergrad = bern(0.8)

    # choose fever based on flue
    if flu == 1: fever = bern(0.9)
    else:        fever = bern(0.05)

    # choose tired based on (undergrade and flu)
    if undergrad == 1 and flu == 1:   tired = bern(1.0)
    elif undergrad == 1 and flu == 0: tired = bern(0.8)
    elif undergrad == 0 and flu == 1: tired = bern(0.9)
    else:                             tired = bern(0.1)

    # a sample from the joint has an
    # assignment to *all* random variables
    return {
          'flu':flu,
          'undergrad':undergrad,
          'fever':fever,
          'tired':tired
      }
```

```python
def make_sample():
    """
    Make Sample
    -------------------
    chose a single sample from the joint distribution
    """
    # prior on causal factors
    flu = bern(0.1)
    undergrad = bern(0.8)

    # choose fever based on flue
    if flu == 1: fever = bern(0.9)
    else:        fever = bern(0.05)

    # choose tired based on (undergrade and flu)
    if undergrad == 1 and flu == 1:   tired = bern(1.0)
    elif undergrad == 1 and flu == 0: tired = bern(0.8)
    elif undergrad == 0 and flu == 1: tired = bern(0.9)
    else:                             tired = bern(0.1)

    # a sample from the joint has an
    # assignment to *all* random variables
    return {
            'flu':flu,
            'undergrad':undergrad,
            'fever':fever,
            'tired':tired
        }
```

```python
def make_sample():
    """
    Make Sample
    -------------------
    chose a single sample from the joint distribution
    """
    # prior on causal factors
    flu = bern(0.1)
    undergrad = bern(0.8)

    # choose fever based on flue
    if flu == 1: fever = bern(0.9)
    else:        fever = bern(0.05)

    # choose tired based on (undergrade and flu)
    if undergrad == 1 and flu == 1:   tired = bern(1.0)
    elif undergrad == 1 and flu == 0: tired = bern(0.8)
    elif undergrad == 0 and flu == 1: tired = bern(0.9)
    else:                             tired = bern(0.1)

    # a sample from the joint has an
    # assignment to *all* random variables
    return {
        'flu':flu,
        'undergrad':undergrad,
        'fever':fever,
        'tired':tired
    }
```

```python
def make_sample():
    """
    Make Sample
    -------------------
    chose a single sample from the joint distribution
    """
    # prior on causal factors
    flu = bern(0.1)
    undergrad = bern(0.8)

    # choose fever based on flue
    if flu == 1: fever = bern(0.9)
    else:        fever = bern(0.05)

    # choose tired based on (undergrade and flu)
    if undergrad == 1 and flu == 1:   tired = bern(1.0)
    elif undergrad == 1 and flu == 0: tired = bern(0.8)
    elif undergrad == 0 and flu == 1: tired = bern(0.9)
    else:                             tired = bern(0.1)

    # a sample from the joint has an
    # assignment to *all* random variables
    return {
            'flu':flu,
            'undergrad':undergrad,
            'fever':fever,
            'tired':tired
        }
```

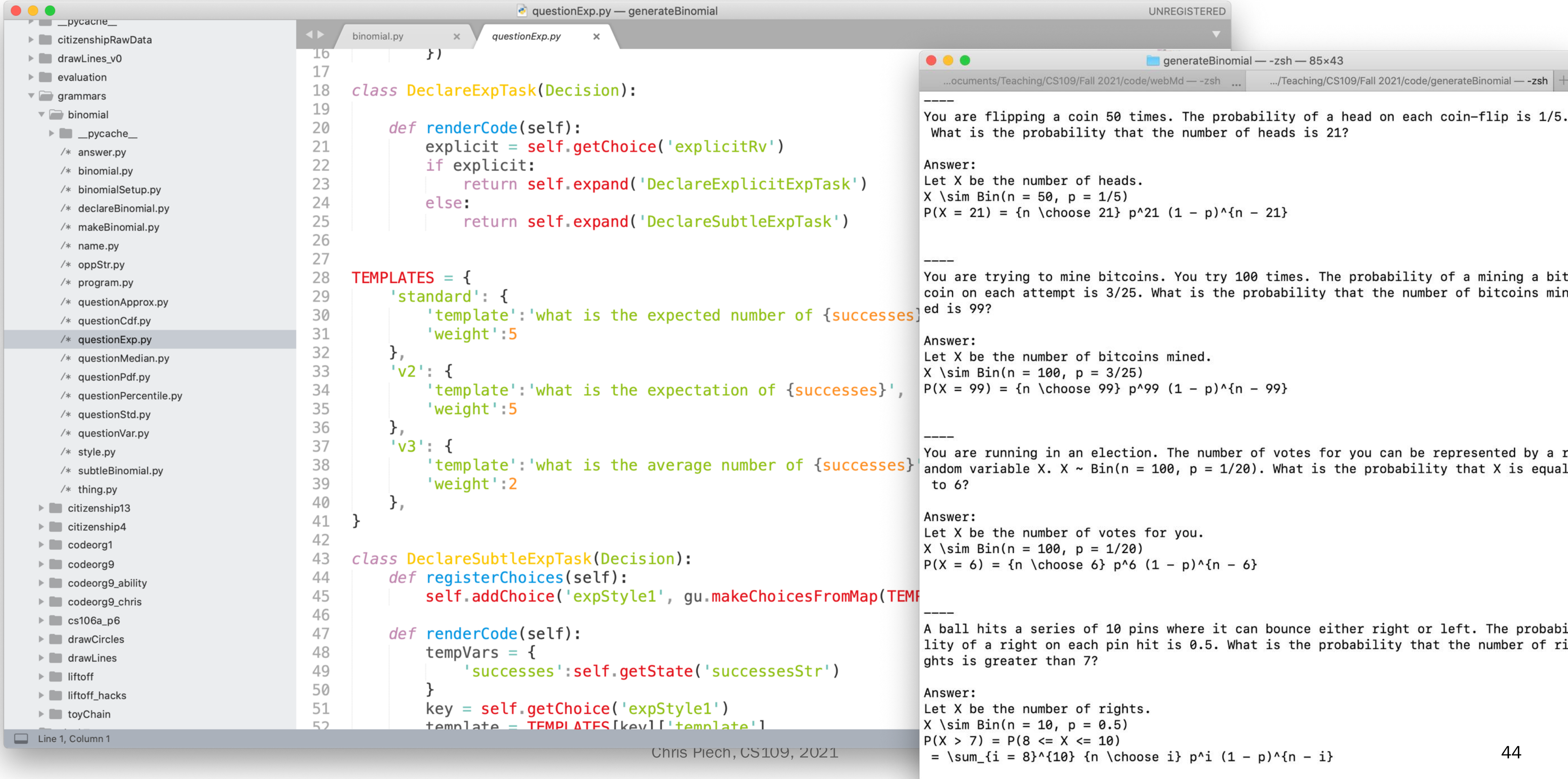
Can You Sample from the Joint?

Writing a python program that can **sample** from the joint, is the same as defining the joint.

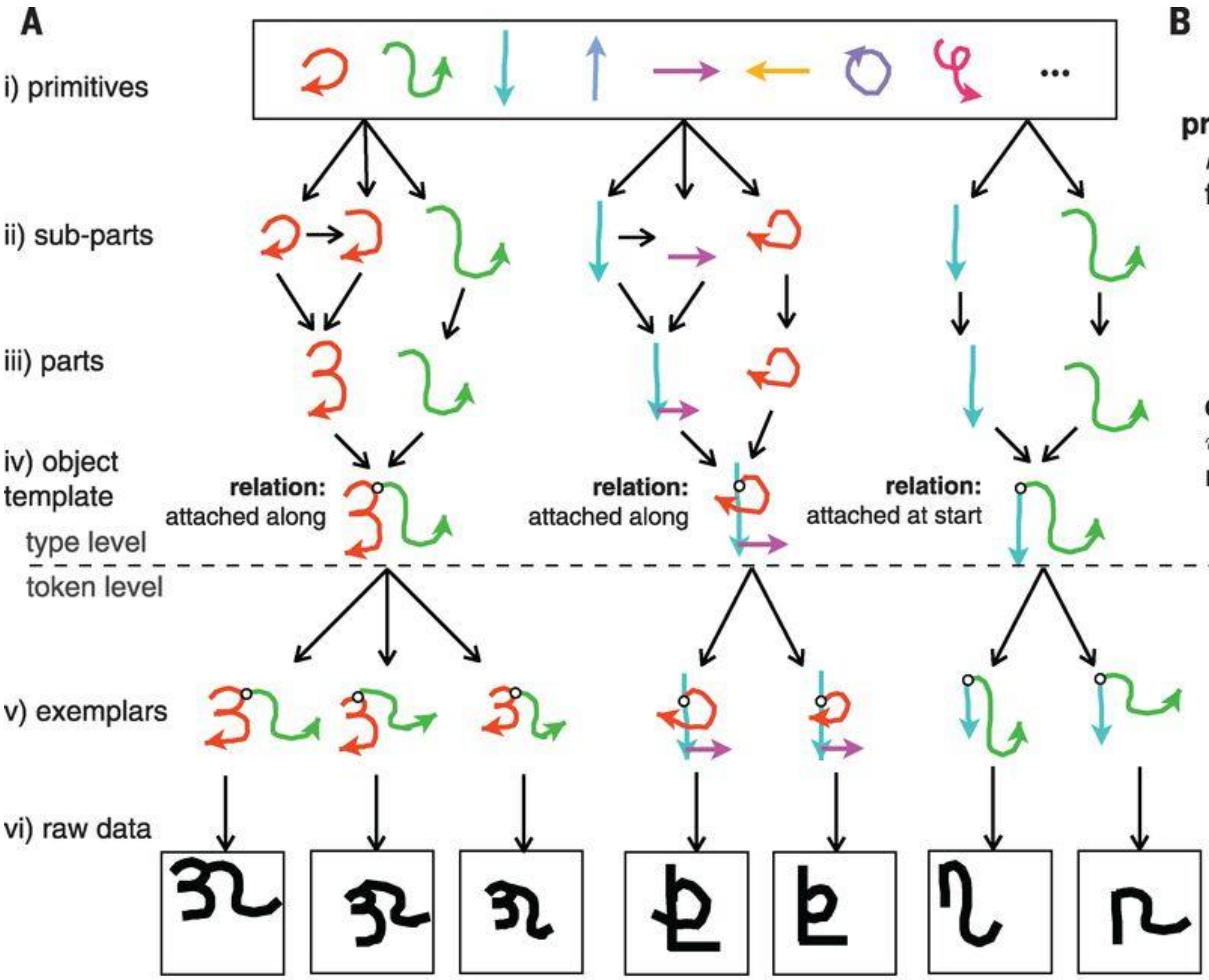
Make a *Generative* Model

A good probabilistic model is **generative**. It explains the process through which the joint is **created**.

Generative Model of Binomial Questions



Generative Model of Hand Written Letters

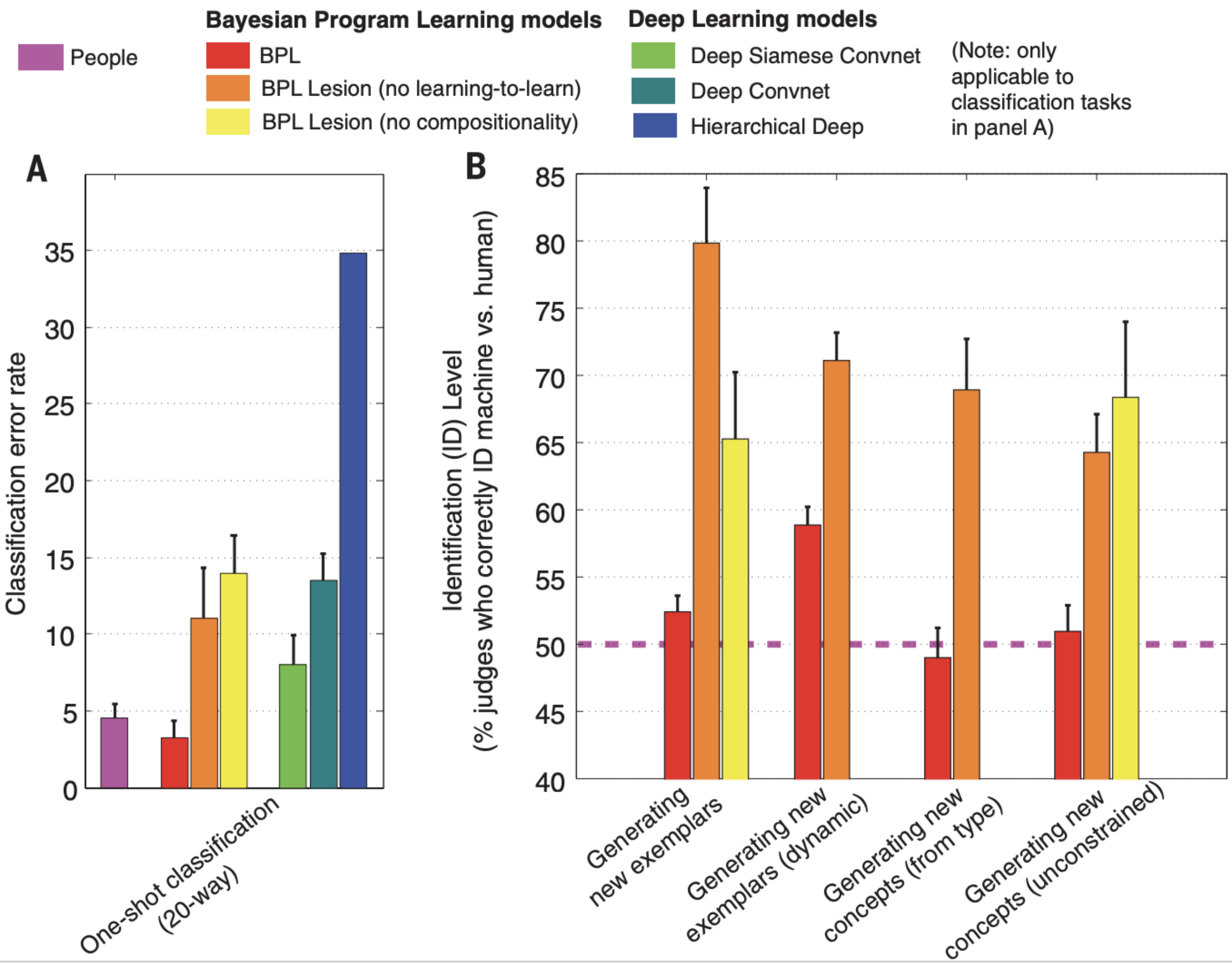


B

procedure GENERATETYPE
$\kappa \leftarrow P(\kappa)$ ▷ Sample number of parts
for i = 1 ... κ **do**
$n_i \leftarrow P(n_i|\kappa)$ ▷ Sample number of sub-parts
for j = 1 ... n_i **do**
$s_{ij} \leftarrow P(s_{ij}|s_{i(j-1)})$ ▷ Sample sub-part sequence
end for
$R_i \leftarrow P(R_i|S_1, ..., S_{i-1})$ ▷ Sample relation
end for
$\psi \leftarrow \{\kappa, R, S\}$
return @GENERATETOKEN(ψ) ▷ Return program

procedure GENERATETOKEN(ψ)
for $i = 1...\kappa$ **do**
$S_i^{(m)} \leftarrow P(S_i^{(m)}|S_i)$ ▷ Add motor variance
$L_i^{(m)} \leftarrow P(L_i^{(m)}|R_i, T_1^{(m)}, ..., T_{i-1}^{(m)})$
▷ Sample part's start location
$T_i^{(m)} \leftarrow f(L_i^{(m)}, S_i^{(m)})$ ▷ Compose a part's trajectory
end for
$A^{(m)} \leftarrow P(A^{(m)})$ ▷ Sample affine transform
$I^{(m)} \leftarrow P(I^{(m)}|T^{(m)}, A^{(m)})$ ▷ Sample image
return $I^{(m)}$

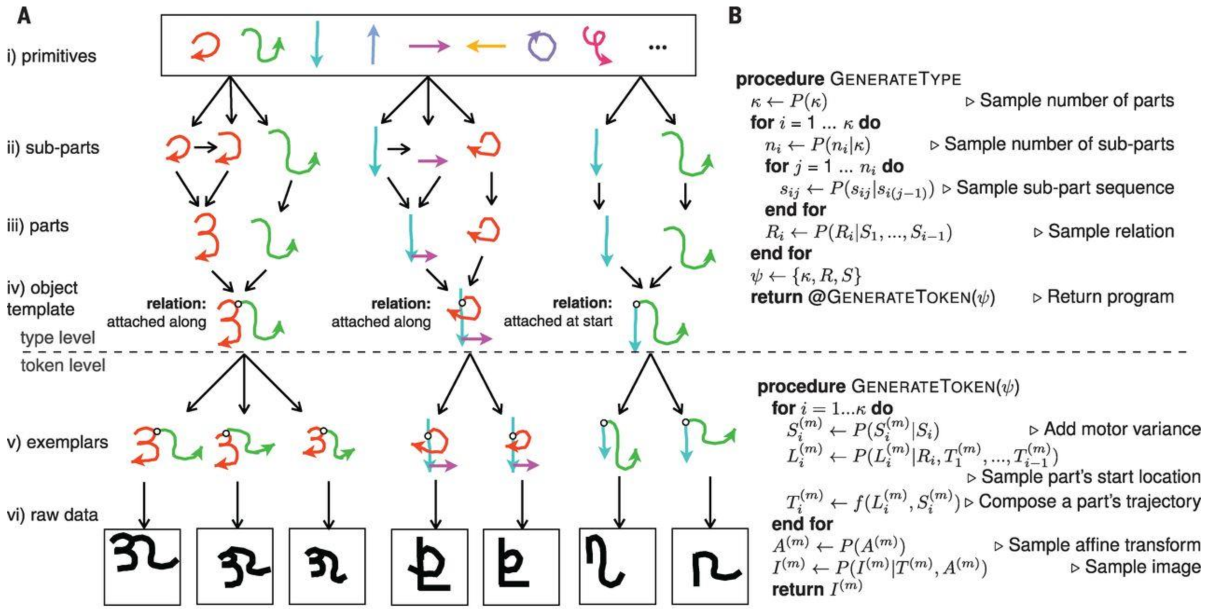
Human Level. And More!



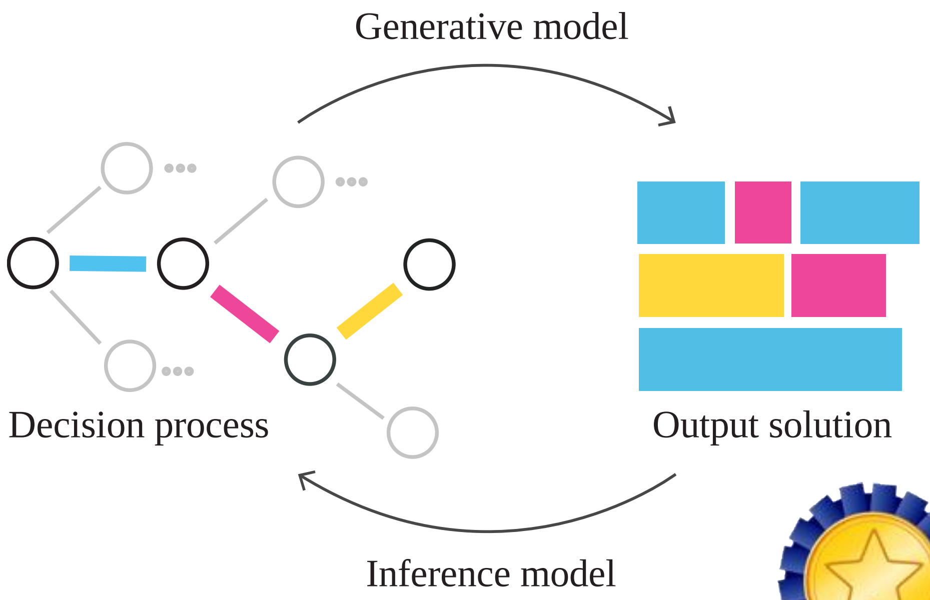
Stanford University

Generative Model of Grading

Lake et al, 2015

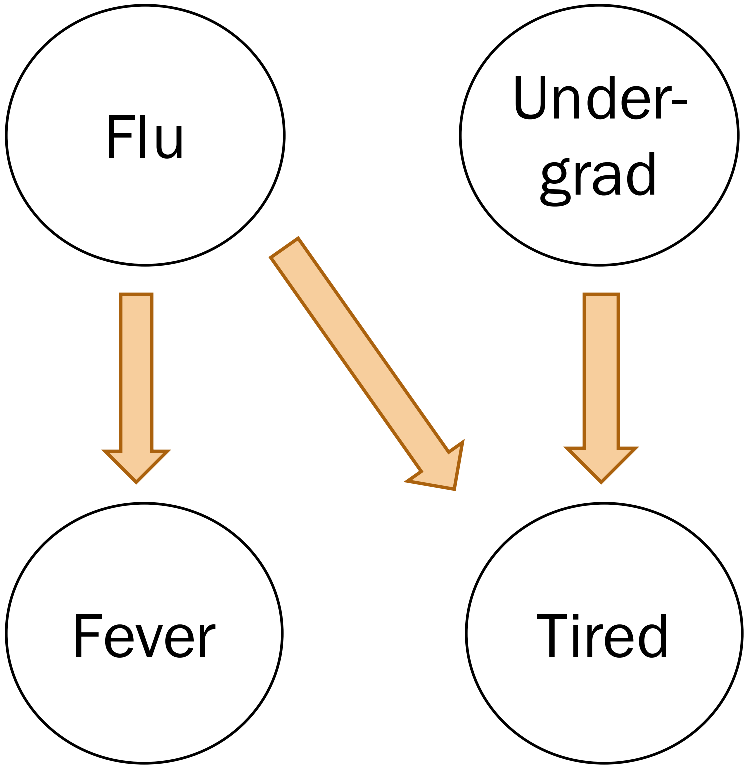


Muke Wu, Ali Malik, Noah Goodman, Chris Piech, 2019



Outstanding Paper Award, AAAI 2019

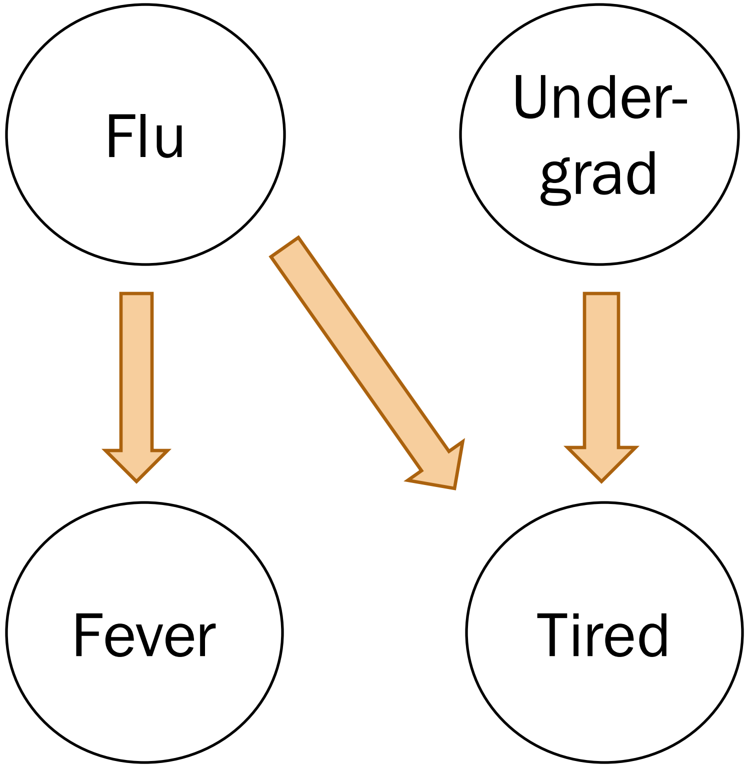
Generative Models make Independence Assumptions



What would a Stanford flu expert do?

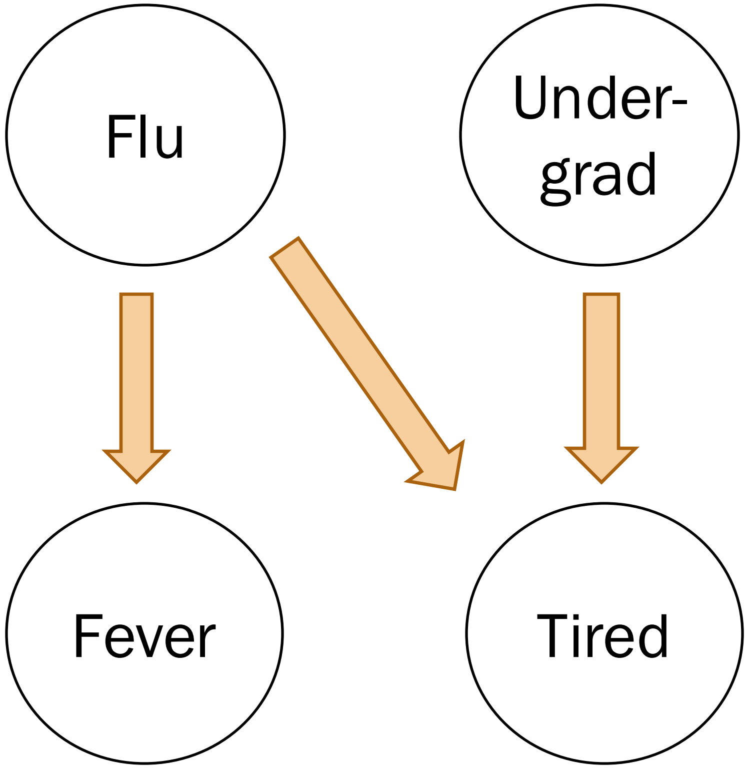
1. ☑ Describe the causality.
2. ☑ Provide $P(\text{values}|\text{causal parents})$ for each random variable
3. Implicitly assumes independences.

Generative Models make Independence Assumptions



Each random variable is **conditionally independent** of its causal non-descendants, **given its causal parents**.

Generative Models make Independence Assumptions

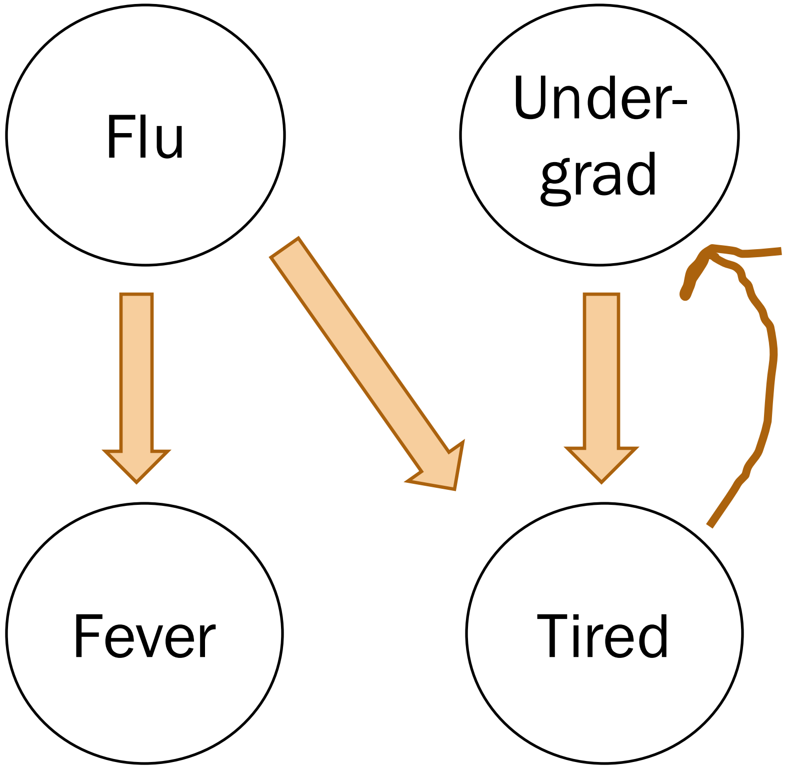


This model assumes that Flu and being an Undergraduate are independent.

Advanced: it also assumes that fever and tired are conditionally independent given Flu.

You need to tell a generative story. The independence assumptions come for free.

Bug: Constructing a Bayesian Network



Must by acyclic!

Three Guiding Questions

1. How do people define large models?
2. How can we do inference in large models?
3. What data can inform the design process?

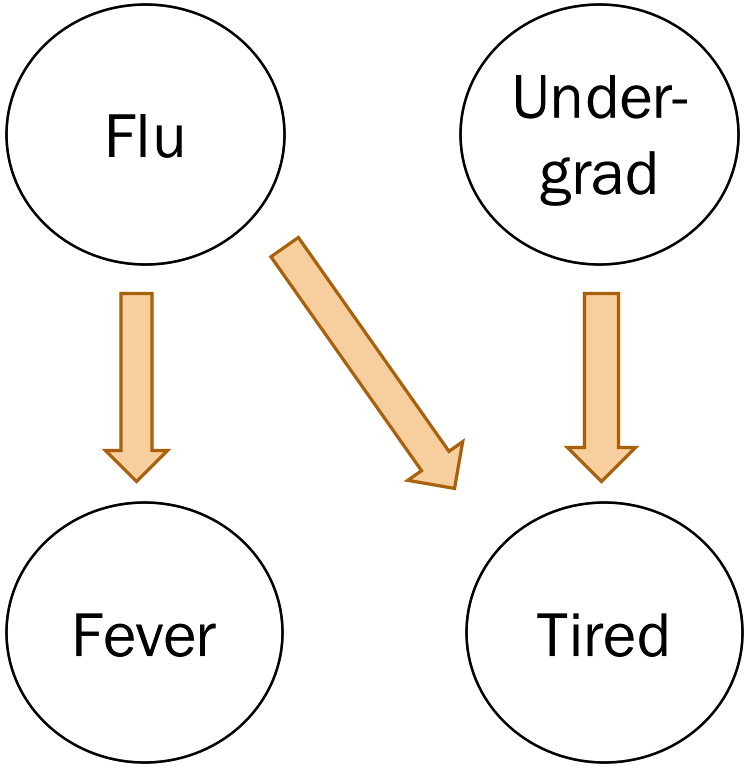
Three Guiding Questions

1. How do people define large models?
2. How can we do inference in large models?
3. What data can inform the design process?

First, the hard way

Algorithm #1: Inference via math

$P(F_{lu} = 1) = 0.1$ $P(U = 1) = 0.8$



$P(F_{ev} = 1|F_{lu} = 1) = 0.9$
$P(F_{ev} = 1|F_{lu} = 0) = 0.05$

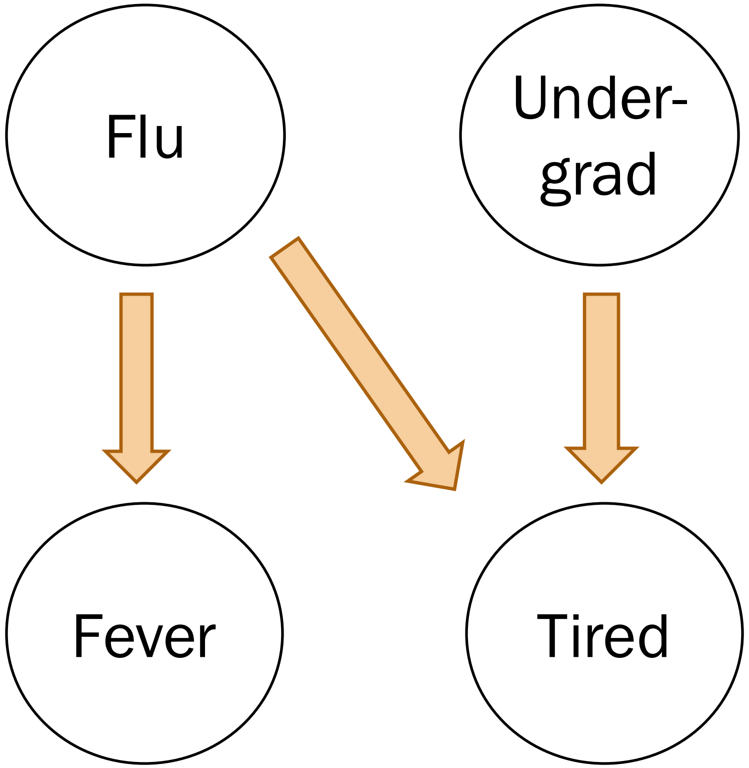
$P(T = 1|F_{lu} = 0, U = 0) = 0.1$
$P(T = 1|F_{lu} = 0, U = 1) = 0.8$
$P(T = 1|F_{lu} = 1, U = 0) = 0.9$
$P(T = 1|F_{lu} = 1, U = 1) = 1.0$

1. $P(F_{lu} = 0, U = 1, F_{ev} = 0, T = 1)$?

$$
\begin{aligned}
=&P(F_{lu} = 0) \\
&\cdot P(U = 1|F_{lu} = 0) \\
&\cdot P(F_{ev} = 0|F_{lu} = 0, U = 1) \\
&\cdot P(T = 1|F_{ev} = 0, F_{lu} = 0, U = 1)
\end{aligned}
$$

Algorithm #1: Inference via math

$P(F_{lu} = 1) = 0.1$ $P(U = 1) = 0.8$



1. $P(F_{lu} = 0, U = 1, F_{ev} = 0, T = 1)$?

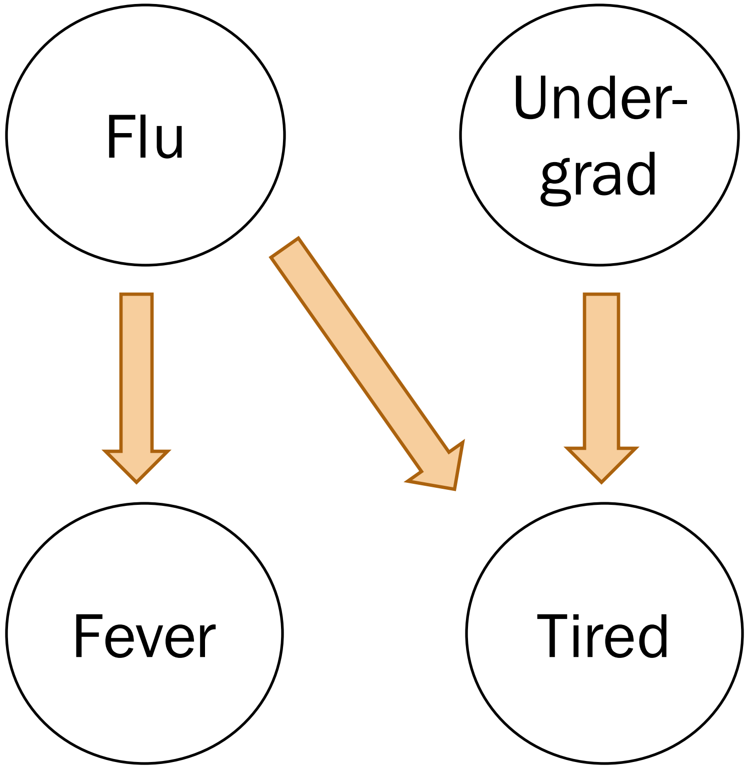
$$= P(F_{lu} = 0) \cdot P(U = 1) \cdot P(F_{ev} = 0 | F_{lu} = 0, U = 1) \cdot P(T = 1 | F_{ev} = 0, F_{lu} = 0, U = 1)$$

$P(F_{ev} = 1 | F_{lu} = 1) = 0.9$
$P(F_{ev} = 1 | F_{lu} = 0) = 0.05$

$P(T = 1 | F_{lu} = 0, U = 0) = 0.1$
$P(T = 1 | F_{lu} = 0, U = 1) = 0.8$
$P(T = 1 | F_{lu} = 1, U = 0) = 0.9$
$P(T = 1 | F_{lu} = 1, U = 1) = 1.0$

Algorithm #1: Inference via math

$P(F_{lu} = 1) = 0.1$ $\quad$ $P(U = 1) = 0.8$



$P(F_{ev} = 1 | F_{lu} = 1) = 0.9$
$P(F_{ev} = 1 | F_{lu} = 0) = 0.05$

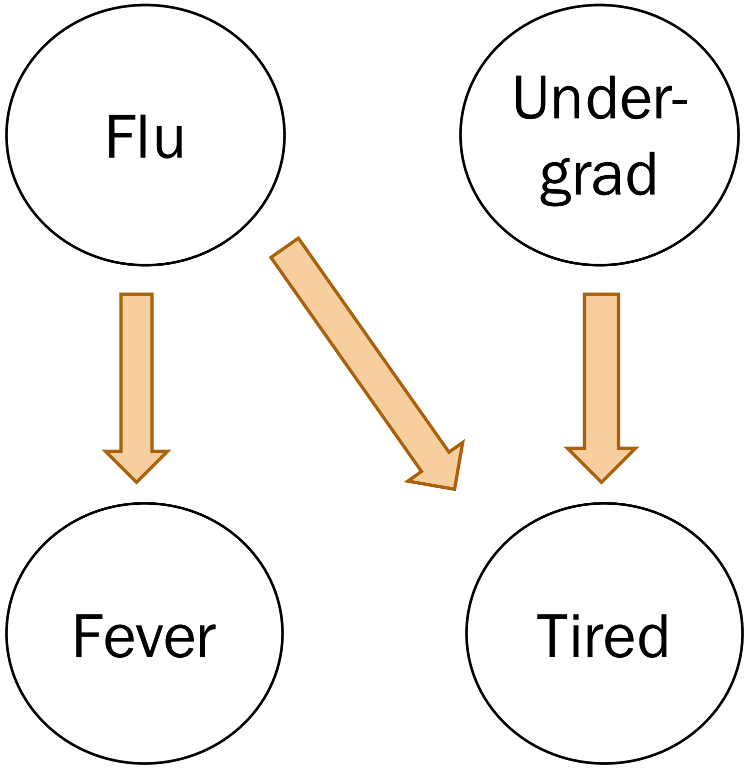
$P(T = 1 | F_{lu} = 0, U = 0) = 0.1$
$P(T = 1 | F_{lu} = 0, U = 1) = 0.8$
$P(T = 1 | F_{lu} = 1, U = 0) = 0.9$
$P(T = 1 | F_{lu} = 1, U = 1) = 1.0$

1. $P(F_{lu} = 0, U = 1, F_{ev} = 0, T = 1)$?

$$= P(F_{lu} = 0) \cdot P(U = 1) \cdot P(F_{ev} = 0 | F_{lu} = 0) \cdot P(T = 1 | F_{ev} = 0, F_{lu} = 0, U = 1)$$

Algorithm #1: Inference via math

$P(F_{lu} = 1) = 0.1$ $\quad$ $P(U = 1) = 0.8$



$P(F_{ev} = 1 | F_{lu} = 1) = 0.9$
$P(F_{ev} = 1 | F_{lu} = 0) = 0.05$

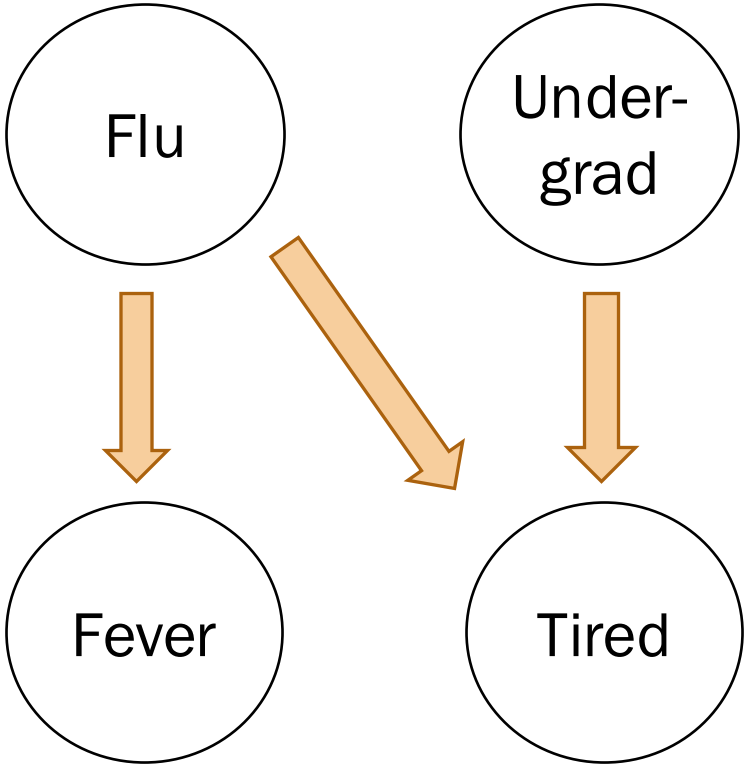
$P(T = 1 | F_{lu} = 0, U = 0) = 0.1$
$P(T = 1 | F_{lu} = 0, U = 1) = 0.8$
$P(T = 1 | F_{lu} = 1, U = 0) = 0.9$
$P(T = 1 | F_{lu} = 1, U = 1) = 1.0$

1. $P(F_{lu} = 0, U = 1, F_{ev} = 0, T = 1)$?

$$= P(F_{lu} = 0) \cdot P(U = 1) \cdot P(F_{ev} = 0 | F_{lu} = 0) \cdot P(T = 1 | F_{lu} = 0, U = 1)$$

Algorithm #1: Inference via math

$P(F_{lu} = 1) = 0.1$ $P(U = 1) = 0.8$



$P(F_{ev} = 1|F_{lu} = 1) = 0.9$
$P(F_{ev} = 1|F_{lu} = 0) = 0.05$

$P(T = 1|F_{lu} = 0, U = 0) = 0.1$
$P(T = 1|F_{lu} = 0, U = 1) = 0.8$
$P(T = 1|F_{lu} = 1, U = 0) = 0.9$
$P(T = 1|F_{lu} = 1, U = 1) = 1.0$

1. $P(F_{lu} = 0, U = 1, F_{ev} = 0, T = 1)$?

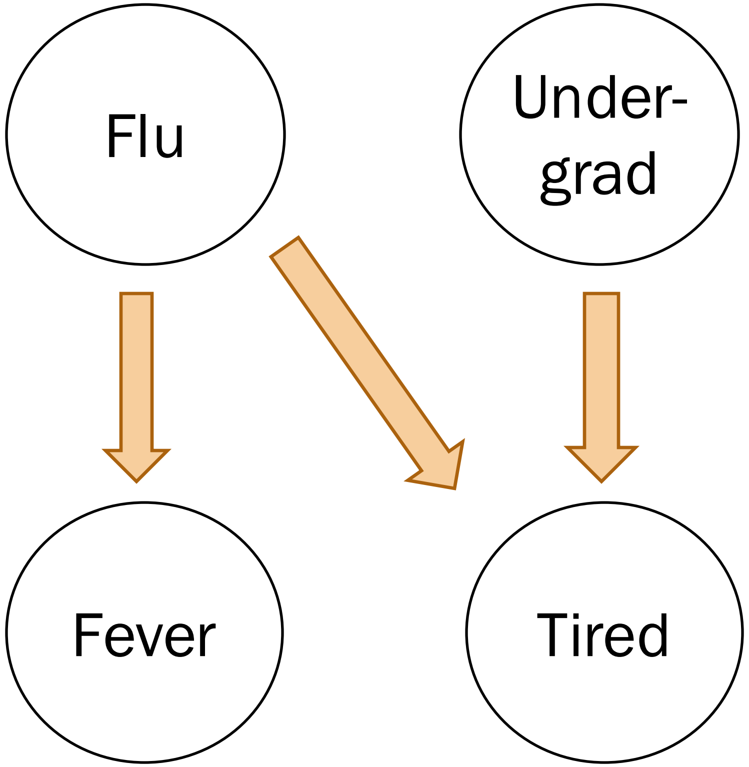
$= P(F_{lu} = 0)$ 0.9

$\cdot P(U = 1)$ 0.2

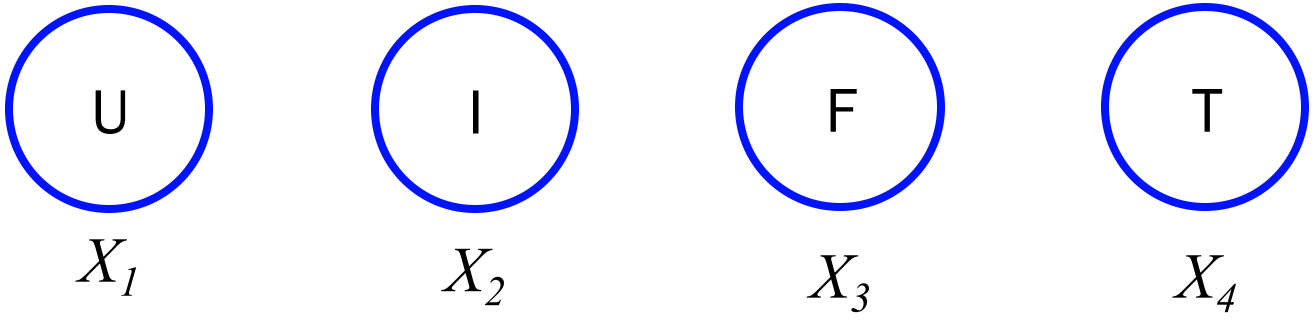
$\cdot P(F_{ev} = 0|F_{lu} = 0)$ 0.05

$\cdot P(T = 1|F_{lu} = 0, U = 1)$ 0.8

Algorithm #1: Inference via math



Order nodes by ancestry



$$P(\text{Joint}) = \prod_i P(x_i | x_{i-1}, \ldots, x_1)$$

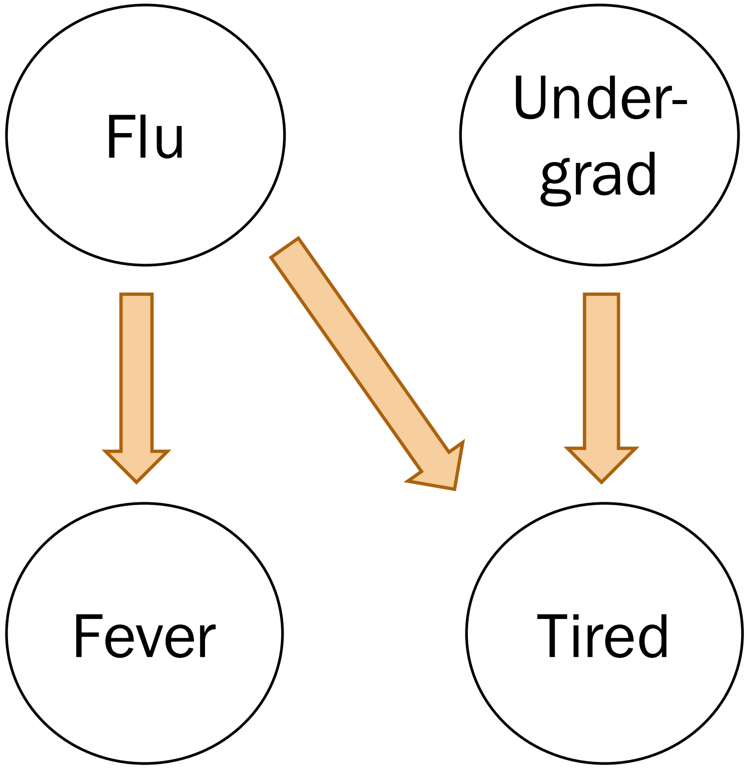
$$= \prod_i P(x_i | \text{Values of parents of } X_i)$$

Each random variable is **conditionally independent** of its causal non-descendants, **given its causal parents**.

Inference via math

$$P(\text{Joint}) = \prod_i P(x_i | \text{Values of parents of } X_i)$$

$P(F_{lu} = 1) = 0.1$ $\quad P(U = 1) = 0.8$



$P(F_{ev} = 1 | F_{lu} = 1) = 0.9$
$P(F_{ev} = 1 | F_{lu} = 0) = 0.05$

$P(T = 1 | F_{lu} = 0, U = 0) = 0.1$
$P(T = 1 | F_{lu} = 0, U = 1) = 0.8$
$P(T = 1 | F_{lu} = 1, U = 0) = 0.9$
$P(T = 1 | F_{lu} = 1, U = 1) = 1.0$

2. $P(F_{lu} = 1 | F_{ev} = 0, U = 0, T = 1)$?

1. Compute joint probabilities

$$P(F_{lu} = 1, F_{ev} = 0, U = 0, T = 1)$$
$$P(F_{lu} = 0, F_{ev} = 0, U = 0, T = 1)$$

2. Definition of conditional probability

$$\frac{P(F_{lu} = 1, F_{ev} = 0, U = 0, T = 1)}{\sum_x P(F_{lu} = x, F_{ev} = 0, U = 0, T = 1)}$$

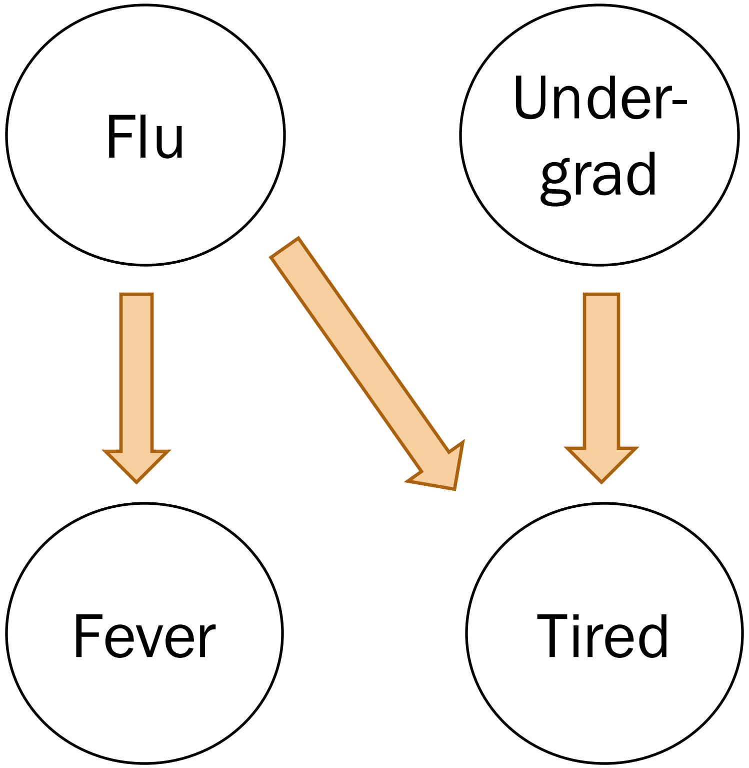
$$= 0.095$$

Want more practice?

Inference via math

$$P(\text{Joint}) = \prod_i P(x_i | \text{Values of parents of } X_i)$$

$P(F_{lu} = 1) = 0.1$ $\quad$ $P(U = 1) = 0.8$



$P(F_{ev} = 1 | F_{lu} = 1) = 0.9$
$P(F_{ev} = 1 | F_{lu} = 0) = 0.05$

$P(T = 1 | F_{lu} = 0, U = 0) = 0.1$
$P(T = 1 | F_{lu} = 0, U = 1) = 0.8$
$P(T = 1 | F_{lu} = 1, U = 0) = 0.9$
$P(T = 1 | F_{lu} = 1, U = 1) = 1.0$

3. $P(F_{lu} = 1 | U = 1, T = 1)$?

   1. Compute joint probabilities

      $$P(F_{lu} = 1, U = 1, F_{ev} = 1, T = 1)$$

      ...

      $$P(F_{lu} = 0, U = 1, F_{ev} = 0, T = 1)?$$

   2. Definition of conditional probability

      $$\frac{\sum_y P(F_{lu} = 1, U = 1, F_{ev} = y, T = 1)}{\sum_x \sum_y P(F_{lu} = x, U = 1, F_{ev} = y, T = 1)}$$

$$= 0.122$$

There must be a better way…

[suspense]

Algorithm #2: Rejection Sampling

```python
def main():
    obs = get_observation()
    samples = sample_a_ton()
    prob = prob_flu_given_obs(samples, obs)
    print('Observation = ', obs)
    print('Pr(Flu | Obs) = ', prob)
```

Algorithm #2: Rejection Sampling

```python
def main():
    obs = get_observation()
    samples = sample_a_ton()
    prob = prob_flu_given_obs(sample
    print('Observation = ', obs)
    print('Pr(Flu | Obs) = ', prob)
```

webMd — -zsh — 56×42


```
{'flu': 1, 'undergrad': 1, 'fever': 1, 'tired': 1}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 0}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 1}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 1}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 1}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 1}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 1}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 1}
{'flu': 0, 'undergrad': 0, 'fever': 0, 'tired': 0}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 0}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 1}
{'flu': 0, 'undergrad': 0, 'fever': 0, 'tired': 0}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 1}
{'flu': 1, 'undergrad': 1, 'fever': 1, 'tired': 1}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 1}
{'flu': 0, 'undergrad': 1, 'fever': 1, 'tired': 1}
{'flu': 0, 'undergrad': 0, 'fever': 0, 'tired': 0}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 1}
{'flu': 0, 'undergrad': 0, 'fever': 0, 'tired': 0}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 1}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 1}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 1}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 1}
{'flu': 1, 'undergrad': 1, 'fever': 1, 'tired': 1}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 0}
{'flu': 0, 'undergrad': 1, 'fever': 1, 'tired': 1}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 1}
{'flu': 0, 'undergrad': 0, 'fever': 0, 'tired': 0}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 0}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 1}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 1}
{'flu': 1, 'undergrad': 1, 'fever': 1, 'tired': 1}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 1}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 1}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 1}
{'flu': 0, 'undergrad': 0, 'fever': 0, 'tired': 0}
{'flu': 0, 'undergrad': 0, 'fever': 0, 'tired': 0}
{'flu': 0, 'undergrad': 1, 'fever': 1, 'tired': 0}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 0}
{'flu': 0, 'undergrad': 1, 'fever': 1, 'tired': 1}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 1}
{'flu': 1, 'undergrad': 1, 'fever': 1, 'tired': 1}
```

Algorithm #2: Rejection Sampling

```python
def main():
    obs = get_observation()
    samples = sample_a_ton()
    prob = prob_flu_given_obs(samples, obs)
    print('Observation = ', obs)
    print('Pr(Flu | Obs) = ', prob)
```

Algorithm #2: Rejection Sampling

```python
def prob_flu_given_obs(samples, obs):
    """
    Calculate the probability of flu given many
    samples from the joint distribution and a set
    of ovservations to condition on.
    """
    # reject all samples which don't align
    # with condition
    keep_samples = []
    for sample in samples:
        if check_obs_match(sample, obs):
            keep_samples.append(sample)

    # from remaining, simply count...
    flu_count = 0
    for sample in keep_samples:
        if sample['flu'] == 1:
            flu_count += 1

    # counting can be so sweet...
    return float(flu_count) / len(keep_samples)
```

Algorithm #2: Rejection Sampling

```python
def prob_flu_given_obs(samples, obs):
    """
    Calculate the probability of flu given many
    samples from the joint distribution and a set
    of ovservations to condition on.
    """
    # reject all samples which don't align
    # with condition
    keep_samples = []
    for sample in samples:
        if check_obs_match(sample, obs):
            keep_samples.append(sample)

    # from remaining, simply count...
    flu_count = 0
    for sample in keep_samples:
        if sample['flu'] == 1:
            flu_count += 1

    # counting can be so sweet...
    return float(flu_count) / len(keep_samples)
```

Algorithm #2: Rejection Sampling

```python
def prob_flu_given_obs(samples, obs):
    """
    Calculate the probability of flu given many
    samples from the joint distribution and a set
    of ovservations to condition on.
    """
    # reject all samples which don't align
    # with condition
    keep_samples = []
    for sample in samples:
        if check_obs_match(sample, obs):
            keep_samples.append(sample)

    # from remaining, simply count...
    flu_count = 0
    for sample in keep_samples:
        if sample['flu'] == 1:
            flu_count += 1

    # counting can be so sweet...
    return float(flu_count) / len(keep_samples)
```

Algorithm #2: Rejection Sampling

```python
def prob_flu_given_obs(samples, obs):
    """
    Calculate the probability of flu given ma
    samples from the joint distribution and a
    of ovservations to condition on.
    """
    # reject all samples which don't align
    # with condition
    keep_samples = []
    for sample in samples:
        if check_obs_match(sample, obs):
            keep_samples.append(sample)

    # from remaining, simply count...
    flu_count = 0
    for sample in keep_samples:
        if sample['flu'] == 1:
            flu_count += 1

    # counting can be so sweet...
    return float(flu_count) / len(keep_samples)
```

webMd — -zsh — 53×25

```
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 1}
{'flu': 0, 'undergrad': 0, 'fever': 0, 'tired': 0}
{'flu': 0, 'undergrad': 0, 'fever': 0, 'tired': 0}
{'flu': 0, 'undergrad': 1, 'fever': 1, 'tired': 0}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 0}
{'flu': 0, 'undergrad': 1, 'fever': 1, 'tired': 1}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 1}
{'flu': 1, 'undergrad': 1, 'fever': 1, 'tired': 1}
{'flu': 1, 'undergrad': 1, 'fever': 1, 'tired': 1}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 1}
{'flu': 0, 'undergrad': 0, 'fever': 0, 'tired': 0}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 1}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 1}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 1}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 0}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 0}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 1}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 1}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 0}
{'flu': 0, 'undergrad': 0, 'fever': 0, 'tired': 0}
----------
Observation =  {'flu': None, 'undergrad': 1, 'fever':
 None, 'tired': 1}
Pr(Flu | Obs) =  0.1228646517739816
piech@Chriss-MBP-5 webMd %
```

Lets try it!

BACK TO THE CODE

Rejection sampling algorithm

Inference question: What is $P(F_{lu} = 1 | U = 1, T = 1)$?

$$\text{probability} \approx \frac{\#\text{ samples with } (F_{lu} = 1, U = 1, T = 1)}{\#\text{ samples with } (U = 1, T = 1)}$$

Why would this definition of approximate probability make sense?

Why would this approximate probability make sense?

Inference question: What is $P(F_{lu} = 1 | U = 1, T = 1)$?

$$\text{probability} \approx \frac{\text{\# samples with } (F_{lu} = 1, U = 1, T = 1)}{\text{\# samples with } (U = 1, T = 1)}$$

Recall our definition of probability as a frequency:

$$P(E) = \lim_{n \to \infty} \frac{n(E)}{n}$$

n = # of total trials
$n(E)$ = # trials where E occurs

webMd — -zsh — 53×25

```
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 1}
{'flu': 0, 'undergrad': 0, 'fever': 0, 'tired': 0}
{'flu': 0, 'undergrad': 0, 'fever': 0, 'tired': 0}
{'flu': 0, 'undergrad': 1, 'fever': 1, 'tired': 0}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 0}
{'flu': 0, 'undergrad': 1, 'fever': 1, 'tired': 1}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 1}
{'flu': 1, 'undergrad': 1, 'fever': 1, 'tired': 1}
{'flu': 1, 'undergrad': 1, 'fever': 1, 'tired': 1}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 1}
{'flu': 0, 'undergrad': 0, 'fever': 0, 'tired': 0}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 1}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 1}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 1}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 0}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 0}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 1}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 1}
{'flu': 0, 'undergrad': 1, 'fever': 0, 'tired': 0}
{'flu': 0, 'undergrad': 0, 'fever': 0, 'tired': 0}
----------
Observation =  {'flu': None, 'undergrad': 1, 'fever':
 None, 'tired': 1}
Pr(Flu | Obs) =  0.1228646517739816
piech@Chriss-MBP-5 webMd %
```

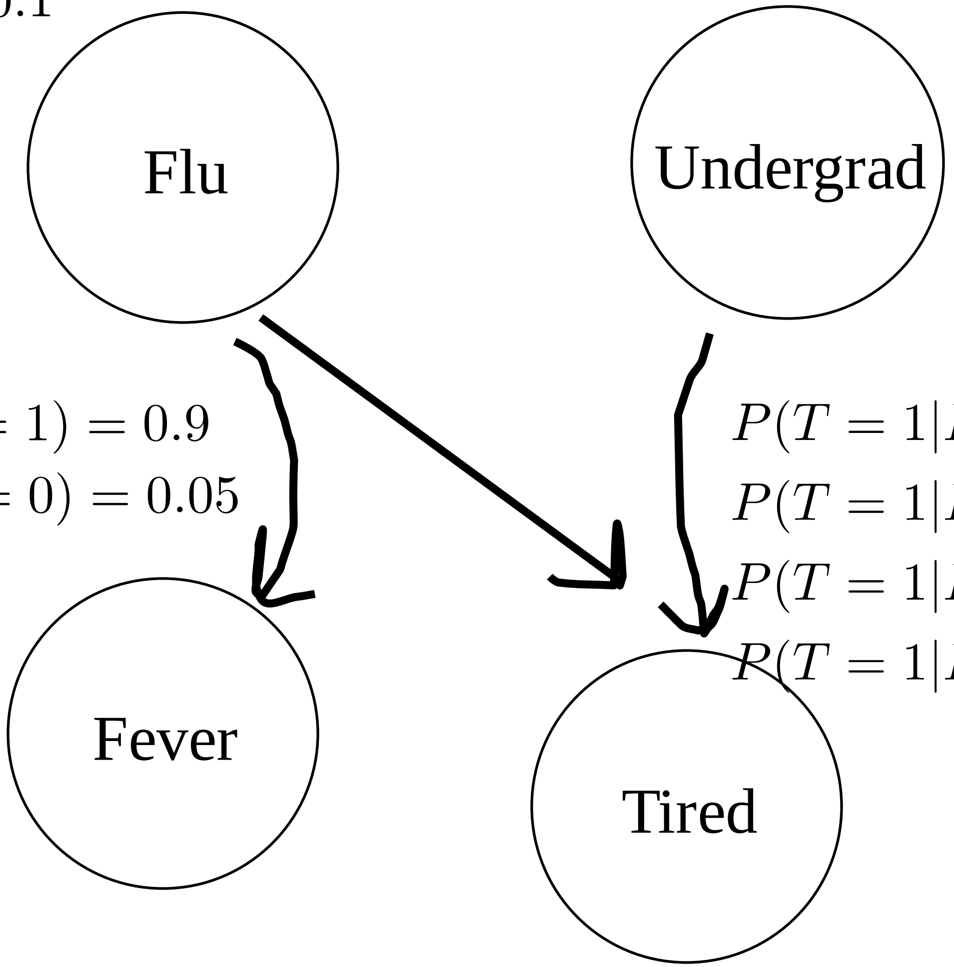
If you can sample enough from the joint distribution, you can answer any probability question

Each one of these is one joint sample

What's the matter with

rejection sampling?

Probabilistic Model



$$P(Fl = 1) = 0.1$$

$$P(U = 1) = 0.8$$

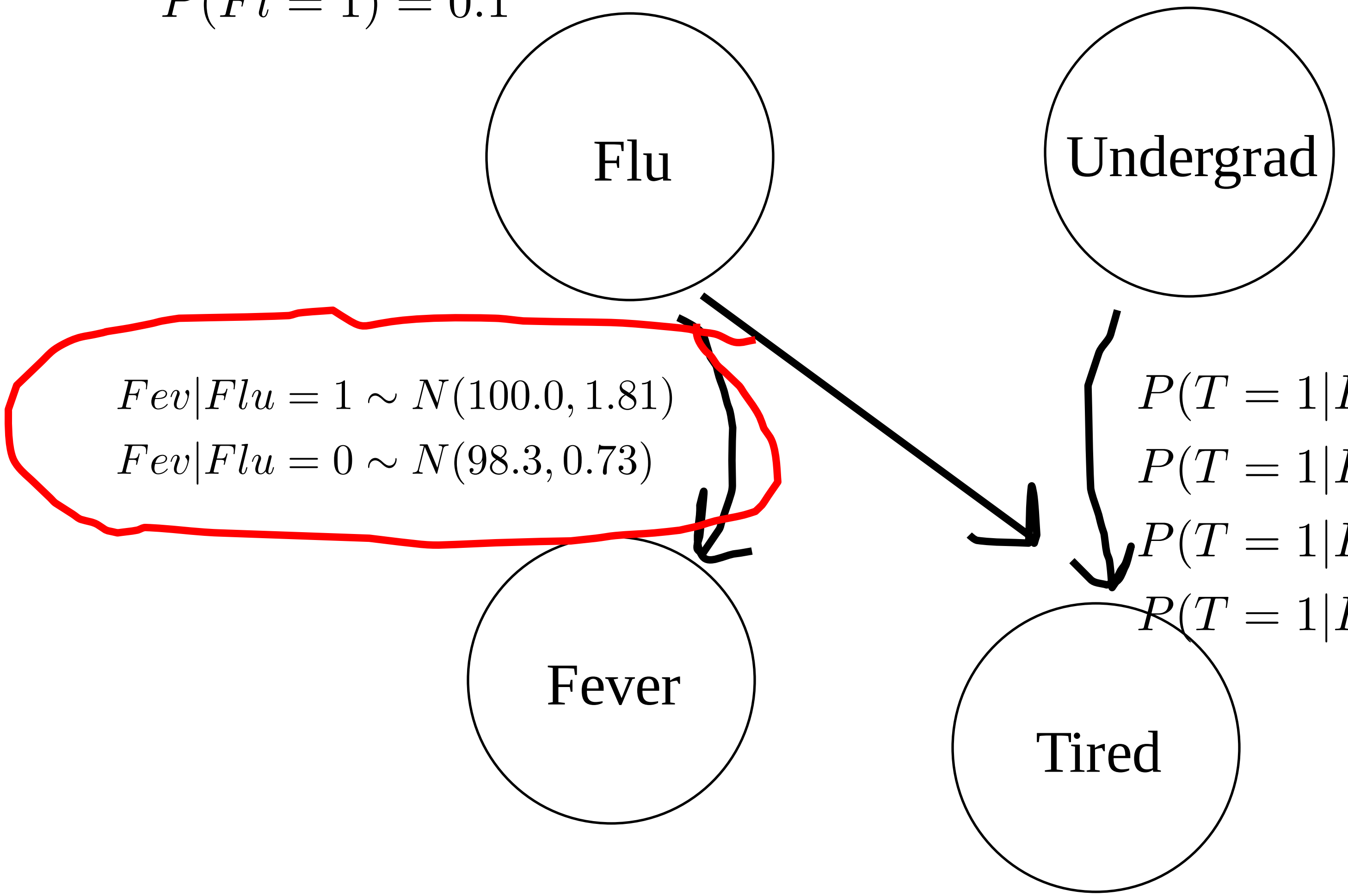
$$P(Fev = 1 | Flu = 1) = 0.9$$
$$P(Fev = 1 | Flu = 0) = 0.05$$

$$P(T = 1 | Flu = 0, U = 0) = 0.1$$
$$P(T = 1 | Flu = 0, U = 1) = 0.8$$
$$P(T = 1 | Flu = 1, U = 0) = 0.9$$
$$P(T = 1 | Flu = 1, U = 1) = 1.0$$

Probabilistic Model

$$P(Fl = 1) = 0.1$$

$$P(U = 1) = 0.8$$



$$Fev|Flu = 1 \sim N(100.0, 1.81)$$
$$Fev|Flu = 0 \sim N(98.3, 0.73)$$

$$P(T = 1|Flu = 0, U = 0) = 0.1$$
$$P(T = 1|Flu = 0, U = 1) = 0.8$$
$$P(T = 1|Flu = 1, U = 0) = 0.9$$
$$P(T = 1|Flu = 1, U = 1) = 1.0$$


The Magic School Bus™
Blast Off!

Many Algorithms

Markov Chain

MCMC

Monte Carlo

Many Algorithms

```
webMd — -bash — 20×20
[1, 1, 101.0, 1]
[1, 1, 101.0, 1]
[0, 1, 101.0, 0]
[0, 0, 101.0, 0]
[1, 0, 101.0, 1]
[1, 0, 101.0, 0]
[1, 0, 101.0, 1]
[1, 0, 101.0, 1]
[1, 1, 101.0, 1]
[1, 1, 101.0, 1]
[1, 1, 101.0, 1]
[1, 1, 101.0, 1]
[1, 1, 101.0, 1]
[1, 1, 101.0, 1]
[1, 1, 101.0, 1]
[1, 0, 101.0, 1]
[1, 1, 101.0, 1]
[1, 1, 101.0, 1]
Pr(Flu) = 0.9773
>
```

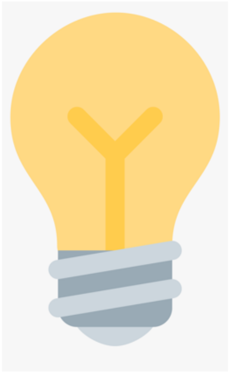
MCMC is a way to sample with conditioned variables fixed

Each one of these is one posterior sample:

[Flu, Undergrad, Fever, Tired]

Many Algorithms

Rejection Sampling	MCMC	Pyro	Idea2Text



Three Guiding Questions

1. How do people define large models?

2. How can we do inference in large models?

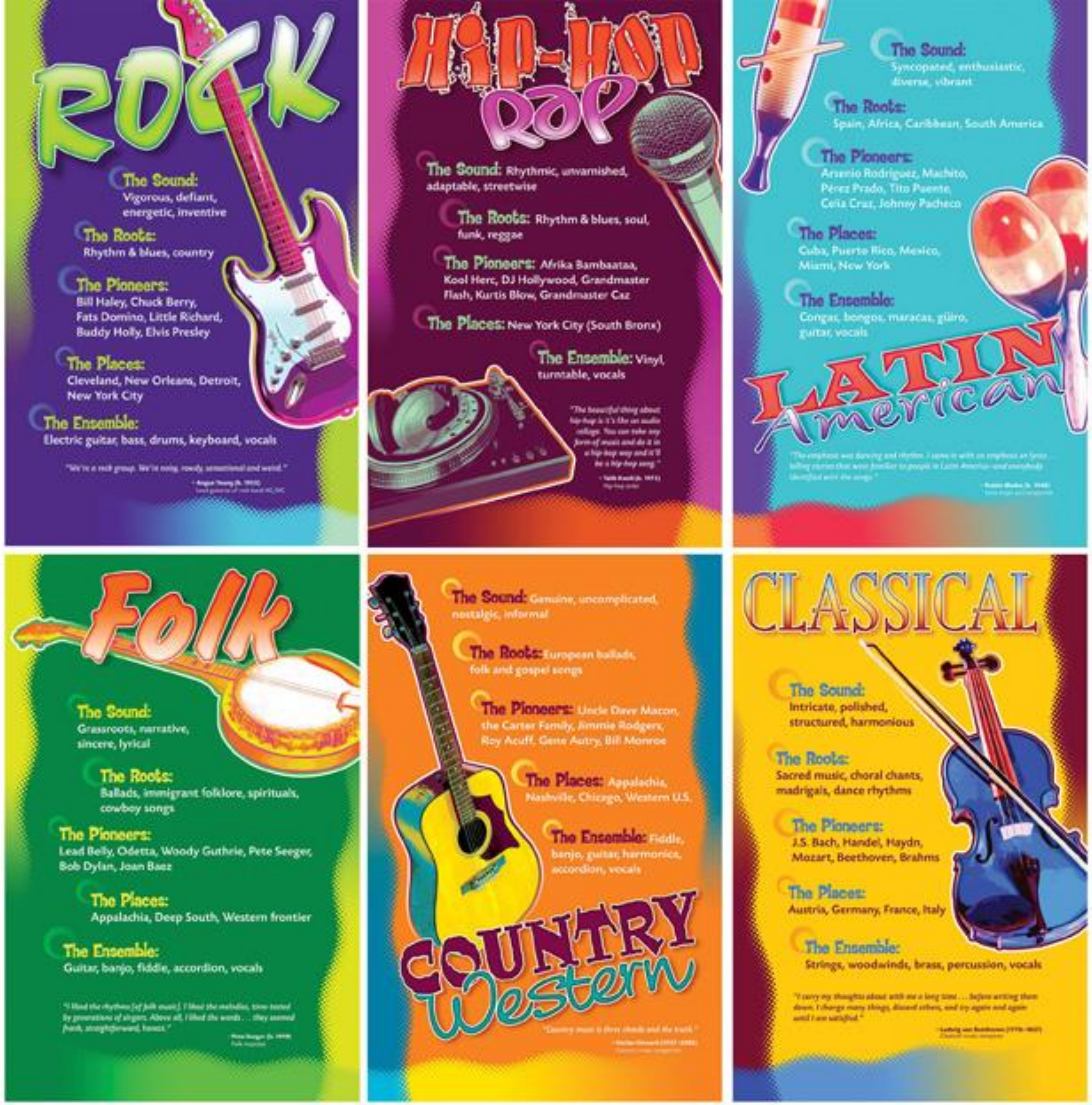
3. What data can inform the design process?

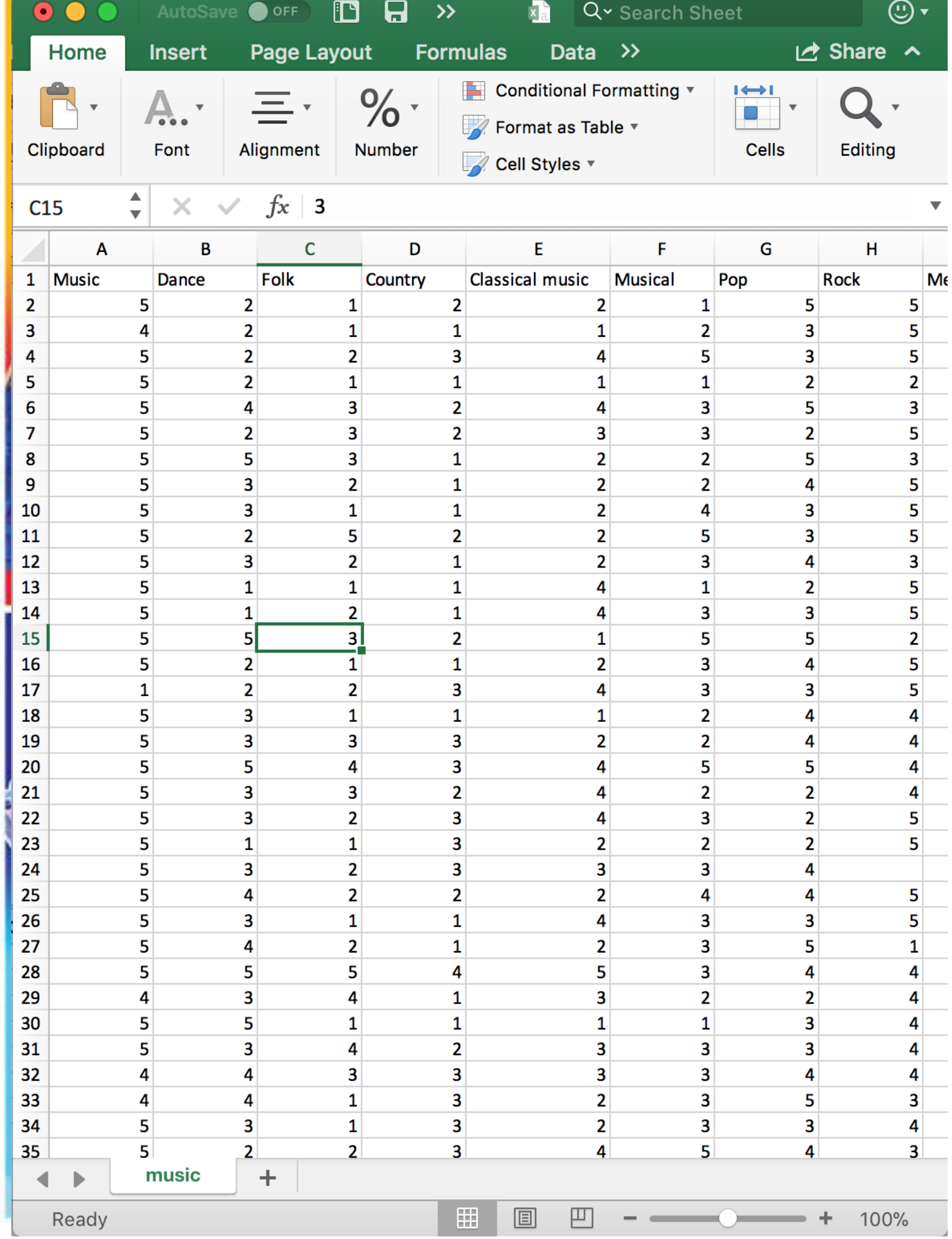
Three Guiding Questions

1. How do people define large models?

2. How can we do inference in large models?

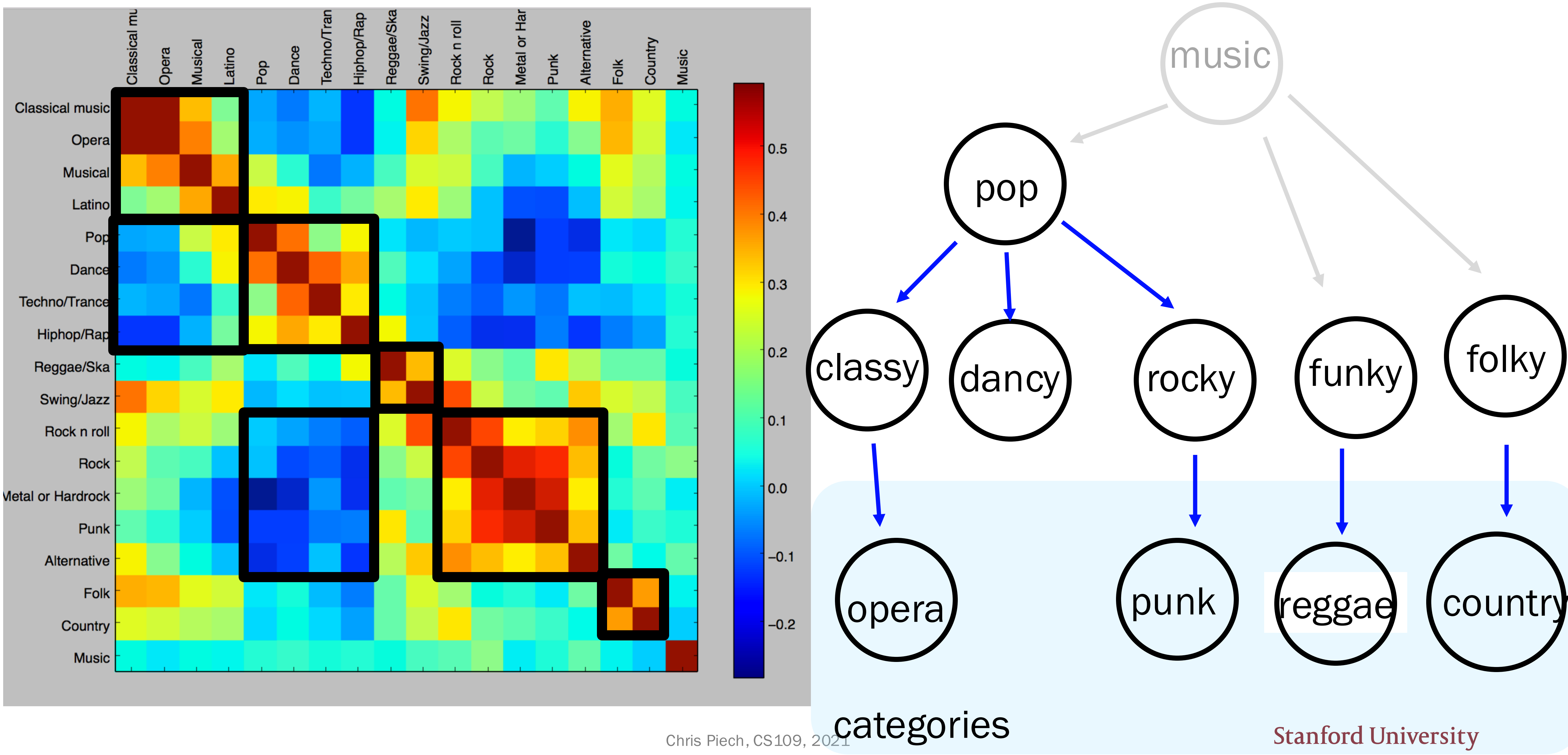
3. What data can inform the design process?



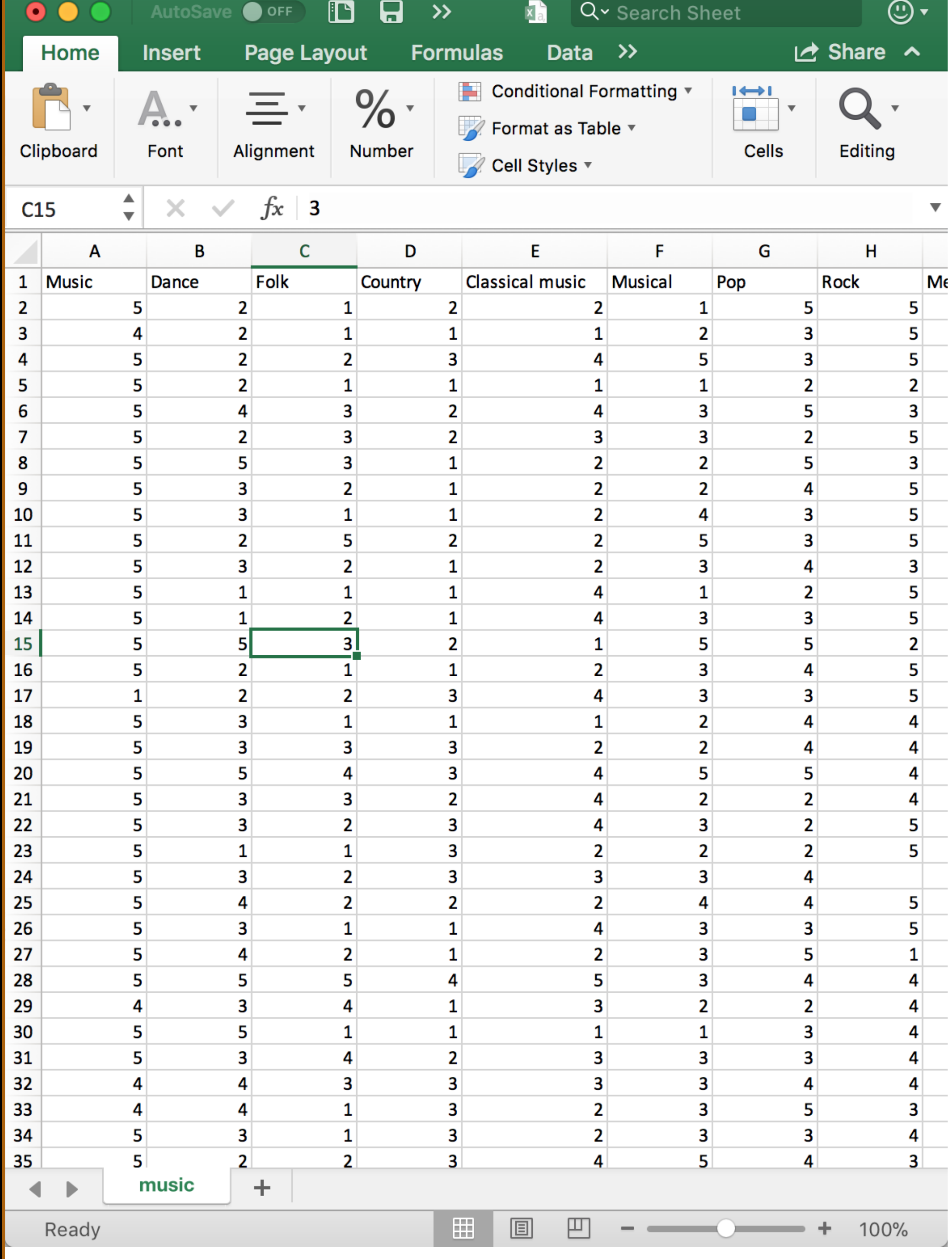


	A	B	C	D	E	F	G	H
1	Music	Dance	Folk	Country	Classical music	Musical	Pop	Rock
2	5	2	1	2	2	1	5	5
3	4	2	1	1	1	2	3	5
4	5	2	2	3	4	5	3	5
5	5	2	1	1	1	1	2	2
6	5	4	3	2	4	3	5	3
7	5	2	3	2	3	3	2	5
8	5	5	3	1	2	2	5	3
9	5	3	2	1	2	2	4	5
10	5	3	1	1	2	4	3	5
11	5	2	5	2	2	5	3	5
12	5	3	2	1	2	3	4	3
13	5	1	1	1	4	1	2	5
14	5	1	2	1	4	3	3	5
15	5	5	3	2	1	5	5	2
16	5	2	1	1	2	3	4	5
17	1	2	2	3	4	3	3	5
18	5	3	1	1	1	2	4	4
19	5	3	3	3	2	2	4	4
20	5	5	4	3	4	5	5	4
21	5	3	3	2	4	2	2	4
22	5	3	2	3	4	3	2	5
23	5	1	1	3	2	2	2	5
24	5	3	2	3	3	3	4	
25	5	4	2	2	2	4	4	5
26	5	3	1	1	4	3	3	5
27	5	4	2	1	2	3	5	1
28	5	5	5	4	5	3	4	4
29	4	3	4	1	3	2	2	4
30	5	5	1	1	1	1	3	4
31	5	3	4	2	3	3	3	4
32	4	4	3	3	3	3	4	4
33	4	4	1	3	2	3	5	3
34	5	3	1	3	2	3	3	4
35	5	2	2	3	4	5	4	3

From Correlation to Bayes Net!



Why is it harder to find independences here than for bat DNA expression?



	Music	Dance	Folk	Country	Classical music	Musical	Pop	Rock
2	5	2	1	2	2	1	5	5
3	4	2	1	1	1	2	3	5
4	5	2	2	3	4	5	3	5
5	5	2	1	1	1	1	2	2
6	5	4	3	2	4	3	5	3
7	5	2	3	2	3	3	2	5
8	5	5	3	1	2	2	5	3
9	5	3	2	1	2	2	4	5
10	5	3	1	1	2	4	3	5
11	5	2	5	2	2	5	3	5
12	5	3	2	1	2	3	4	3
13	5	1	1	1	4	1	2	5
14	5	1	2	1	4	3	3	5
15	5	5	3	2	1	5	5	2
16	5	2	1	1	2	3	4	5
17	1	2	2	3	4	3	3	5
18	5	3	1	1	1	2	4	4
19	5	3	3	3	2	2	4	4
20	5	5	4	3	4	5	5	4
21	5	3	3	2	4	2	2	4
22	5	3	2	3	4	3	2	5
23	5	1	1	3	2	2	2	5
24	5	3	2	3	3	3	4	
25	5	4	2	2	2	4	4	5
26	5	3	1	1	4	3	3	5
27	5	4	2	1	2	3	5	1
28	5	5	5	4	5	3	4	4
29	4	3	4	1	3	2	2	4
30	5	5	1	1	1	1	3	4
31	5	3	4	2	3	3	3	4
32	4	4	3	3	3	3	4	4
33	4	4	1	3	2	3	5	3
34	5	3	1	3	2	3	3	4
35	5	2	2	3	4	5	4	3

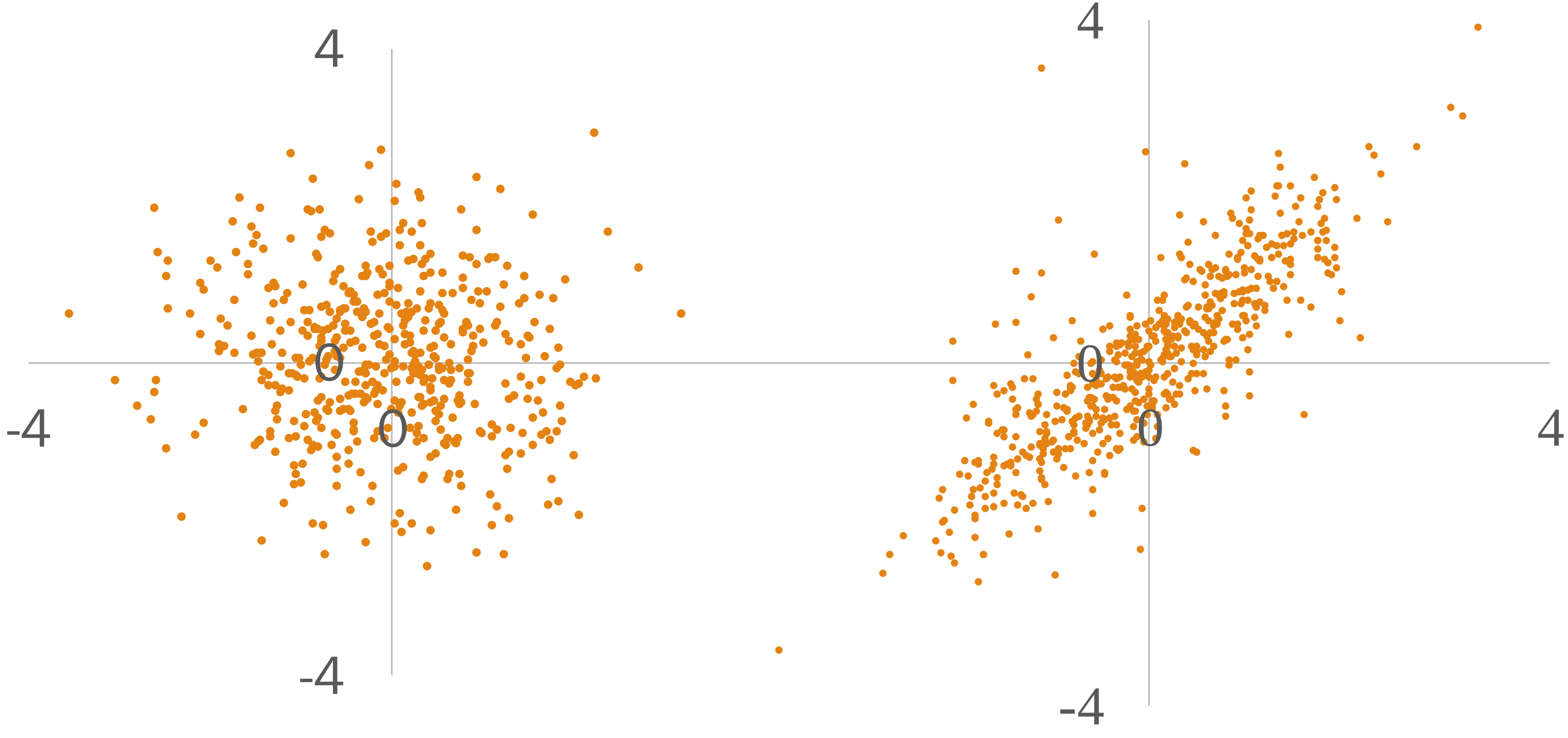
Bat Data

Gene1	Gene2	Gene3	Gene4	Gene5	Trait
TRUE	FALSE	TRUE	TRUE	FALSE	FALSE
FALSE	FALSE	TRUE	TRUE	TRUE	TRUE
TRUE	FALSE	TRUE	FALSE	FALSE	FALSE
TRUE	FALSE	TRUE	TRUE	TRUE	FALSE
FALSE	TRUE	TRUE	TRUE	TRUE	TRUE
FALSE	FALSE	FALSE	TRUE	FALSE	FALSE
TRUE	FALSE	FALSE	TRUE	FALSE	FALSE
TRUE	FALSE	FALSE	TRUE	FALSE	FALSE
TRUE	FALSE	TRUE	FALSE	FALSE	FALSE
FALSE	TRUE	FALSE	TRUE	FALSE	FALSE
TRUE	TRUE	FALSE	TRUE	FALSE	FALSE
TRUE	FALSE	FALSE	TRUE	FALSE	FALSE
TRUE	FALSE	TRUE	TRUE	TRUE	FALSE
FALSE	FALSE	TRUE	TRUE	FALSE	FALSE
TRUE	FALSE	FALSE	TRUE	FALSE	FALSE
TRUE	FALSE	FALSE	TRUE	FALSE	FALSE
		…			
TRUE	FALSE	FALSE	TRUE	FALSE	FALSE

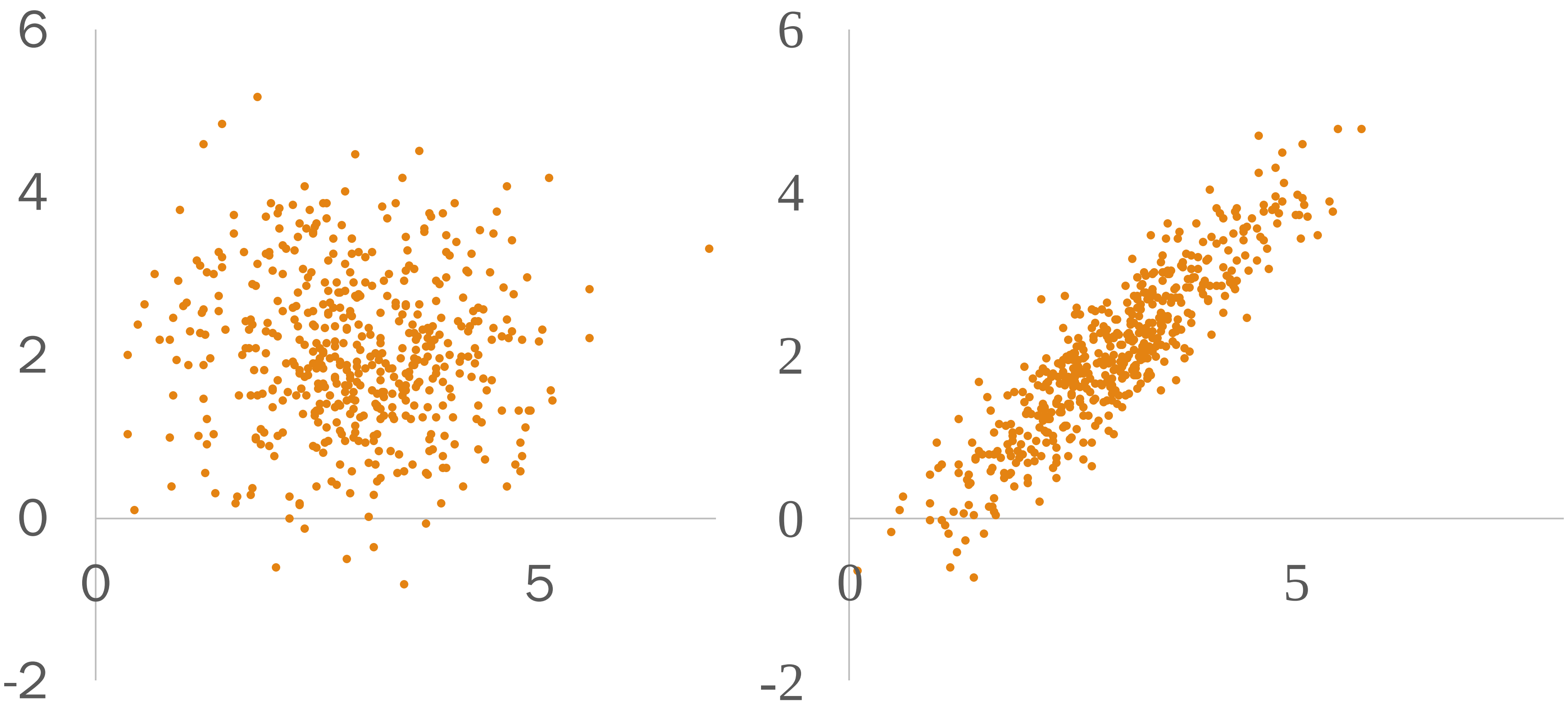
Expression Amount

Gene5	Trait
0.76	0.83
0.94	0.85
0.82	0.03
0.94	0.32
0.50	0.10
0.40	0.53
0.90	0.67
0.29	0.71
0.72	0.25
0.15	0.24
0.79	0.98
0.68	0.77
0.71	0.37
0.36	0.18
0.62	0.08
0.59	0.38
0.82	0.76

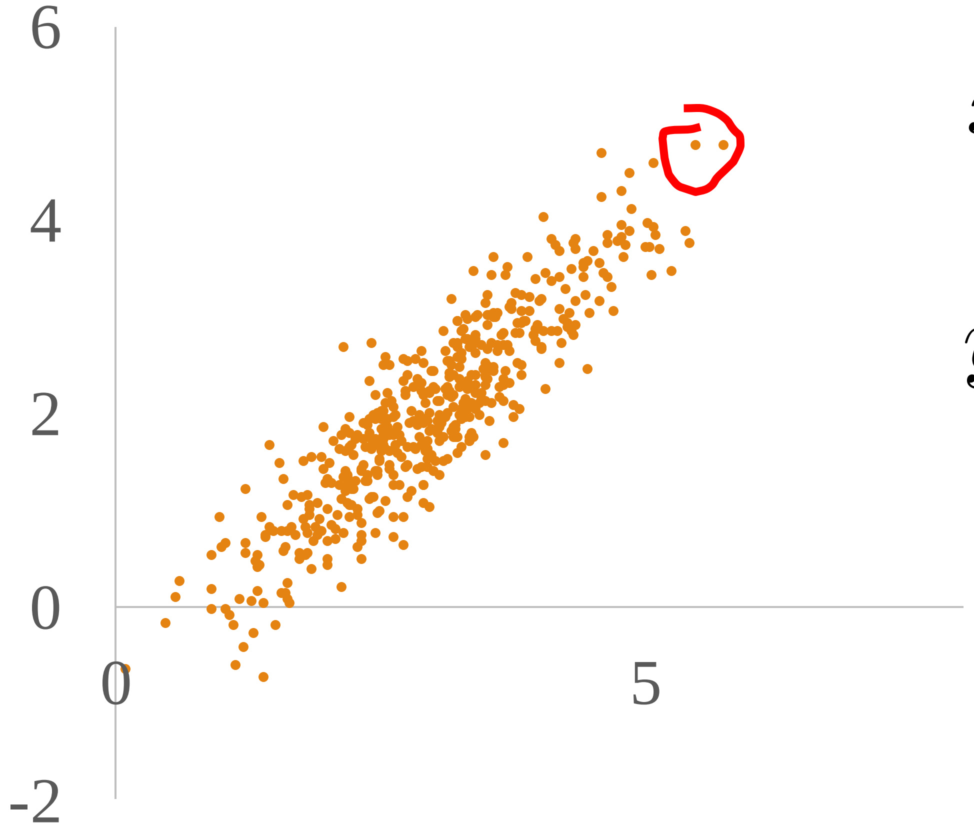
Spot The Difference



Spot The Difference



Vary Together

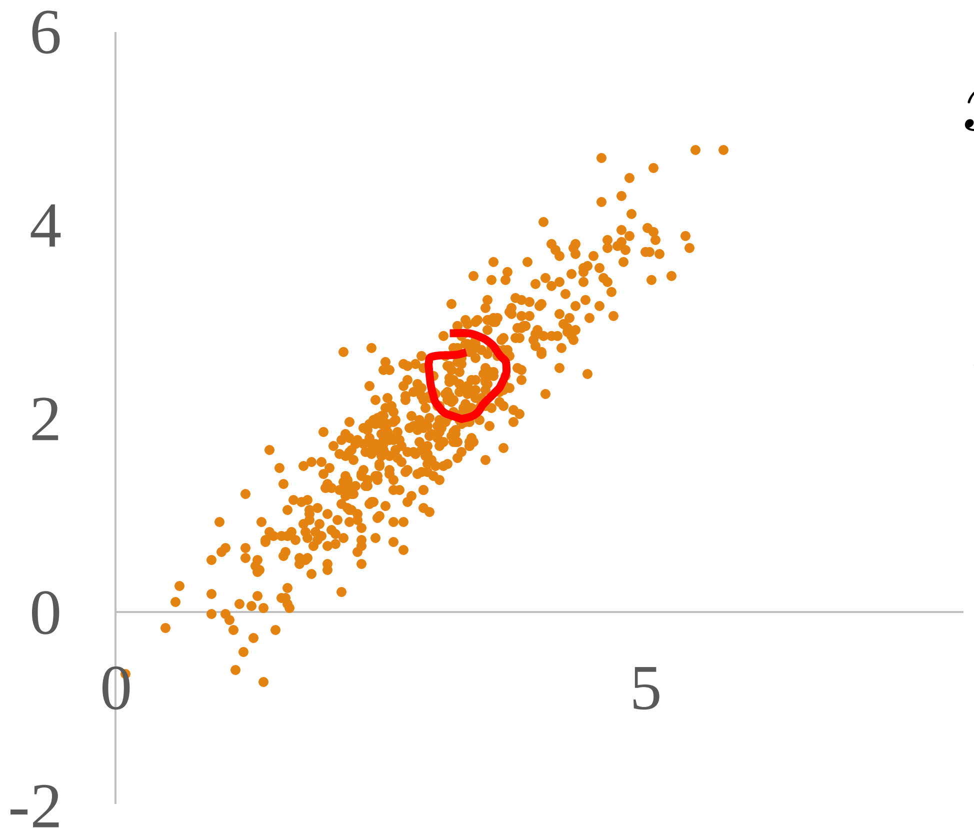


$$x - E[x] = 3$$

$$y - E[y] = 2.6$$

$$(x - E[x])(y - E[y]) = 7.8$$

Vary Together

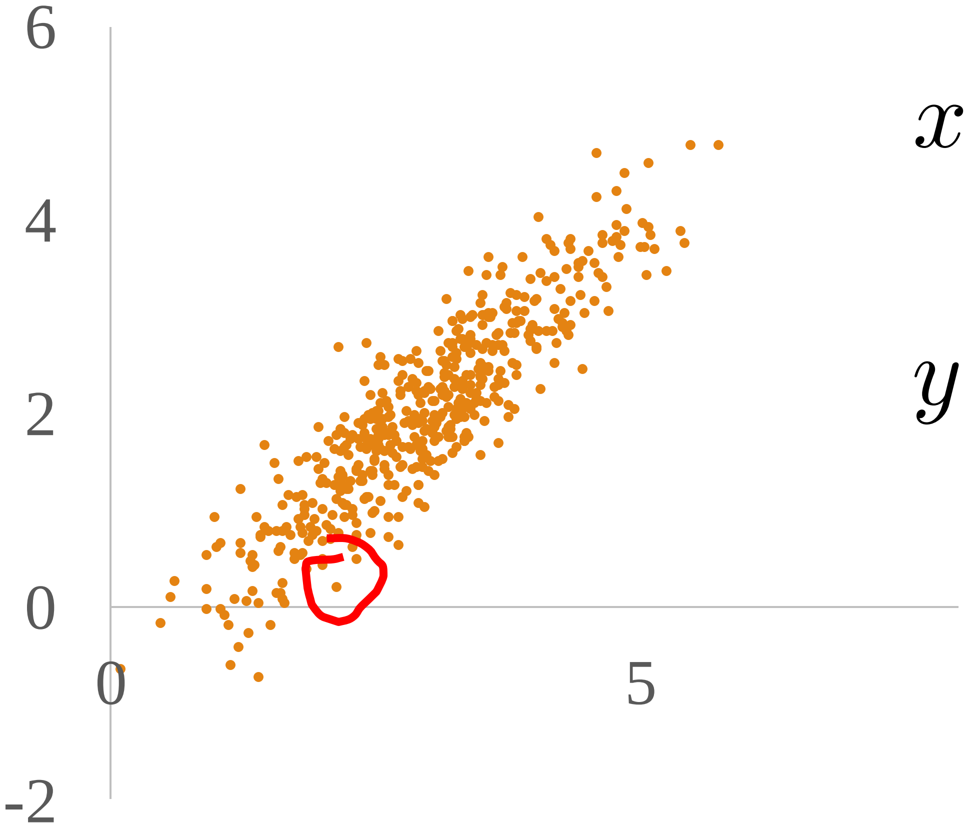


$$x - E[x] \approx 0$$

$$y - E[y] \approx 0$$

$$(x - E[x])(y - E[y]) = 0$$

Vary Together

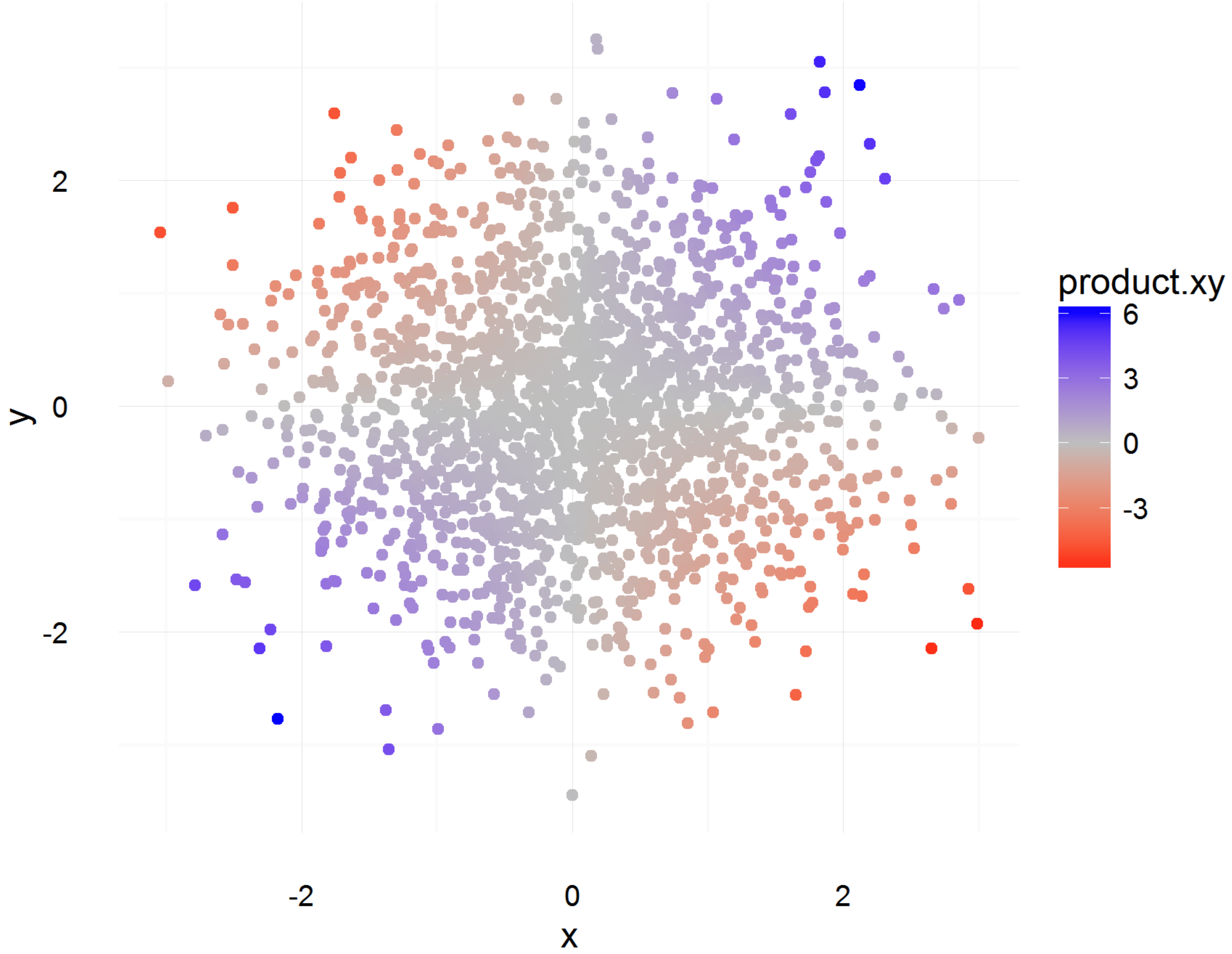


$$x - E[x] = -1.1$$

$$y - E[y] = -2.8$$

$$(x - E[x])(y - E[y]) \approx 3.1$$

Understanding Covariance



The Dance of the Covariance

Say X and Y are arbitrary random variables

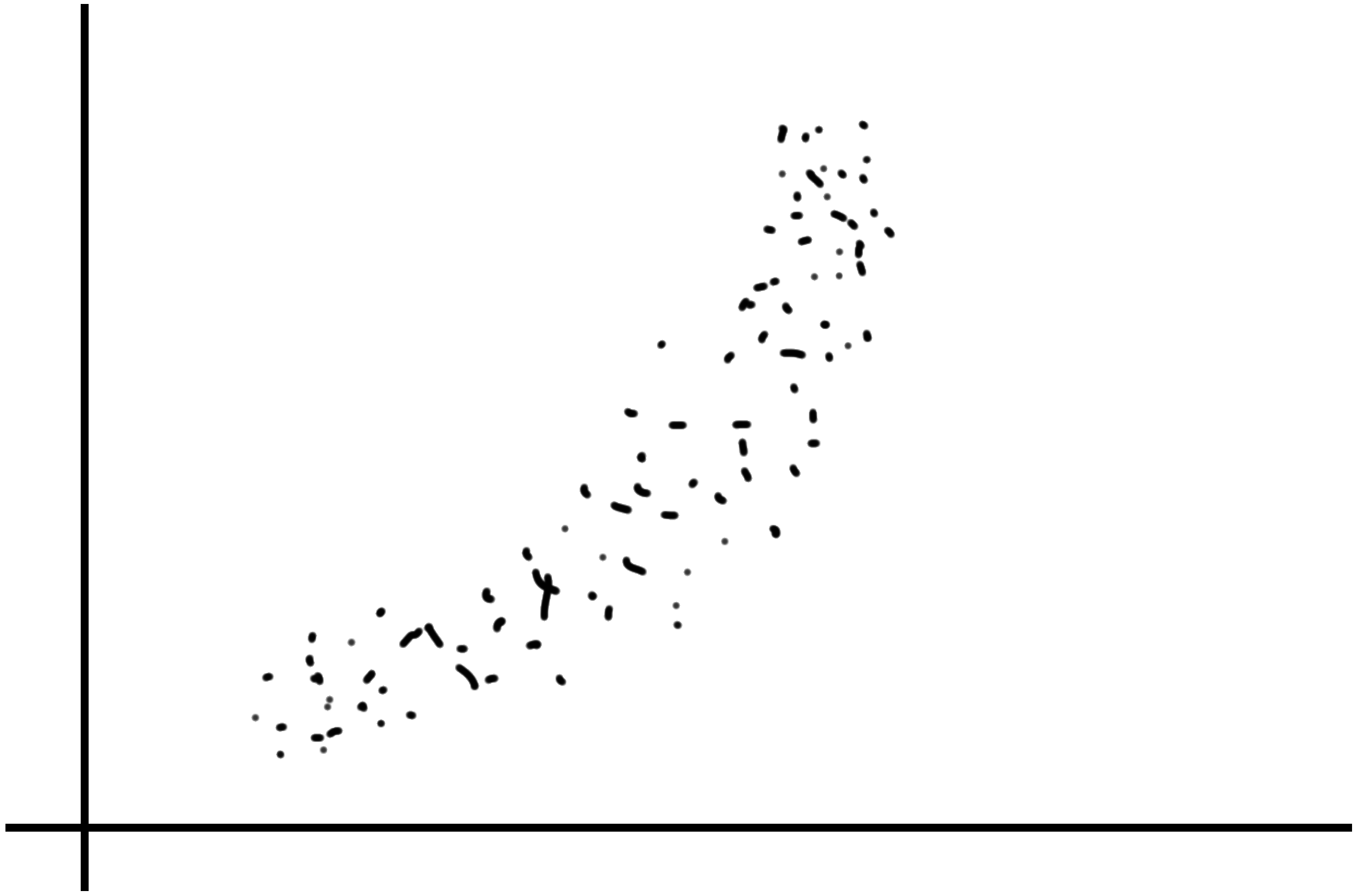
Covariance of X and Y:

$$\text{Cov}(X, Y) = E[(X - E[X])(Y - E[Y])]$$

x	y	(x – E[X])(y – E[Y])p(x,y)
Above mean	Above mean	Positive
Bellow mean	Bellow mean	Positive
Bellow mean	Above mean	Negative
Above mean	Bellow mean	Negative

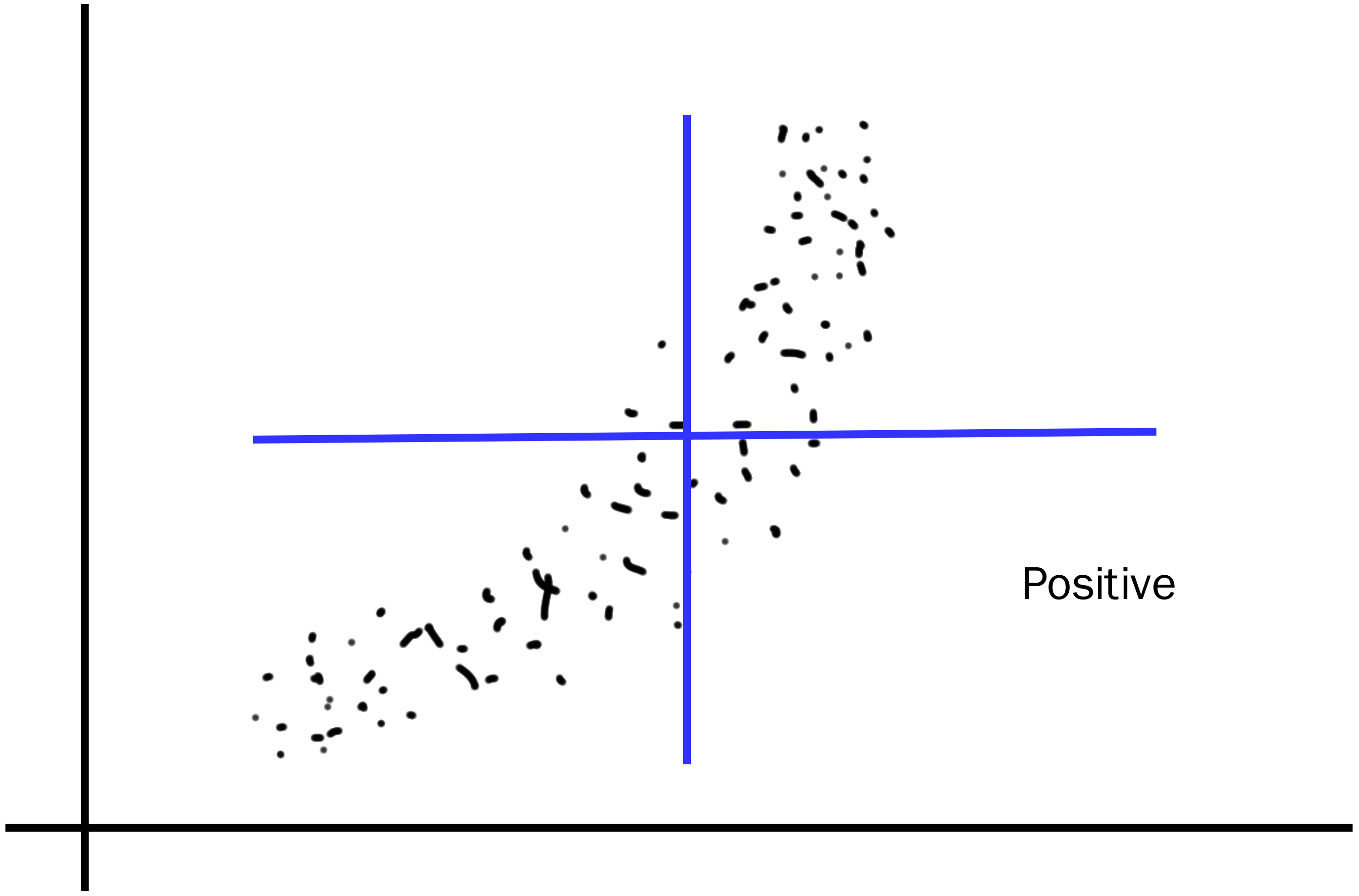
Covariance

Poll: (a) positive, (b) negative, (c) zero



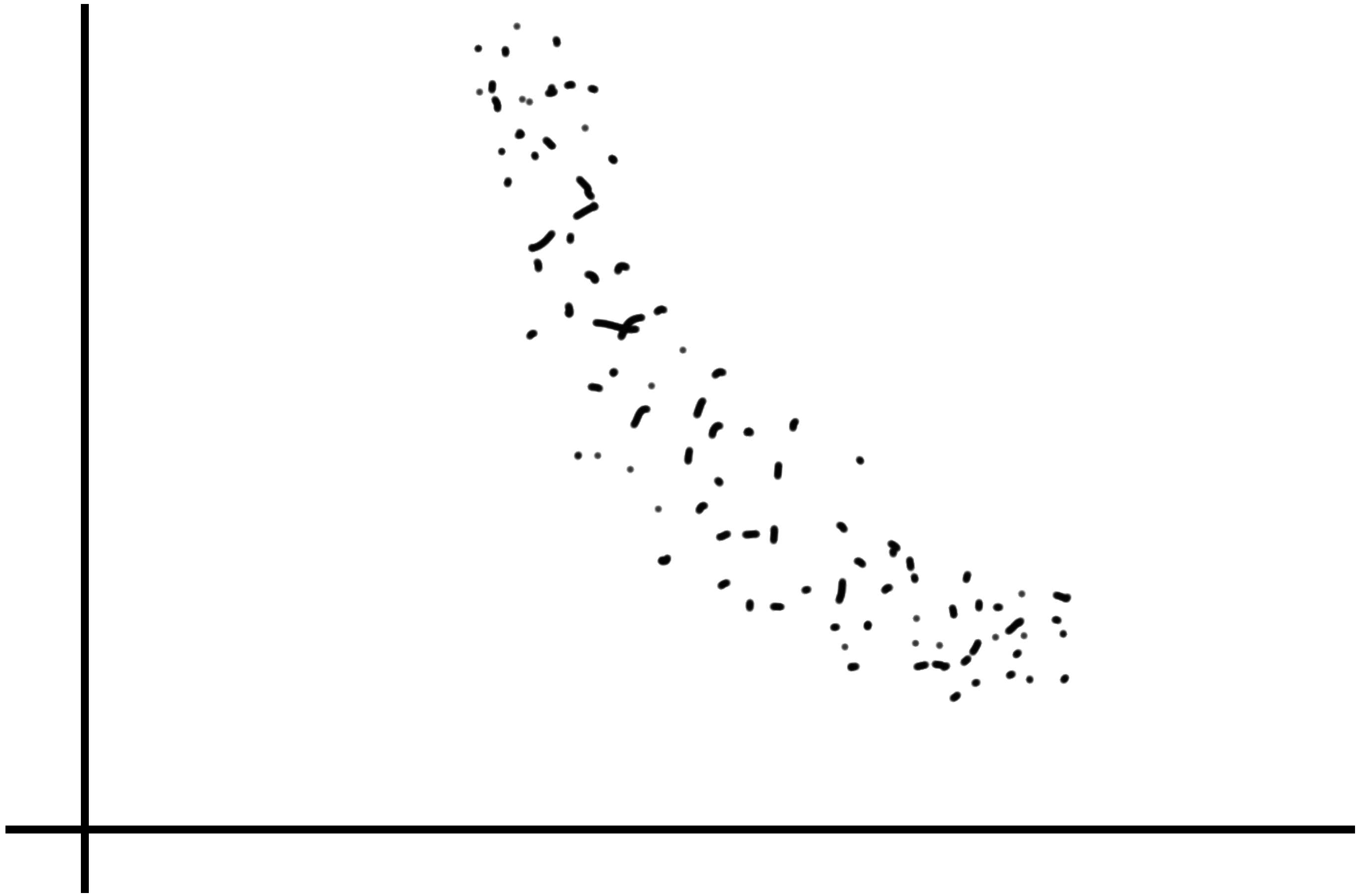
Covariance

Is the Covariance: (a) positive, (b) negative, (c) zero



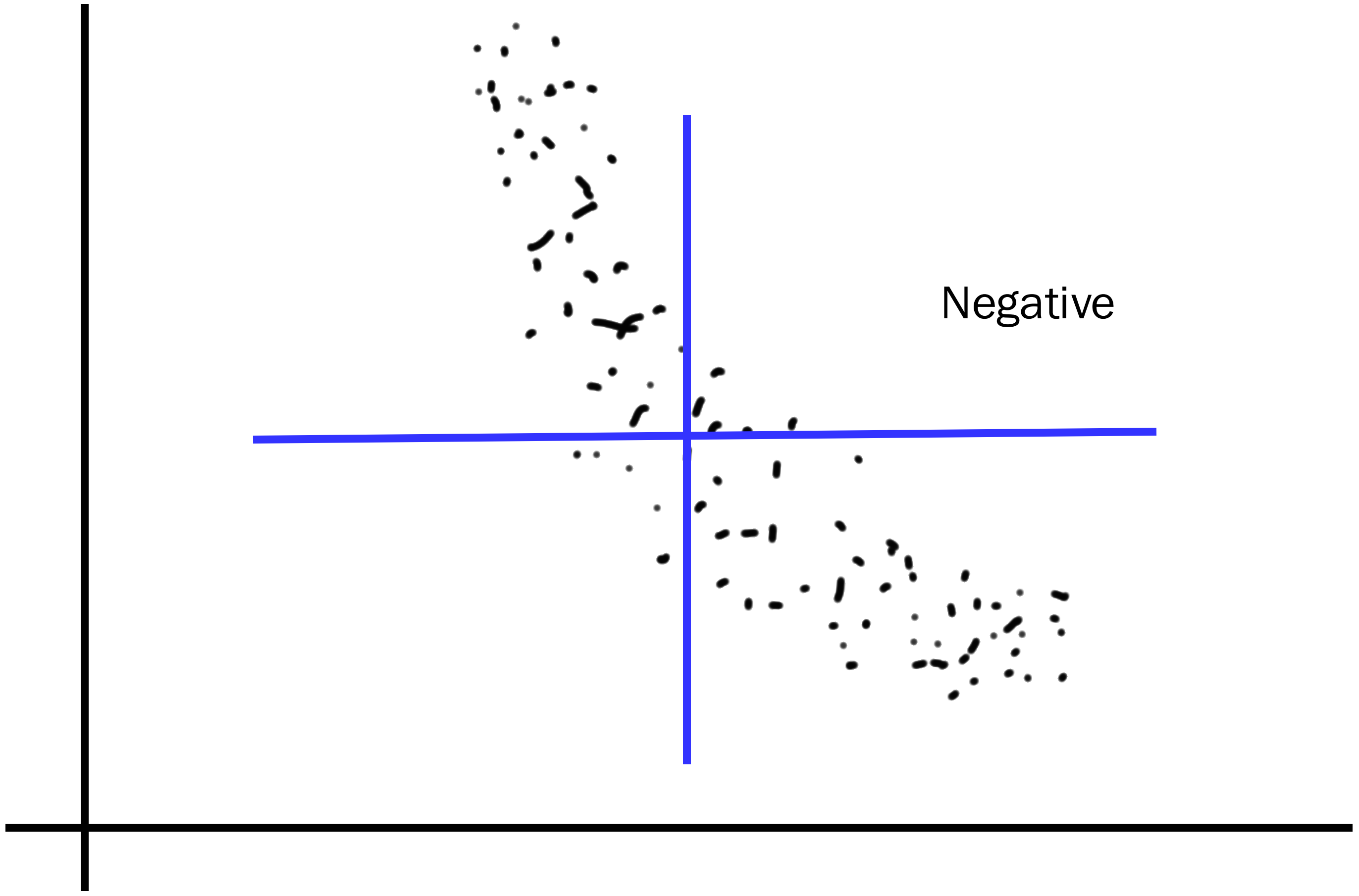
Covariance

Is the Covariance: (a) positive, (b) negative, (c) zero



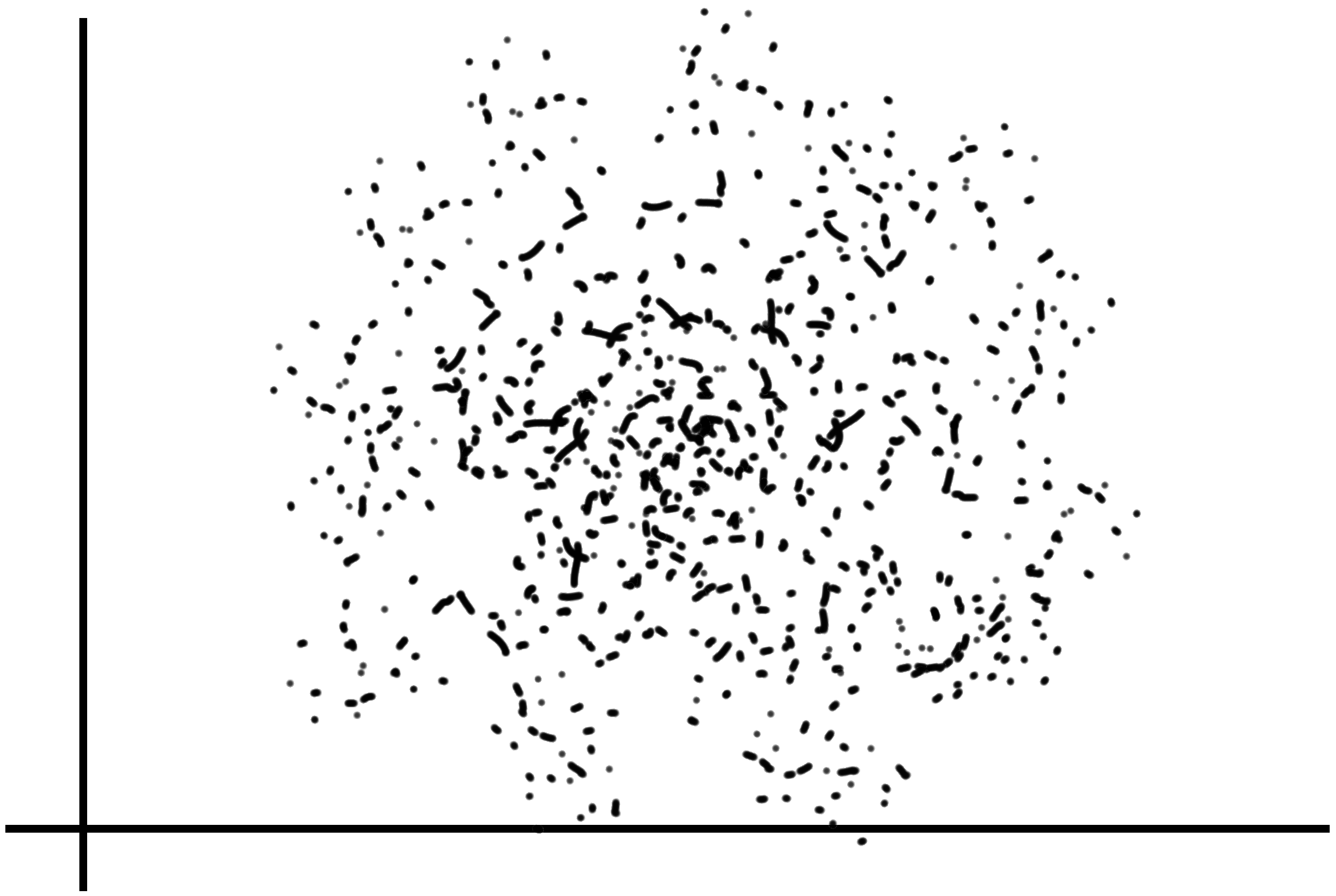
Covariance

Is the Covariance: (a) positive, (b) negative, (c) zero



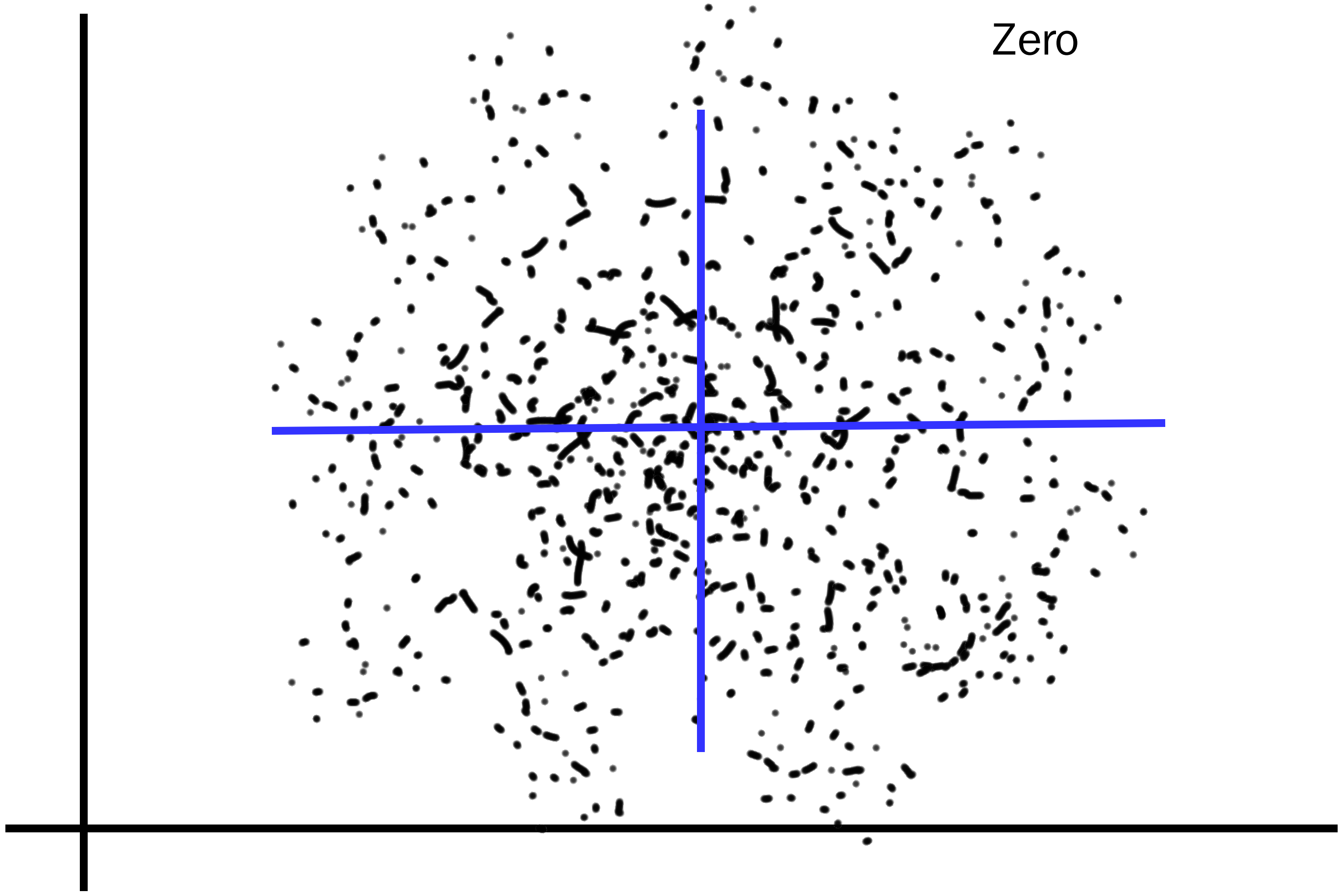
Covariance

Is the Covariance: (a) positive, (b) negative, (c) zero



Covariance

Is the Covariance: (a) positive, (b) negative, (c) zero



The Dance of the Covariance

Say X and Y are arbitrary random variables

Covariance of X and Y:

$$\text{Cov}(X,Y) = E[(X - E[X])(Y - E[Y])]$$

Equivalently:

$$\text{Cov}(X,Y) = E[XY - E[X]Y - XE[Y] + E[Y]E[X]]$$

$$= E[XY] - E[X]E[Y] - E[X]E[Y] + E[X]E[Y]$$

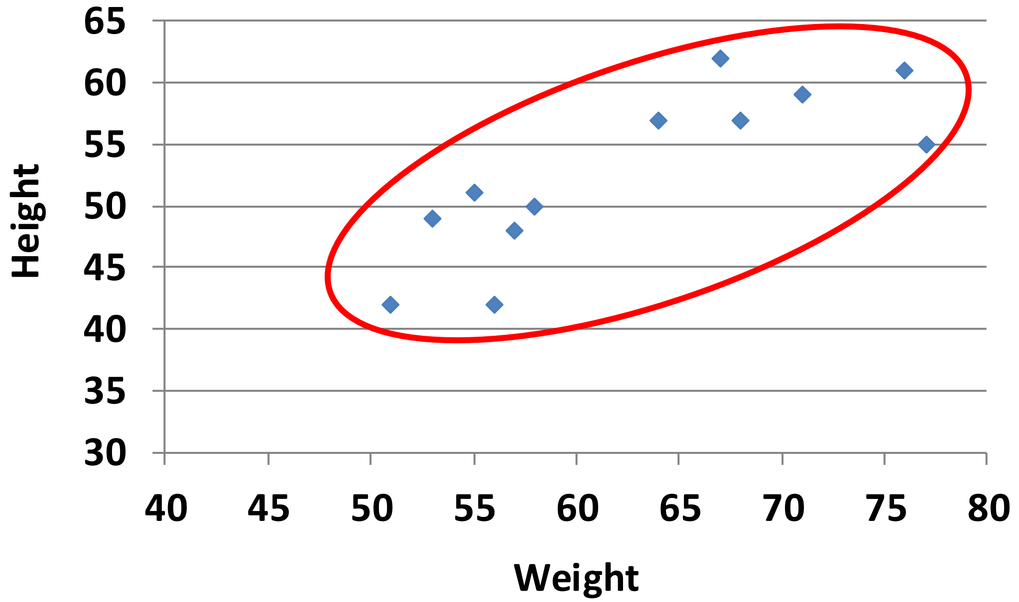
$$= E[XY] - E[X]E[Y]$$

- X and Y independent → Cov(X,Y) = 0
- But Cov(X,Y) = 0 does **not** imply X and Y independent!

Covariance and Data

Consider the following data:

Weight	Height	Weight * Height
64	57	3648
71	59	4189
53	49	2597
67	62	4154
55	51	2805
58	50	2900
77	55	4235
57	48	2736
56	42	2352
51	42	2142
76	61	4636
68	57	3876
E[W] = 62.75	E[H] = 52.75	E[W*H] = 3355.83



Cov(W, H) = E[W*H] – E[W]E[H]
= 3355.83 – (62.75)(52.75)
= 45.77

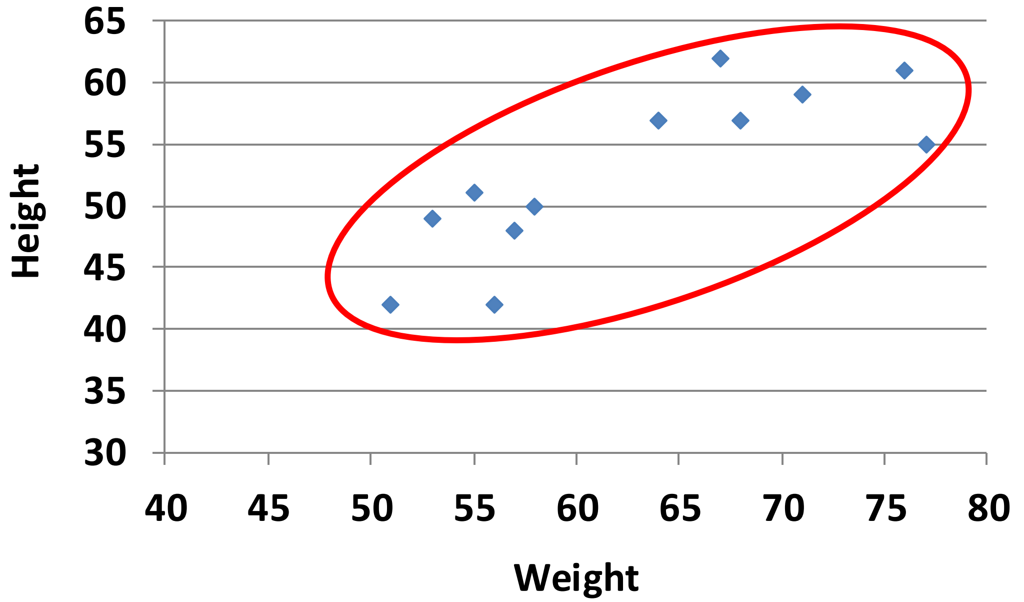
Correlation

What is Wrong With This?

Consider the following data:

Weight	Height	Weight * Height
64	57	3648
71	59	4189
53	49	2597
67	62	4154
55	51	2805
58	50	2900
77	55	4235
57	48	2736
56	42	2352
51	42	2142
76	61	4636
68	57	3876

E[W] = 62.75 **E[H] = 52.75** **E[W*H] = 3355.83**

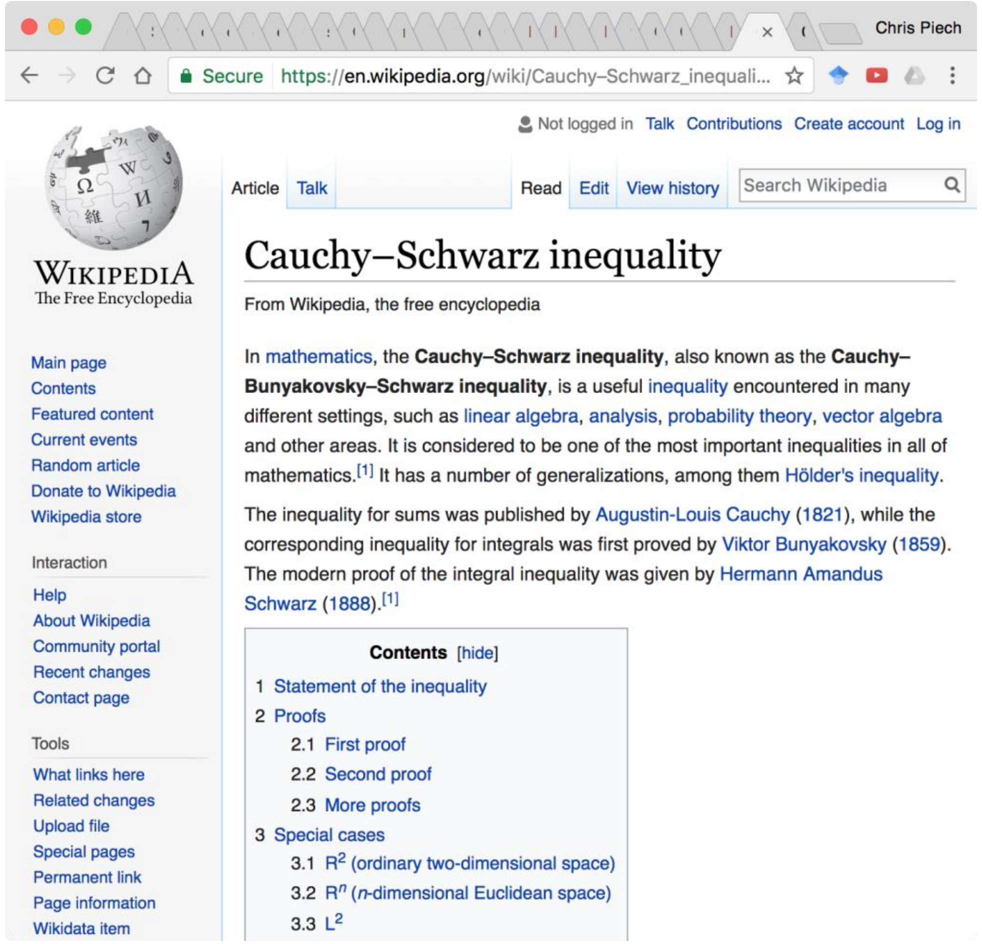


Cov(W, H) = E[W*H] – E[W]E[H]

= 3355.83 – (62.75)(52.75)

= 45.77

Cauchy Schwarz, a great way to normalize!



$$-\text{Std}(X)\text{Std}(Y) \leq \text{Cov}(X, Y) \leq \text{Std}(X)\text{Std}(Y)$$

Viva La Correlatión

Say X and Y are arbitrary random variables

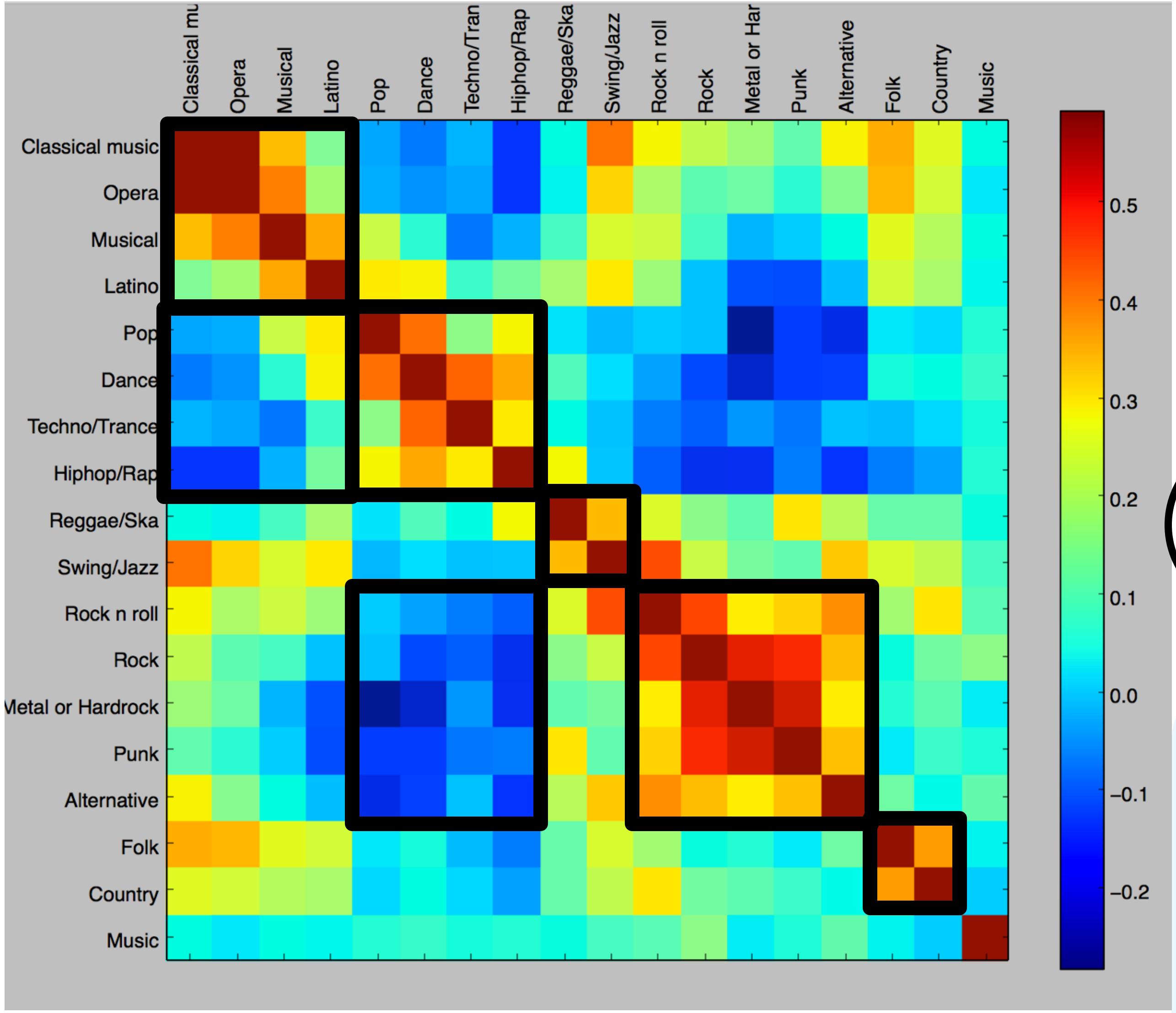
- Correlation of X and Y, denoted $\rho(X, Y)$:

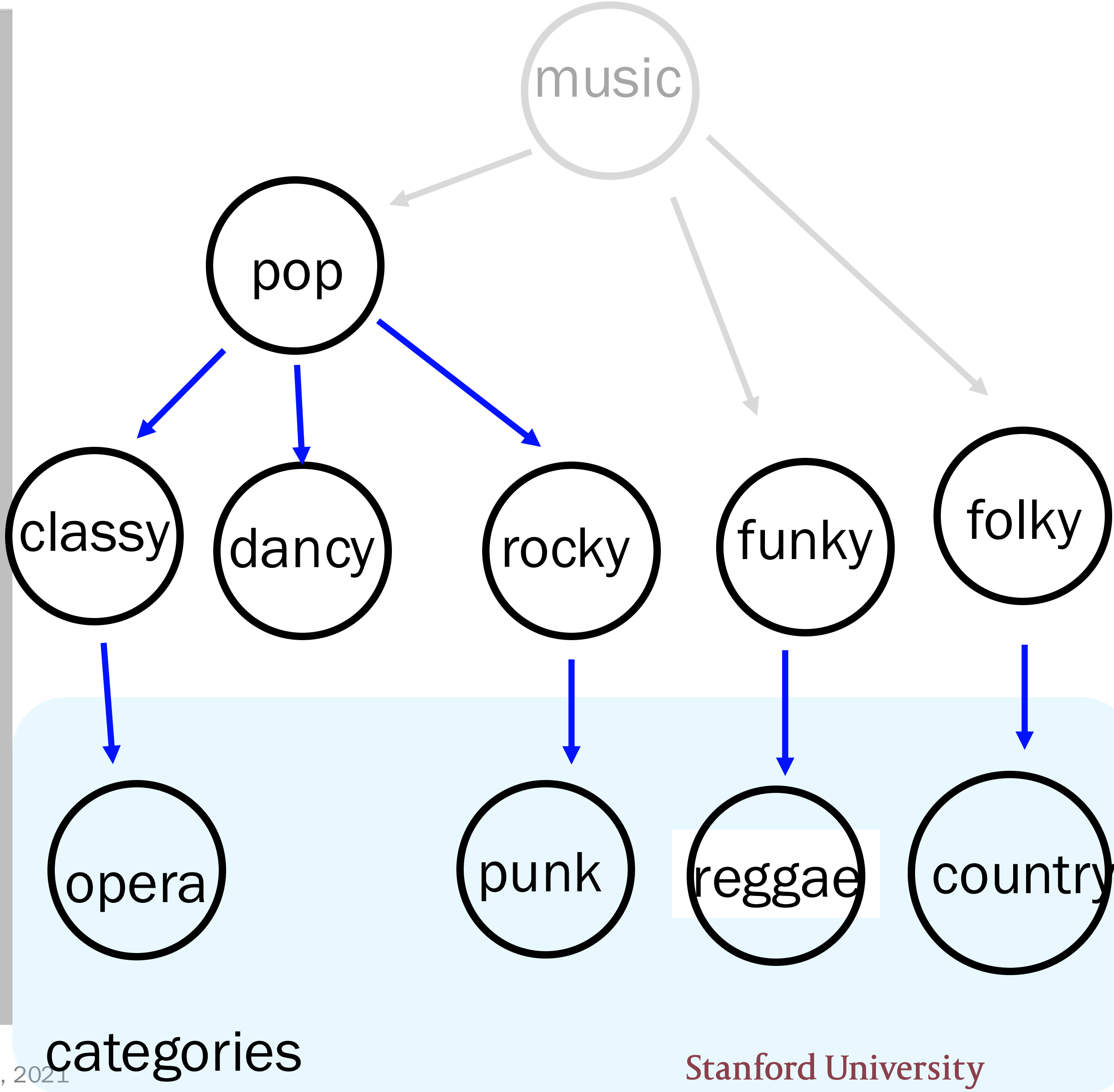
$$\rho(X,Y) = \frac{\text{Cov}(X,Y)}{\sqrt{\text{Var}(X)\text{Var}(Y)}}$$

- Note: $-1 \leq \rho(X, Y) \leq 1$

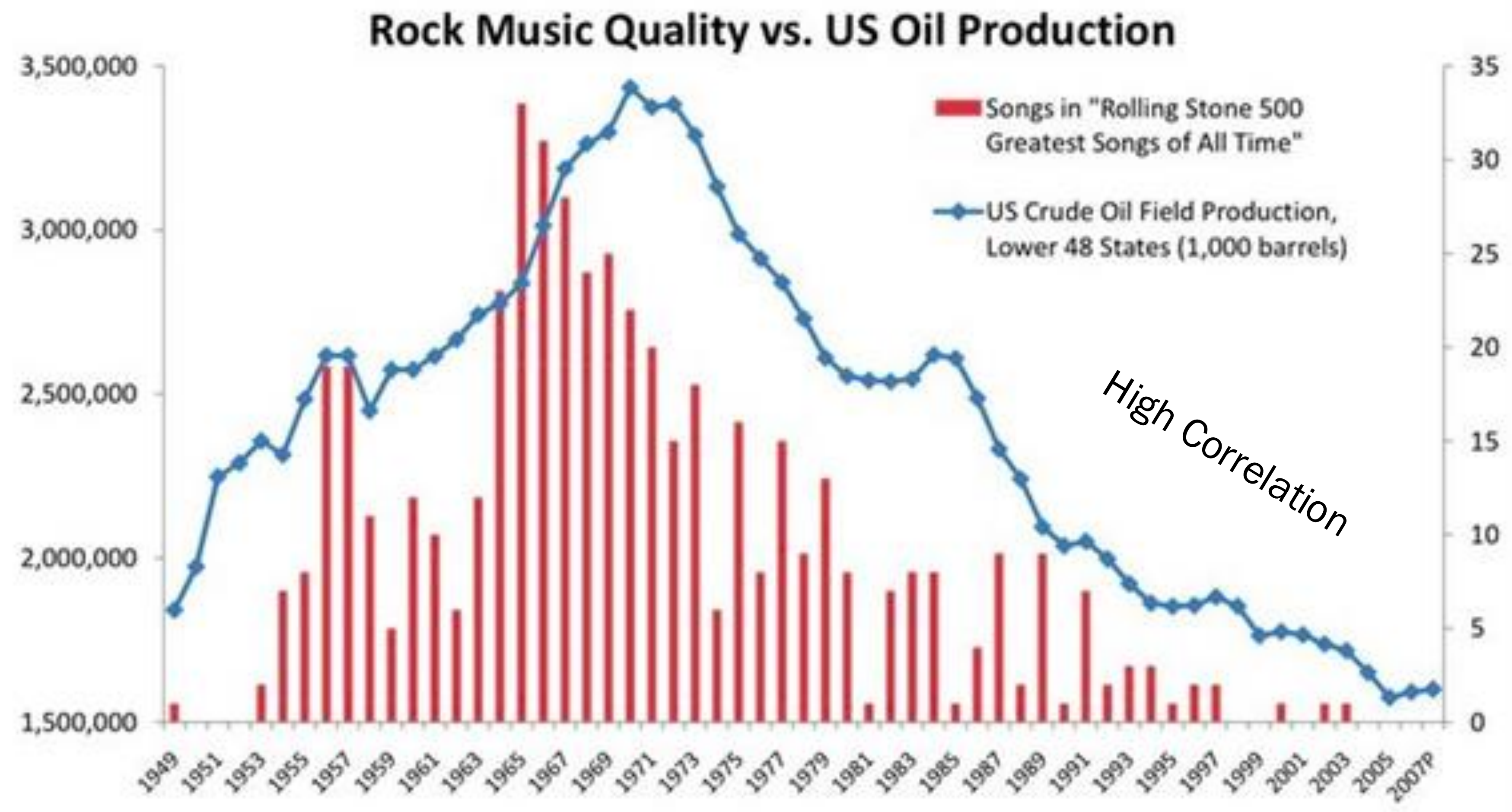
- $\rho(X, Y) = 1$ $\Rightarrow$ perfectly correlated
- $\rho(X, Y) = -1$ $\Rightarrow$ perfectly negatively correlated
- $\rho(X, Y) = 0$ $\Rightarrow$ absence of linear relationship
  - But, X and Y can still be related in some other way!
- If $\rho(X, Y) = 0$, we say X and Y are "uncorrelated"

Recall: It is a useful starting point





Rock Music Vs Oil?

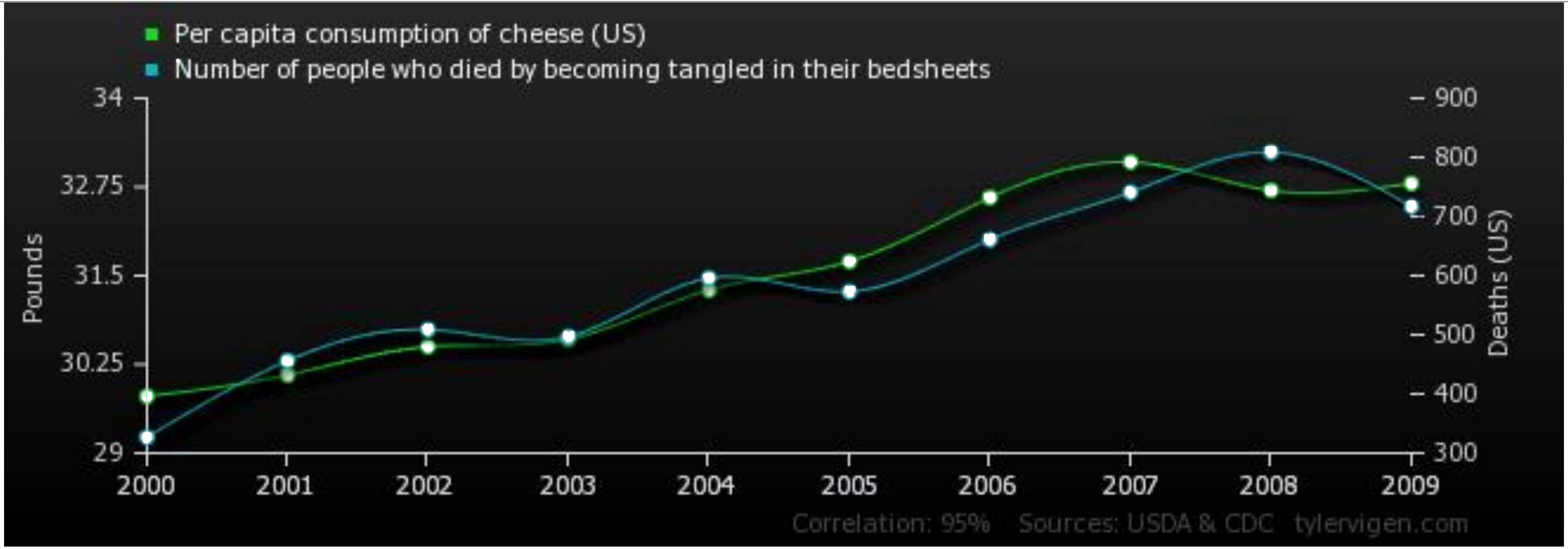


Hubbert Peak Theory

http://www.aei.org/publication/blog/

Stanford University

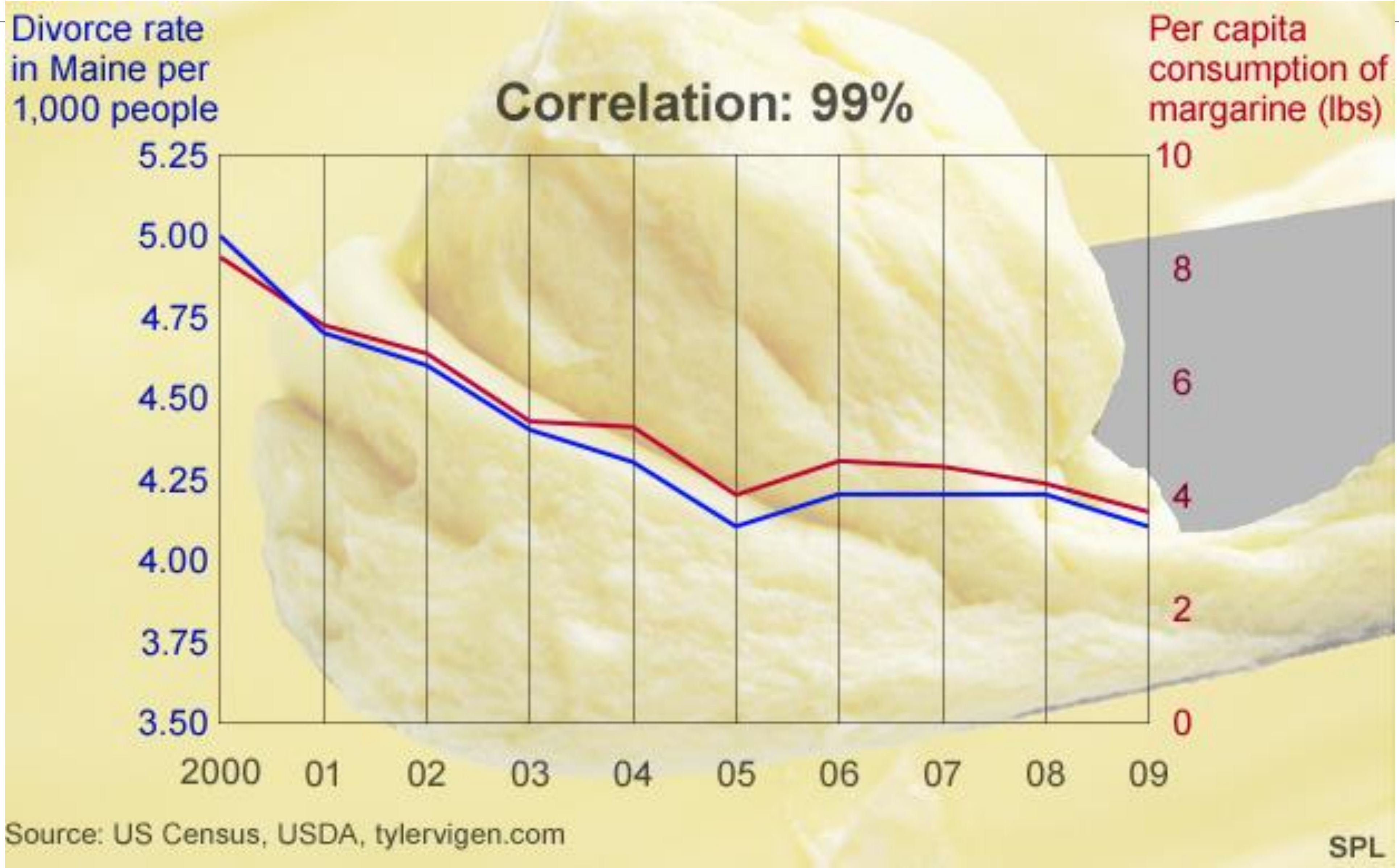
Tell your friends!



	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009
Per capita consumption of cheese (US) *Pounds (USDA)*	29.8	30.1	30.5	30.6	31.3	31.7	32.6	33.1	32.7	32.8
Number of people who died by becoming tangled in their bedsheets *Deaths (US) (CDC)*	327	456	509	497	596	573	661	741	809	717

Correlation: 0.947091

Divorce Vs Butter?



http://www.bbc.com/news/magazine-27537142

Three Guiding Questions

1. How do people actually define large models?

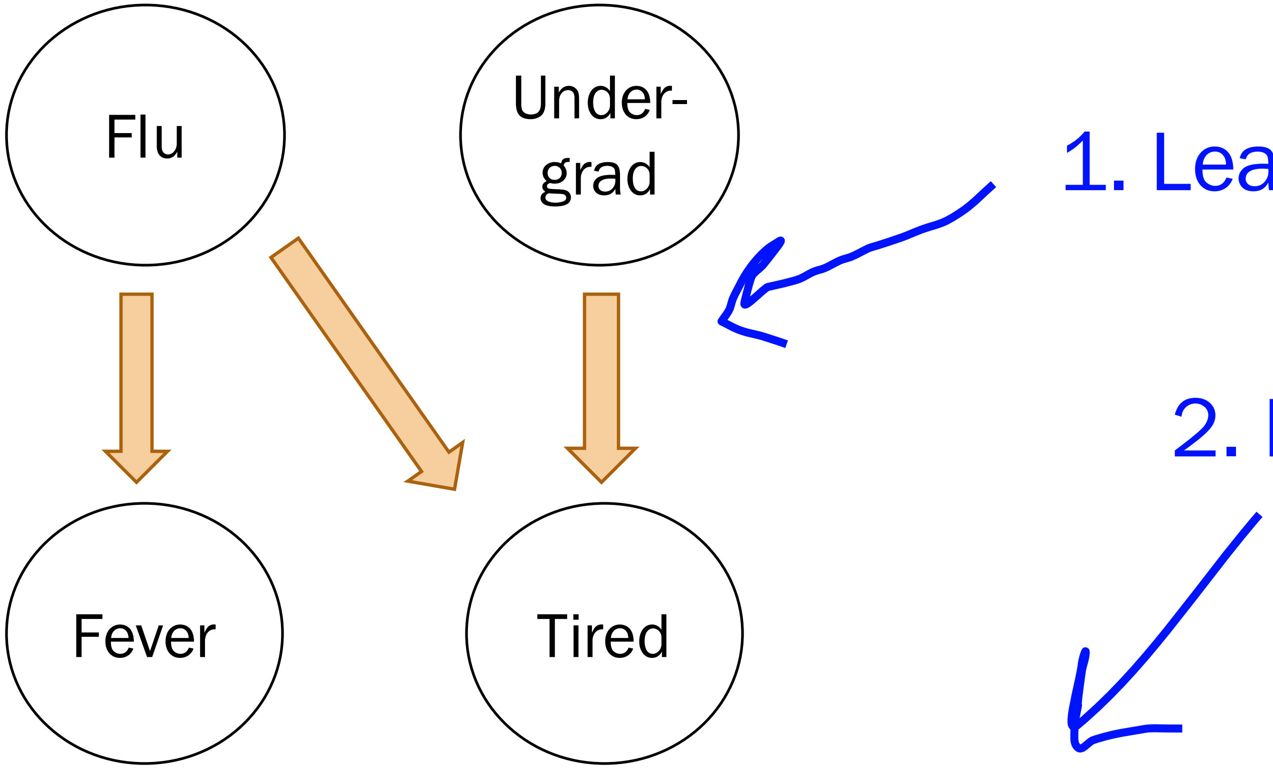
2. How can we do inference in large models?

3. What data can inform the design process?

What haven’t we talked about?

Machine Learning (last section of CS109)

$P(F_{lu} = 1) = 0.1$ $\quad$ $P(U = 1) = 0.8$



1. Learn this from data

2. Learn this from data

$P(F_{ev} = 1|F_{lu} = 1) = 0.9$
$P(F_{ev} = 1|F_{lu} = 0) = 0.05$

$P(T = 1|F_{lu} = 0, U = 0) = 0.1$
$P(T = 1|F_{lu} = 0, U = 1) = 0.8$
$P(T = 1|F_{lu} = 1, U = 0) = 0.9$
$P(T = 1|F_{lu} = 1, U = 1) = 1.0$