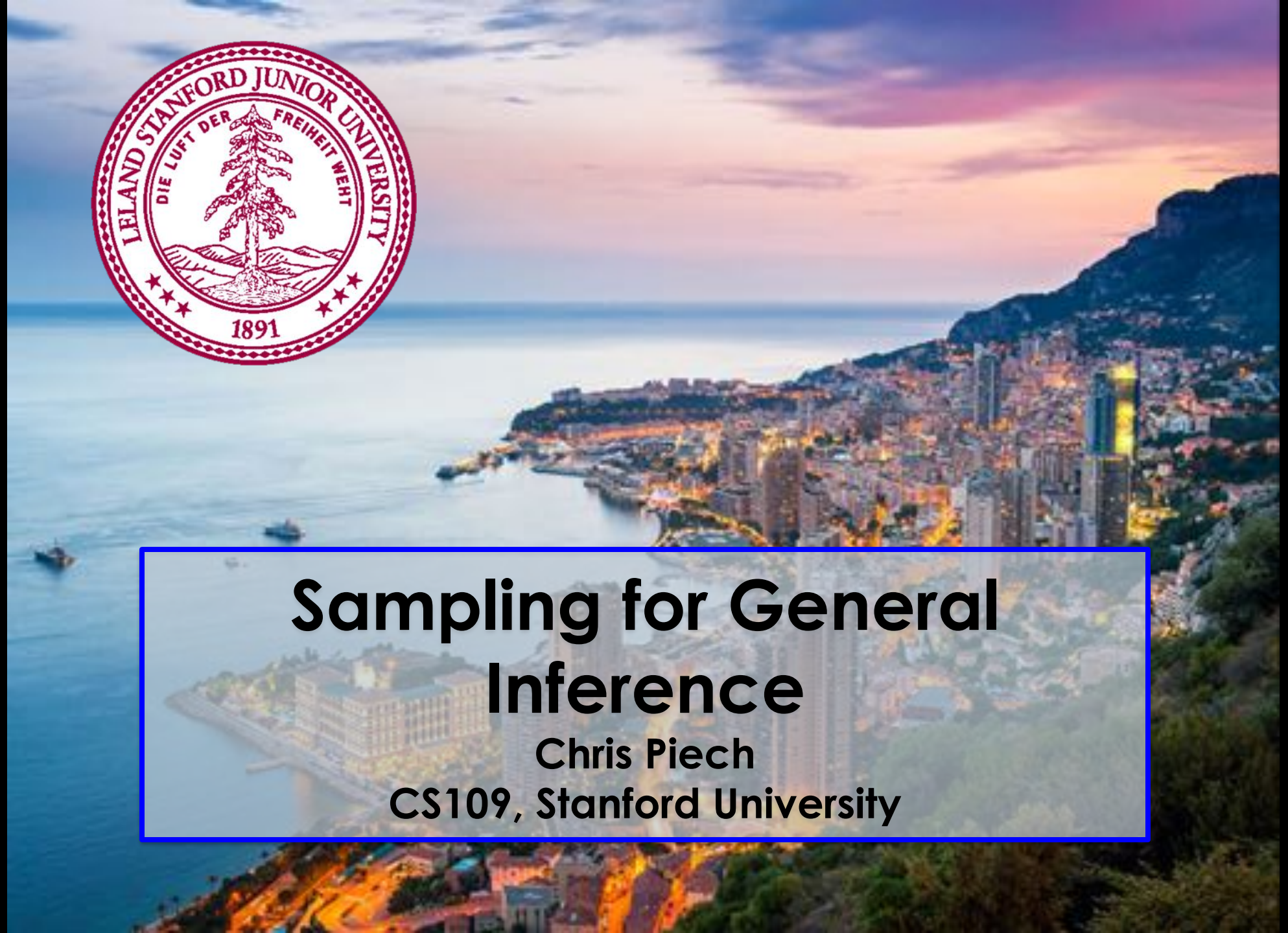


Sampling for General Inference

Chris Piech
CS109, Stanford University

CS109 Contest

Null Hypothesis Test

Nepal Happiness	Bhutan Happiness
4.44	2.15
3.36	3.01
5.87	2.02
2.31	1.43
...	...
3.70	1.83
$\mu_1 = 3.1$	$\mu_2 = 2.4$

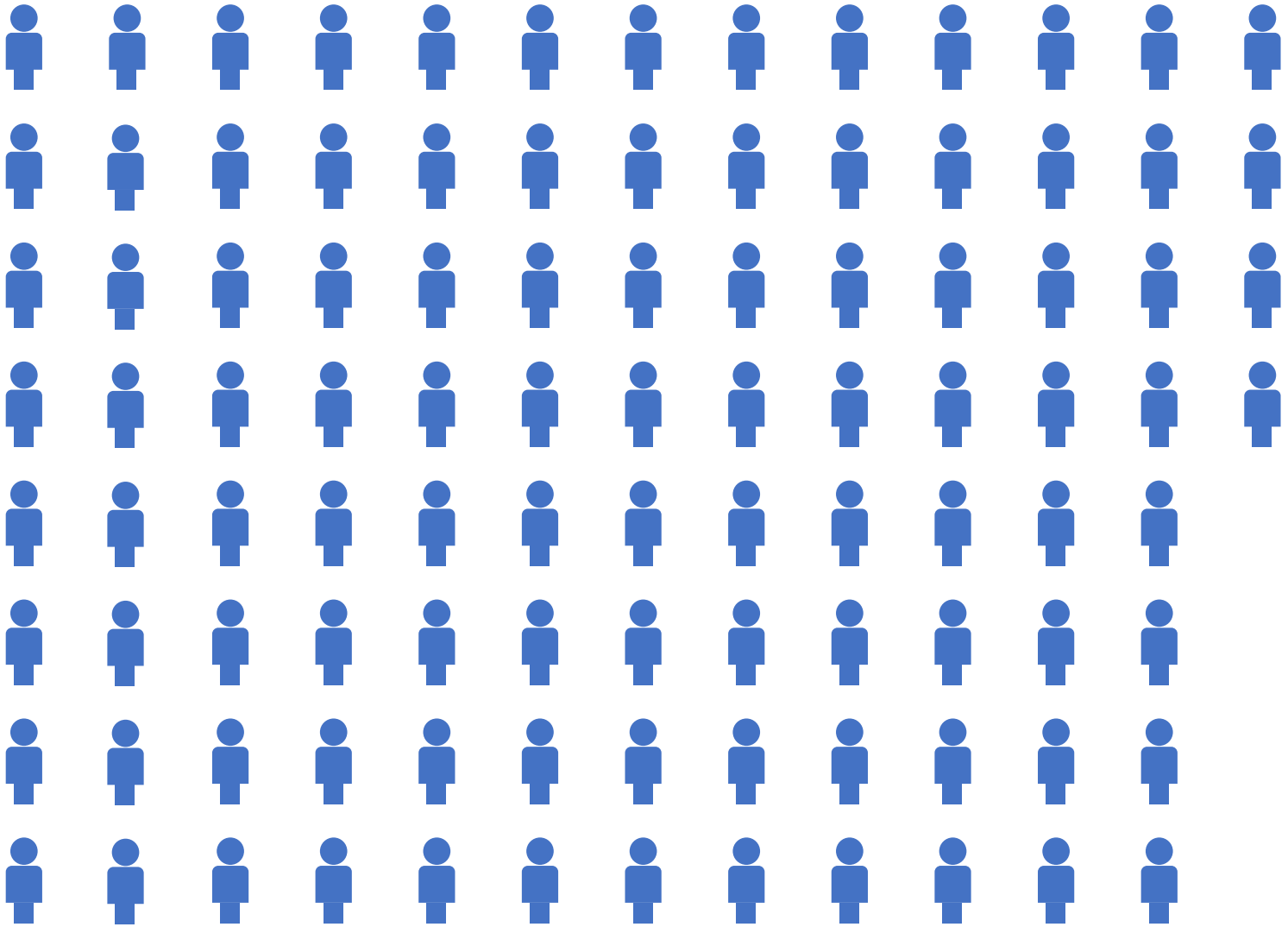
Claim: The difference in happiness between Nepal and Bhutan is 0.7 happiness points (p = 0.008).

Something brand **new**…

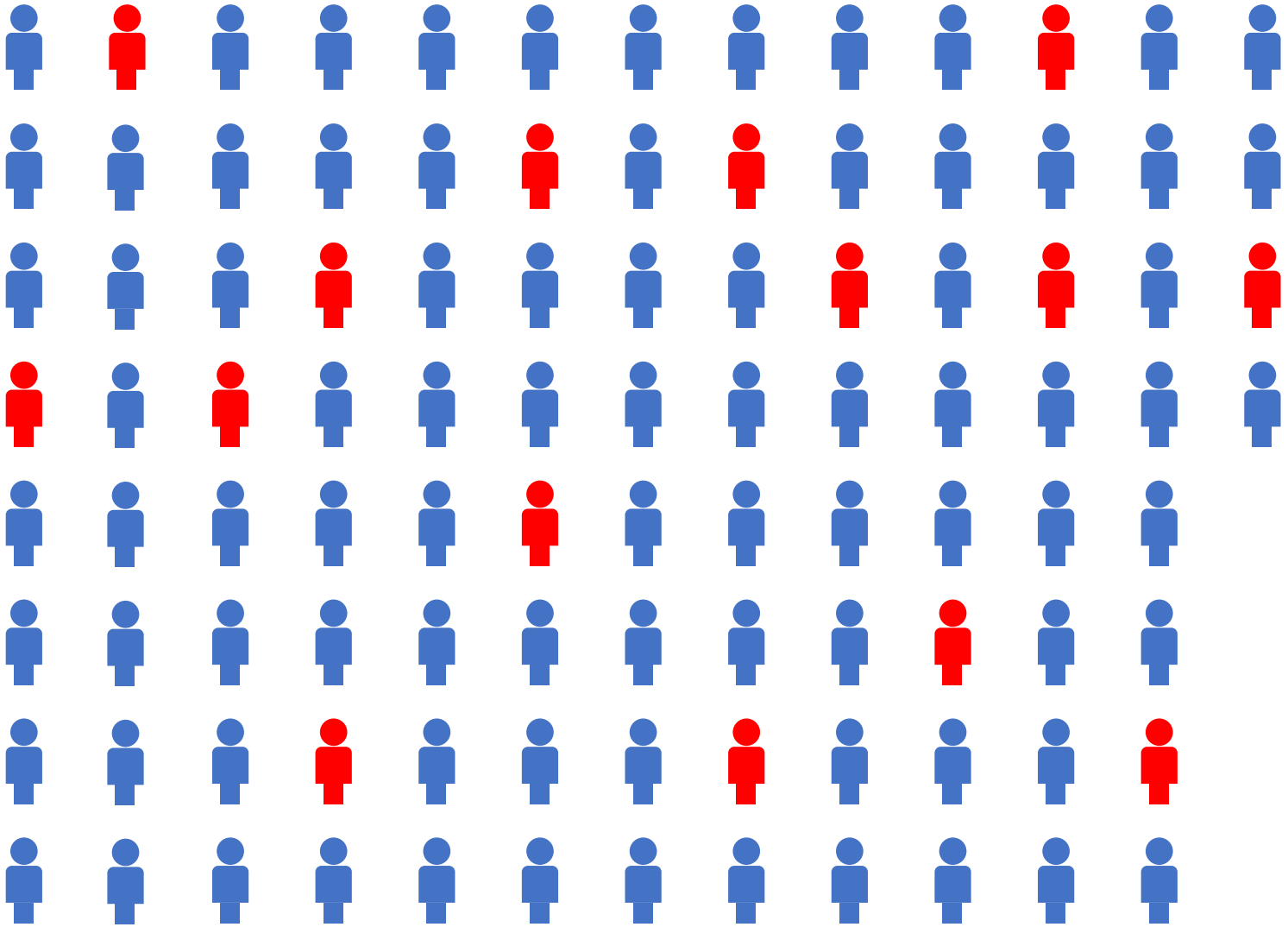
General “Inference”

Review

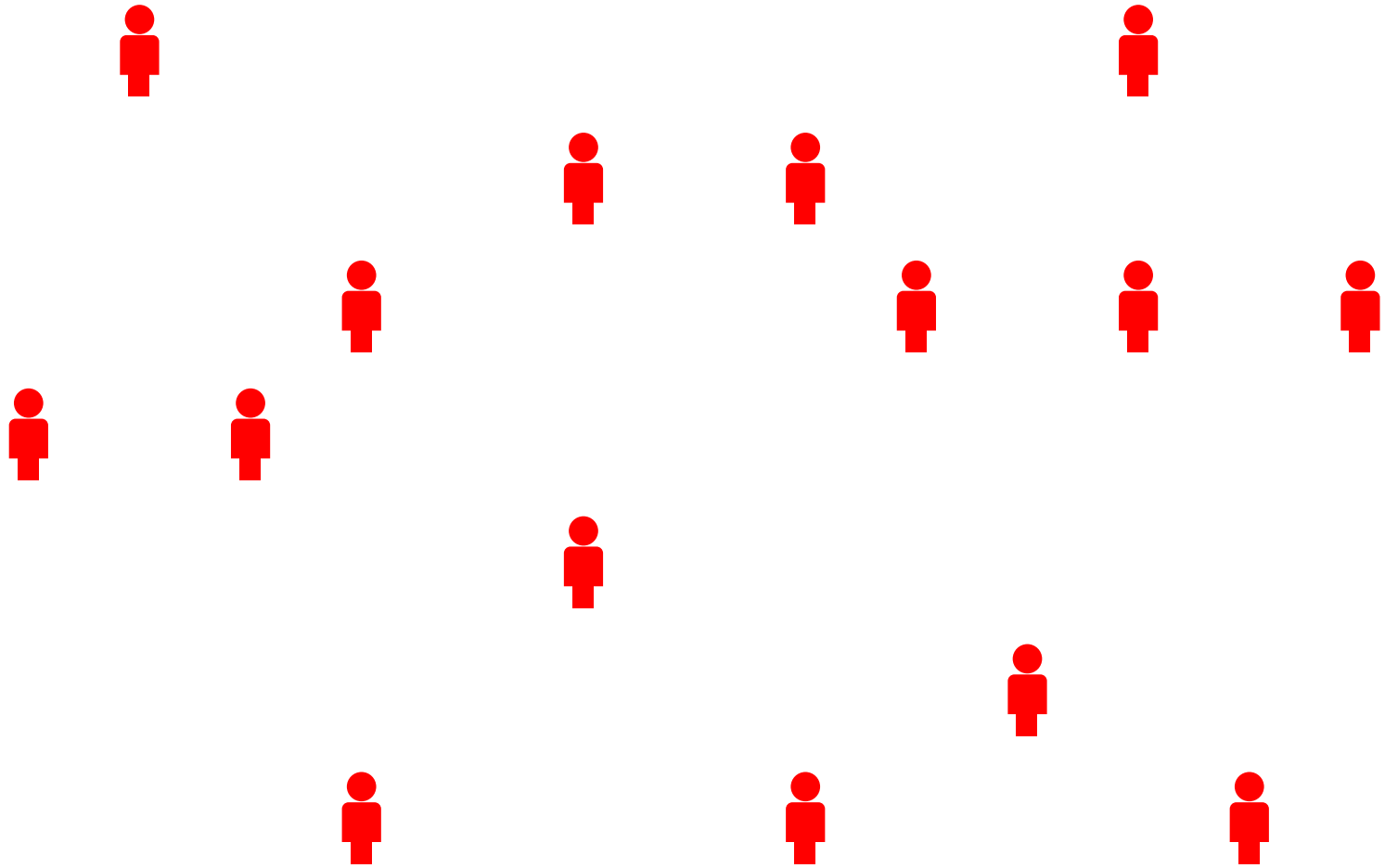
Population



Sample

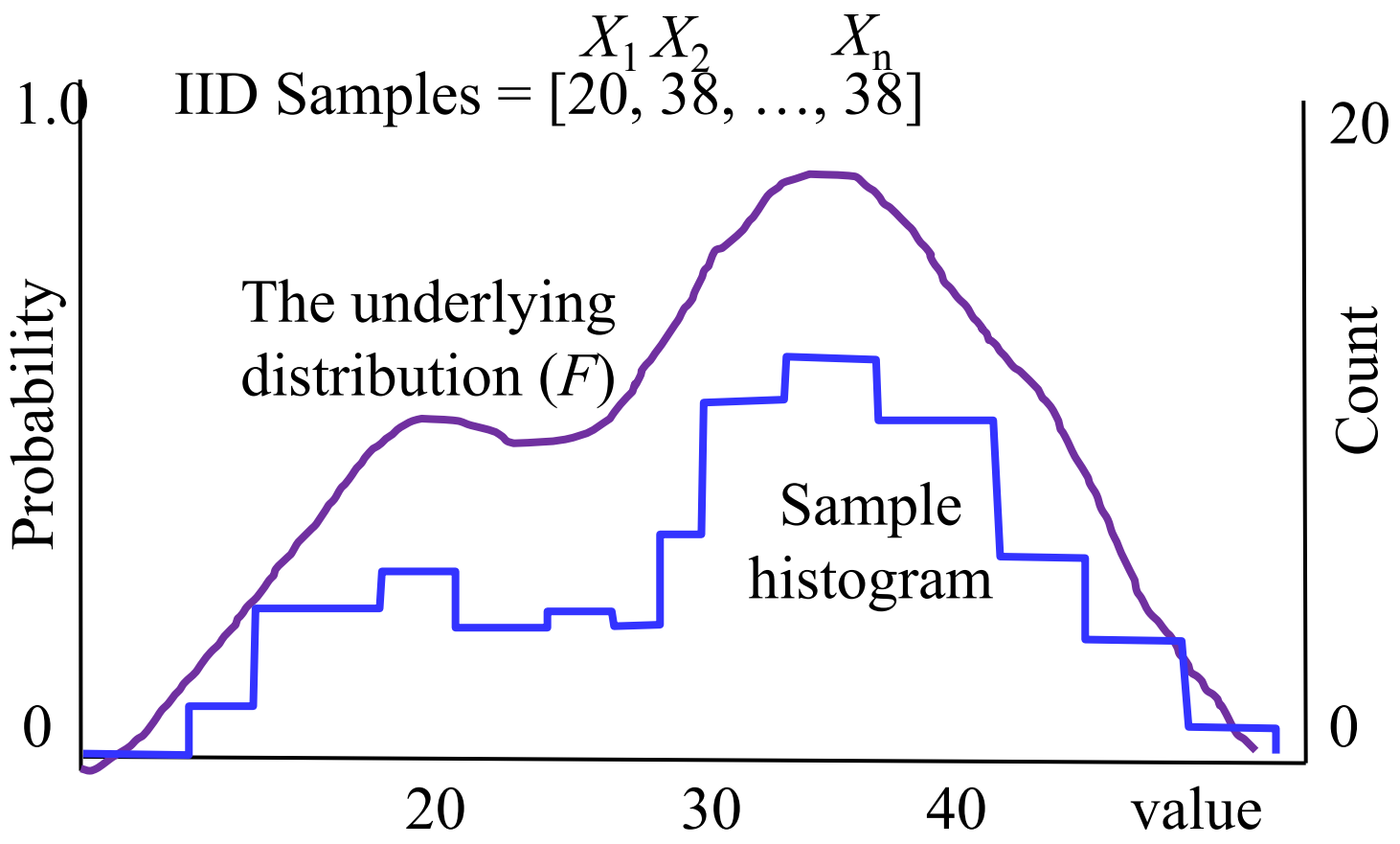


Sample

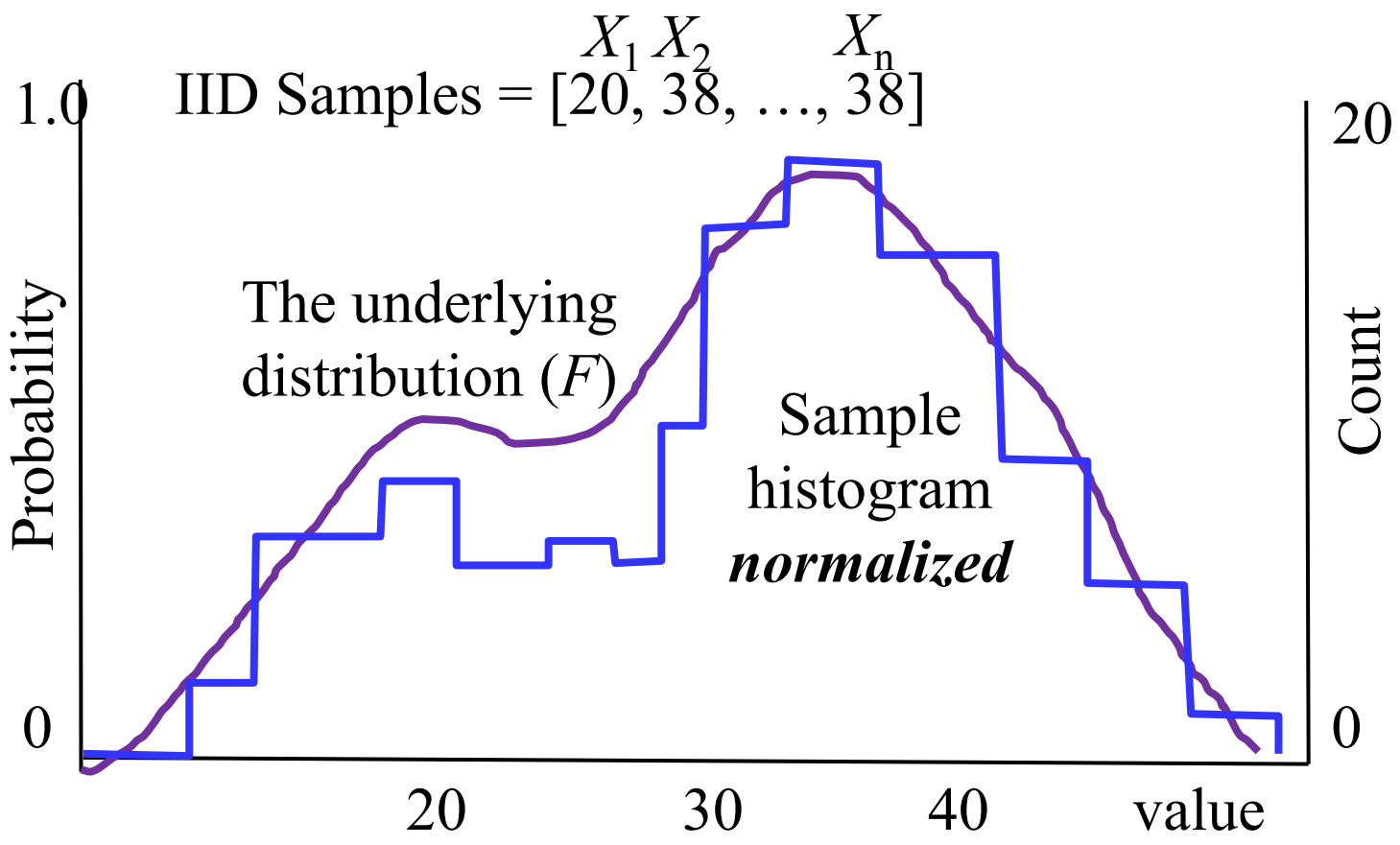


Collect one (or more) numbers from each person

Samples


X_1 X_2 X_n
IID Samples = [20, 38, …, 38]
1.0
0
Probability
20
Count
0
The underlying distribution (F)
Sample histogram
20
30
40
value

Samples


X_1 X_2 X_n
IID Samples = [20, 38, …, 38]
1.0
0
Probability
20
Count
0
The underlying distribution (F)
Sample histogram
normalized
20
30
40
value

Sample Statistics

Sample Mean

$$\bar{X} = \sum_{i=1}^{n} \frac{X_i}{n}$$

Sample Variance

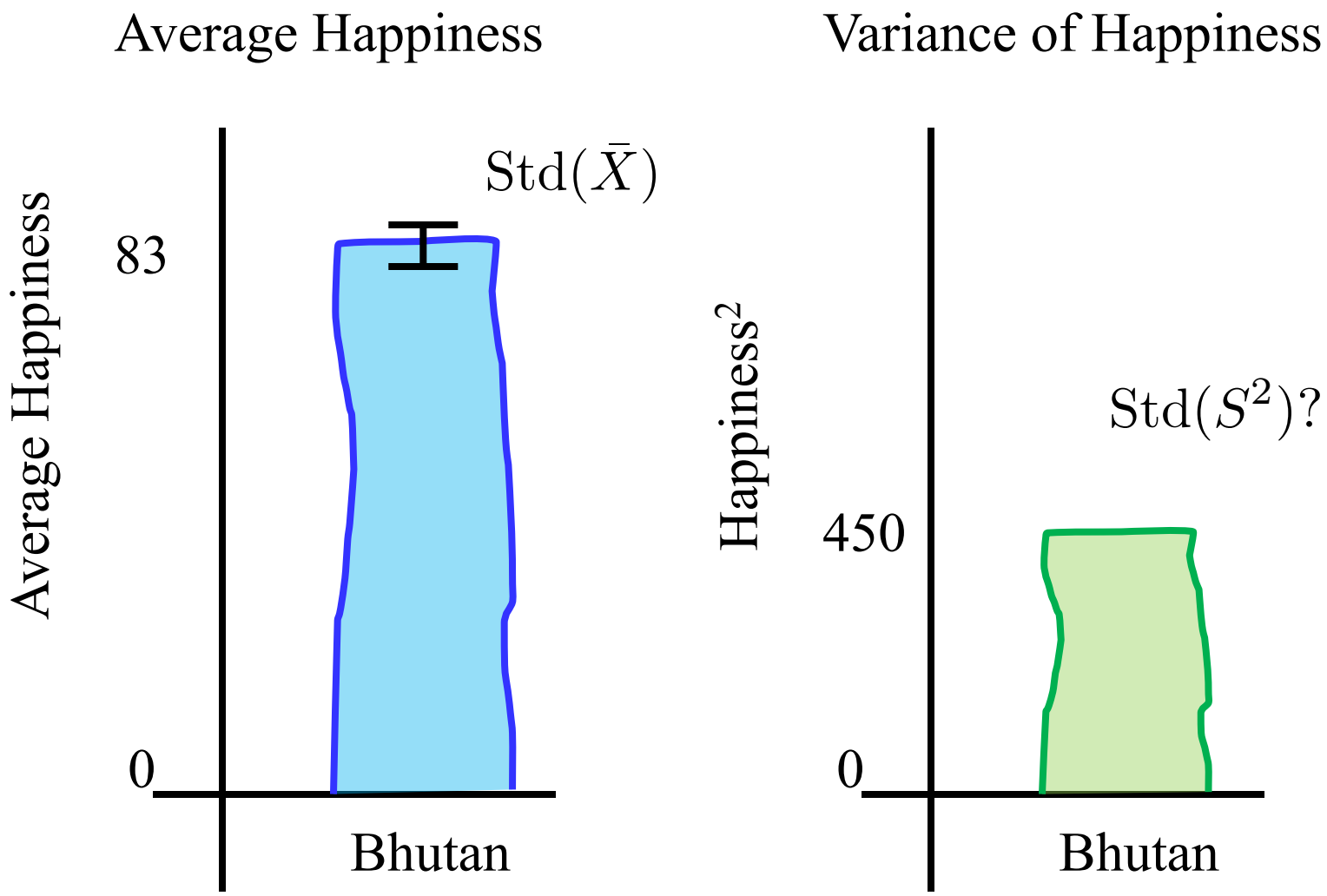
$$S^2 = \sum_{i=1}^{n} \frac{(X_i - \bar{X})^2}{n-1}$$

Oh my that can be thought of as a random variable

Var of Sample Mean

$$\mathrm{Var}(\bar{X}) = \frac{S^2}{n}$$

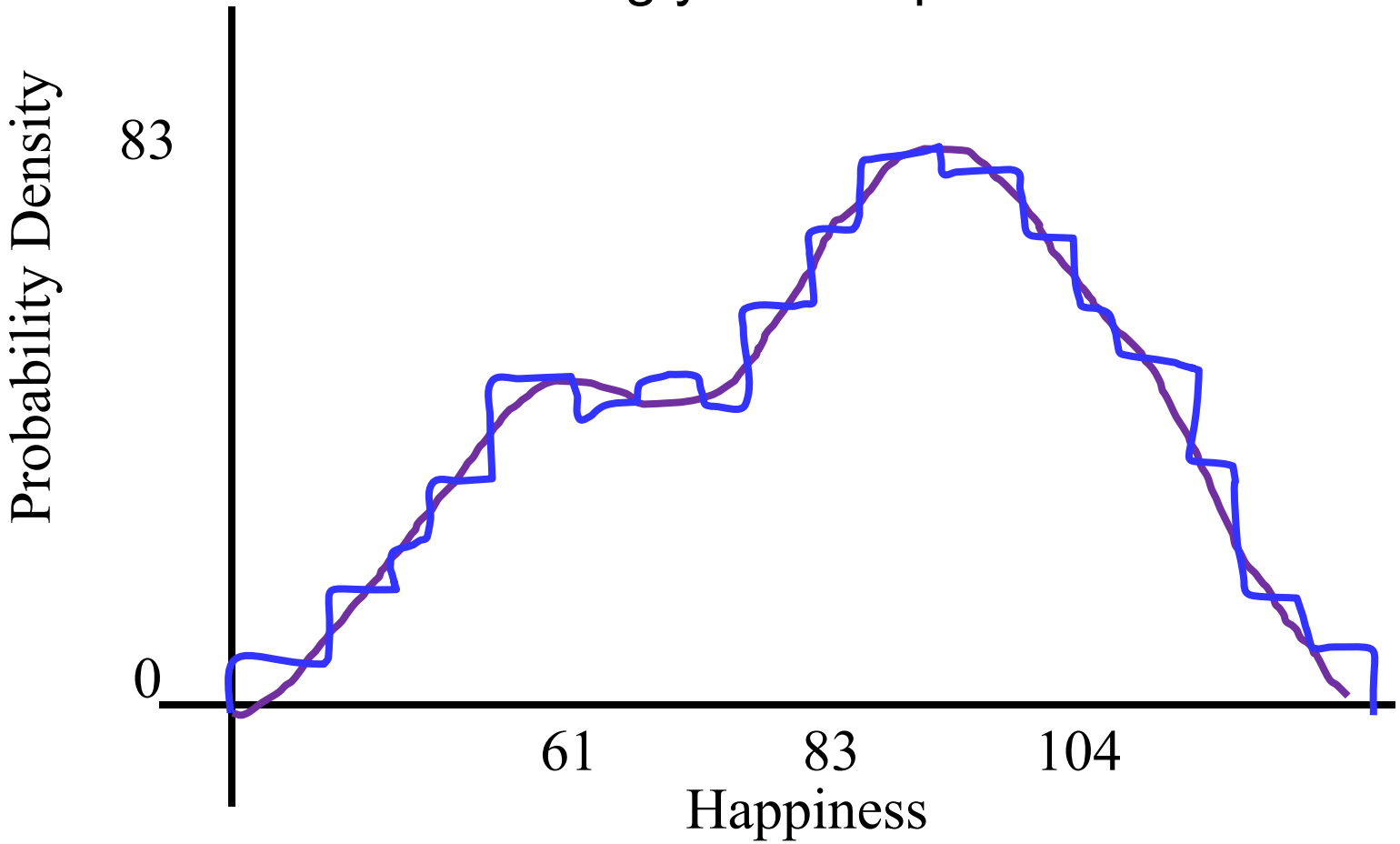
Sample Mean



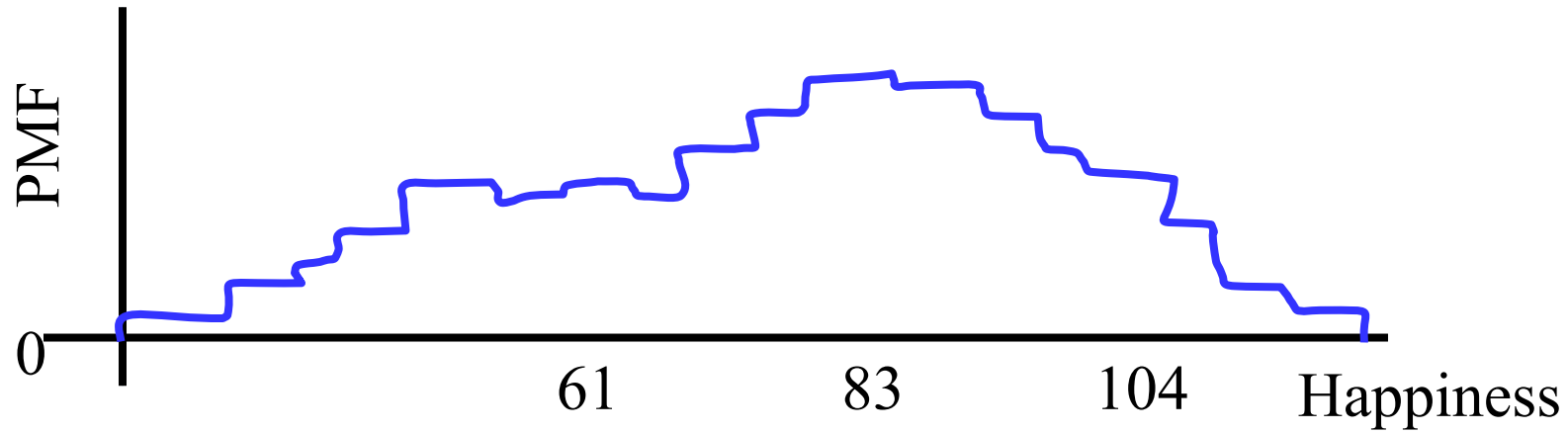
Claim: The average happiness of Bhutan is 83 ± 2

Bootstrap Insight

You can estimate the PMF of the underlying distribution, using your sample.



Bootstrap for Any Stat



Bootstrap Algorithm (sample):

1. Repeat **10,000** times:
   a. **Choose len(sample) elems from sample, with replacement**
   b. Recalculate the stat on the resample
2. You now have a **distribution of your stat**

Bootstrap for p values

Null Hypothesis Test

Population 1	Population 2
4.44	2.15
3.36	3.01
5.87	2.02
2.31	1.43
...	...
3.70	1.83
$\mu_1 = 3.1$	$\mu_2 = 2.4$

End Review

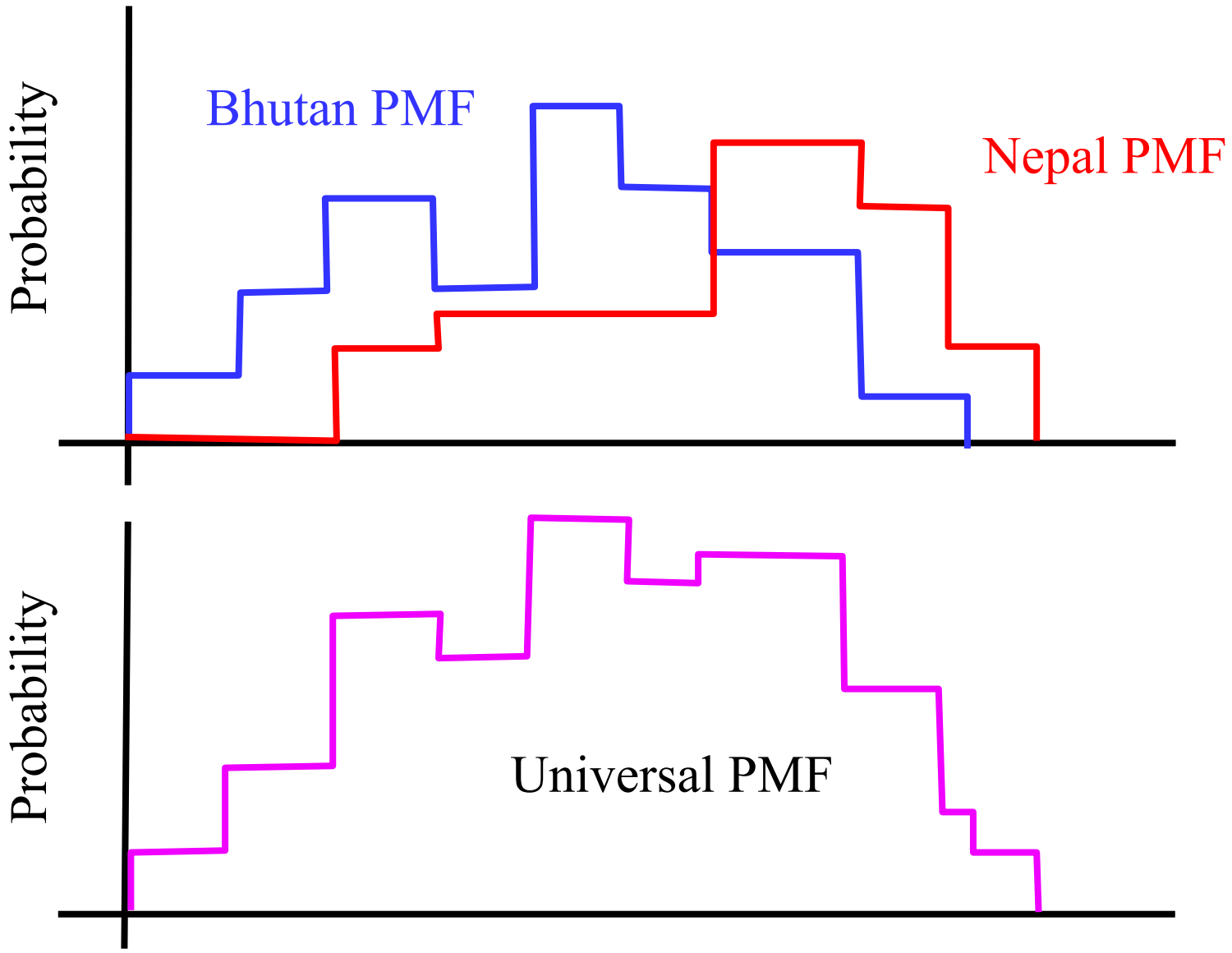
Null Hypothesis Test

Nepal Happiness	Bhutan Happiness
4.44	2.15
3.36	3.01
5.87	2.02
2.31	1.43
...	...
3.70	1.83
$\mu_1 = 3.1$	$\mu_2 = 2.4$

Claim: The difference in happiness between Nepal and Bhutan is 0.7 happiness points.

Null hypothesis: even if there is no pattern (ie the two samples are identically distributed) your results might have arisen by chance

Universal Sample



Bootstrap for P Values

```
def pvalueBootstrap(bhutanSample,nepalSample):
    N = size of the bhutanSample
    M = size of the nepalSample

    uniSamples = combine bhutanSamples and nepalSamples
    count = 0

    repeat 10,000 times:
        bhutanResample = draw N resamples from the uniSamples
        nepalResample = draw M resamples from the uniSamples
        muBhutan = sample mean of the bhutanResample
        muNepal = sample mean of the nepalResample
        meanDiff = |muNepal - muBhutan|
        if meanDiff > observedDifference:
            count += 1

    pValue = count / 10,000
```

Bootstrap for P Values

```
def pvalueBootstrap(bhutanSample,nepalSample):
    N = size of the bhutanSample
    M = size of the nepalSample

    uniSamples = combine bhutanSamples and nepalSamples
    count = 0

    repeat 10,000 times:
        bhutanResample = draw N resamples from the uniSamples
        nepalResample = draw M resamples from the uniSamples
        muBhutan = sample mean of the bhutanResample
        muNepal = sample mean of the nepalResample
        meanDiff = |muNepal - muBhutan|
        if meanDiff > observedDifference:
            count += 1

    pValue = count / 10,000
```

Bootstrap for P Values

```
def pvalueBootstrap(bhutanSample,nepalSample):
    N = size of the bhutanSample
    M = size of the nepalSample

    uniSamples = combine bhutanSamples and nepalSamples
    count = 0

    repeat 10,000 times:
        bhutanResample = draw N resamples from the uniSamples
        nepalResample = draw M resamples from the uniSamples
        muBhutan = sample mean of the bhutanResample
        muNepal = sample mean of the nepalResample
        meanDiff = |muNepal - muBhutan|
        if meanDiff > observedDifference:
            count += 1

    pValue = count / 10,000
```

Bootstrap for P Values

```
def pvalueBootstrap(bhutanSample,nepalSample):
    N = size of the bhutanSample
    M = size of the nepalSample

    uniSamples = combine bhutanSamples and nepalSamples
    count = 0

    repeat 10,000 times:
        bhutanResample = draw N resamples from the uniSamples
        nepalResample = draw M resamples from the uniSamples
        muBhutan = sample mean of the bhutanResample
        muNepal = sample mean of the nepalResample
        meanDiff = |muNepal - muBhutan|
        if meanDiff > observedDifference:
            count += 1

    pValue = count / 10,000
```

Bootstrap for P Values

```
def pvalueBootstrap(bhutanSample,nepalSample):
    N = size of the bhutanSample
    M = size of the nepalSample

    uniSamples = combine bhutanSamples and nepalSamples
    count = 0

    repeat 10,000 times:
        bhutanResample = draw N resamples from the uniSamples
        nepalResample = draw M resamples from the uniSamples
        muBhutan = sample mean of the bhutanResample
        muNepal = sample mean of the nepalResample
        meanDiff = |muNepal - muBhutan|
        if meanDiff > observedDifference:
            count += 1

    pValue = count / 10,000
```

Bootstrap for P Values

```
def pvalueBootstrap(bhutanSample,nepalSample):
    N = size of the bhutanSample
    M = size of the nepalSample

    uniSamples = combine bhutanSamples and nepalSamples
    count = 0

    repeat 10,000 times:
        bhutanResample = draw N resamples from the uniSamples
        nepalResample = draw M resamples from the uniSamples
        muBhutan = sample mean of the bhutanResample
        muNepal = sample mean of the nepalResample
        meanDiff = |muNepal - muBhutan|
        if meanDiff > observedDifference:
            count += 1

    pValue = count / 10,000
```

Bootstrap for P Values

```
def pvalueBootstrap(bhutanSample,nepalSample):
    N = size of the bhutanSample
    M = size of the nepalSample

    uniSamples = combine bhutanSamples and nepalSamples
    count = 0

    repeat 10,000 times:
        bhutanResample = draw N resamples from the uniSamples
        nepalResample = draw M resamples from the uniSamples
        muBhutan = sample mean of the bhutanResample
        muNepal = sample mean of the nepalResample
        meanDiff = |muNepal - muBhutan|
        if meanDiff > observedDifference:
            count += 1

    pValue = count / 10,000
```

Bootstrap for P Values

```
def pvalueBootstrap(bhutanSample,nepalSample):
    N = size of the bhutanSample
    M = size of the nepalSample

    uniSamples = combine bhutanSamples and nepalSamples
    count = 0

    repeat 10,000 times:
        bhutanResample = draw N resamples from the uniSamples
        nepalResample = draw M resamples from the uniSamples
        muBhutan = sample mean of the bhutanResample
        muNepal = sample mean of the nepalResample
        meanDiff = |muNepal - muBhutan|
        if meanDiff > observedDifference:
            count += 1

    pValue = count / 10,000
```

With replacement!

Algorithm in Practice

```
def drawWithReplace(samples):
    # Estimate the PMF using the samples
    # Draw K new samples from the PMF
```

Algorithm in Practice

```
def drawWithReplace(samples):
    # Estimate the PMF using the samples
    # Draw K new samples from the PMF
    return np.random.choice(samples, K)
```

Bootstrap

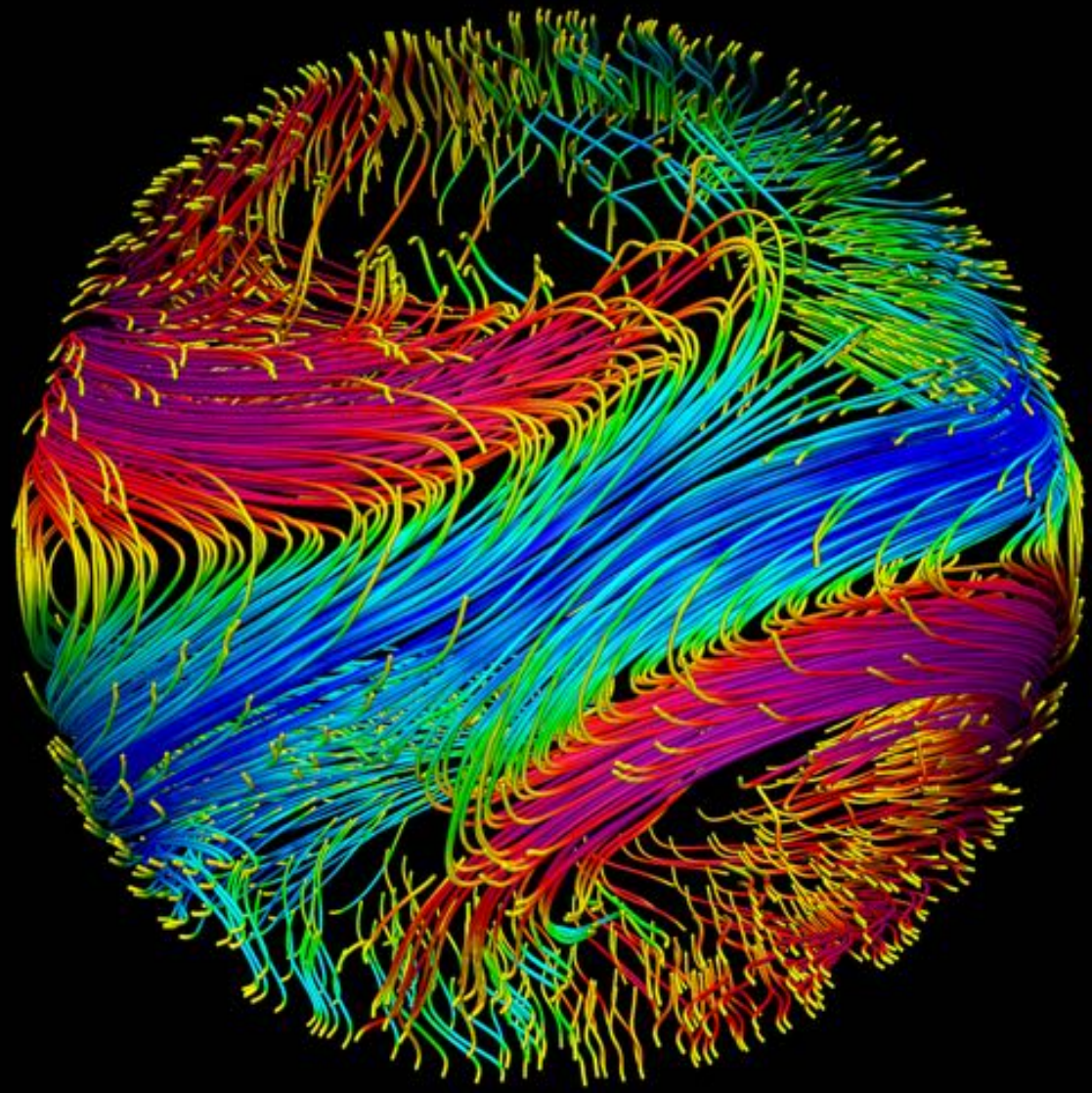
Lets try it!

Null Hypothesis Test

Nepal Happiness	Bhutan Happiness
4.44	2.15
3.36	3.01
5.87	2.02
2.31	1.43
...	...
3.70	1.83
$\mu_1 = 3.1$	$\mu_2 = 2.4$

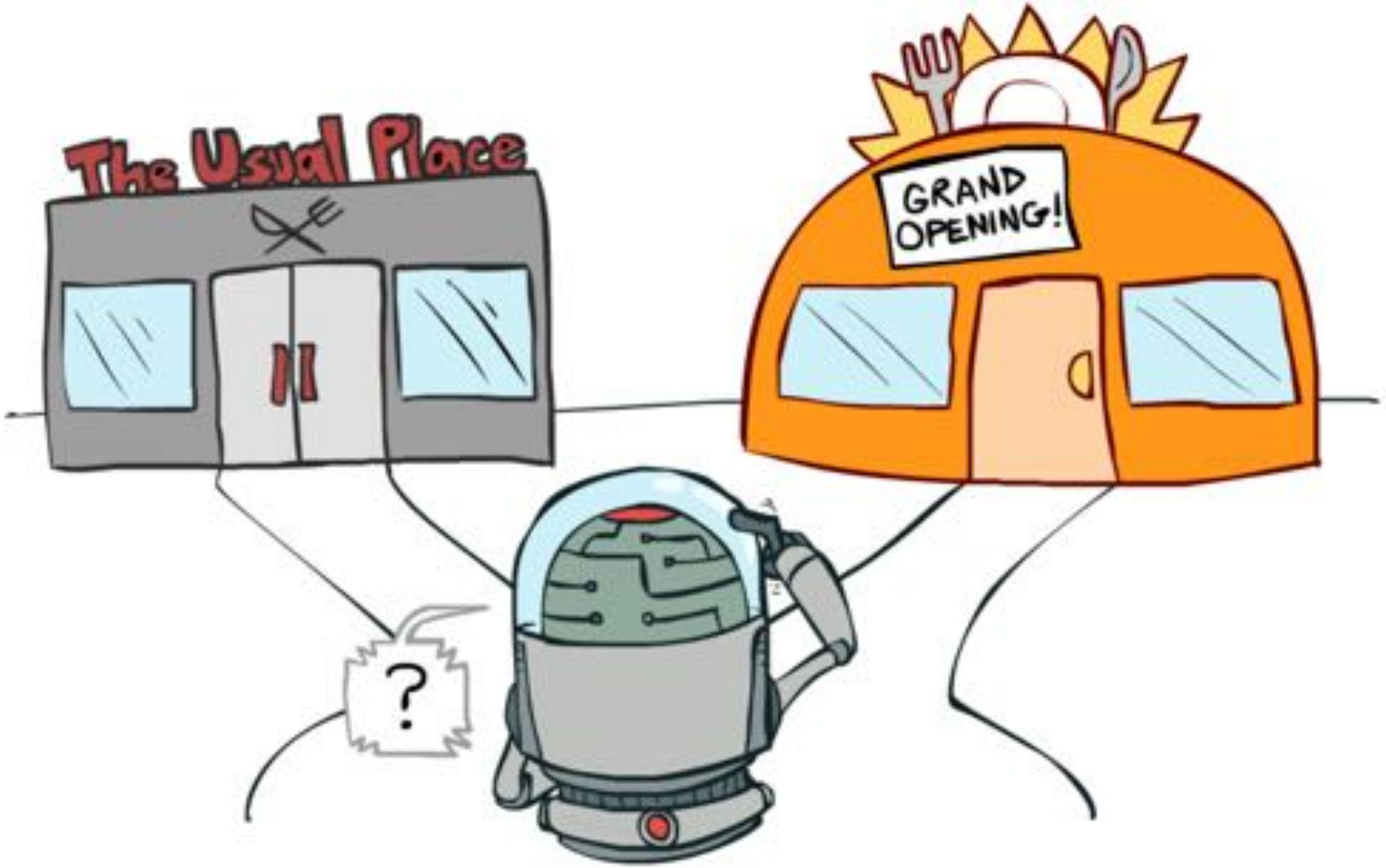
Claim: The difference in happiness between Nepal and Bhutan is 0.7 happiness points (p = 0.008).

Randomized Algorithms



Bootstrapping

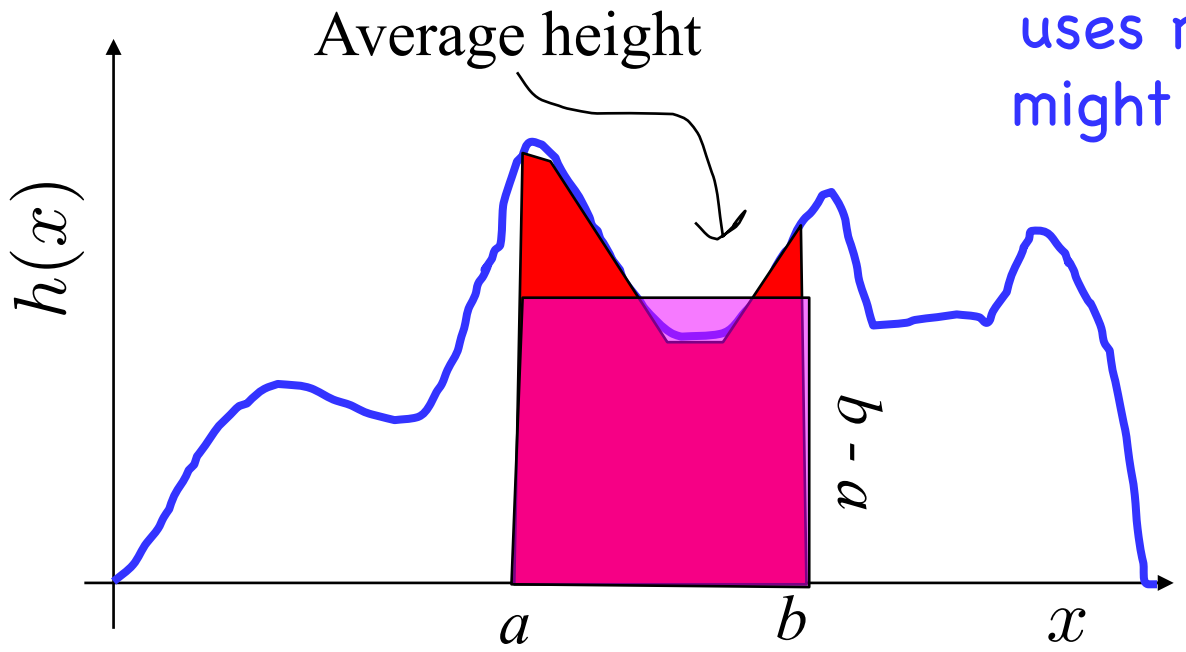
Thompson Sampling



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:

$$\int_a^b h(x)\,dx \approx \frac{(b-a)}{N} \sum_{i=1}^{N} h(X_i)$$



A "Monte Carlo" algorithm uses randomization but might not get the right answer

A Rose by Any Other Name

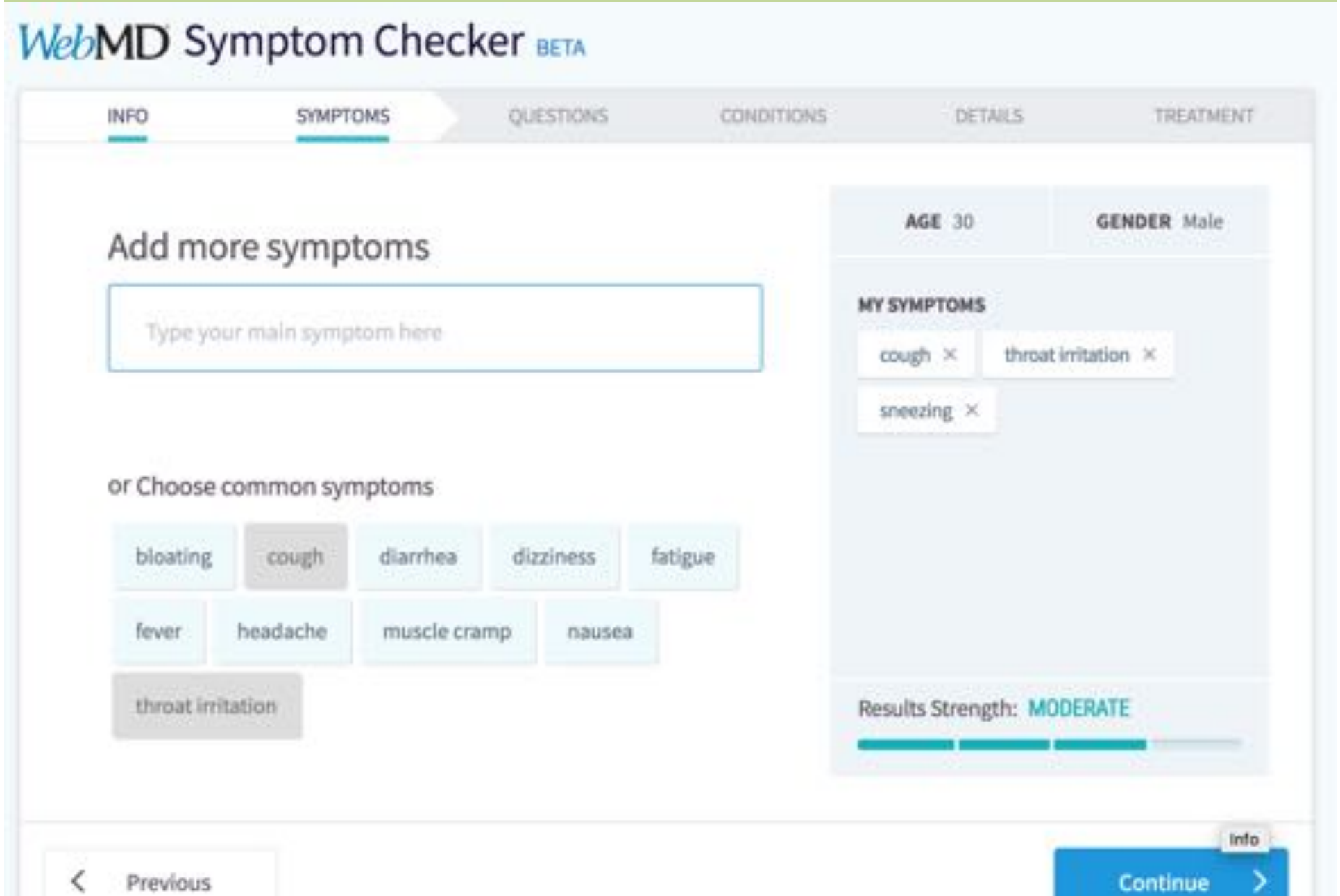
Las Vegas, Nevada

Monte Carlo, Monaco

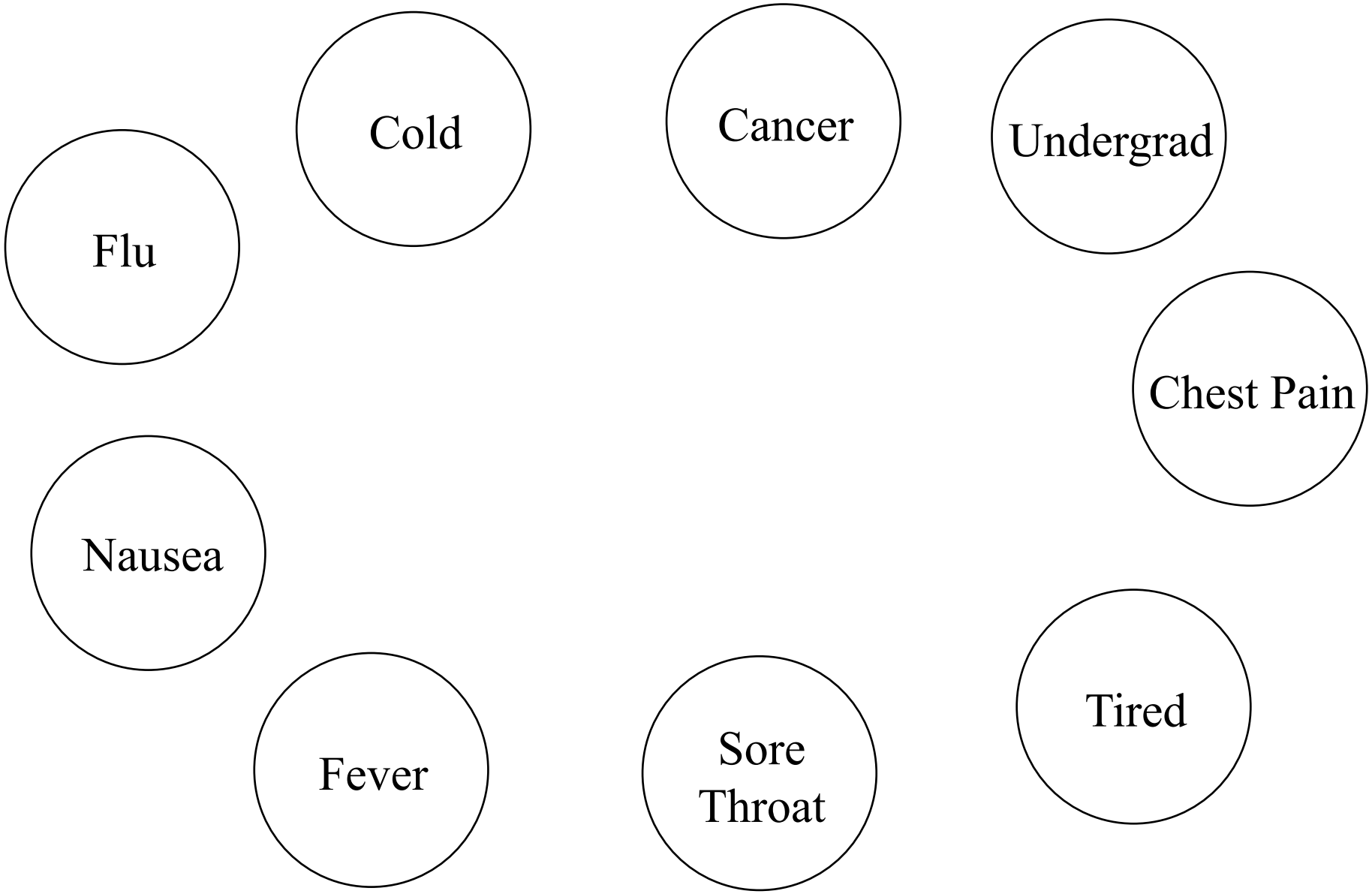
Something brand **new**…

General “Inference”

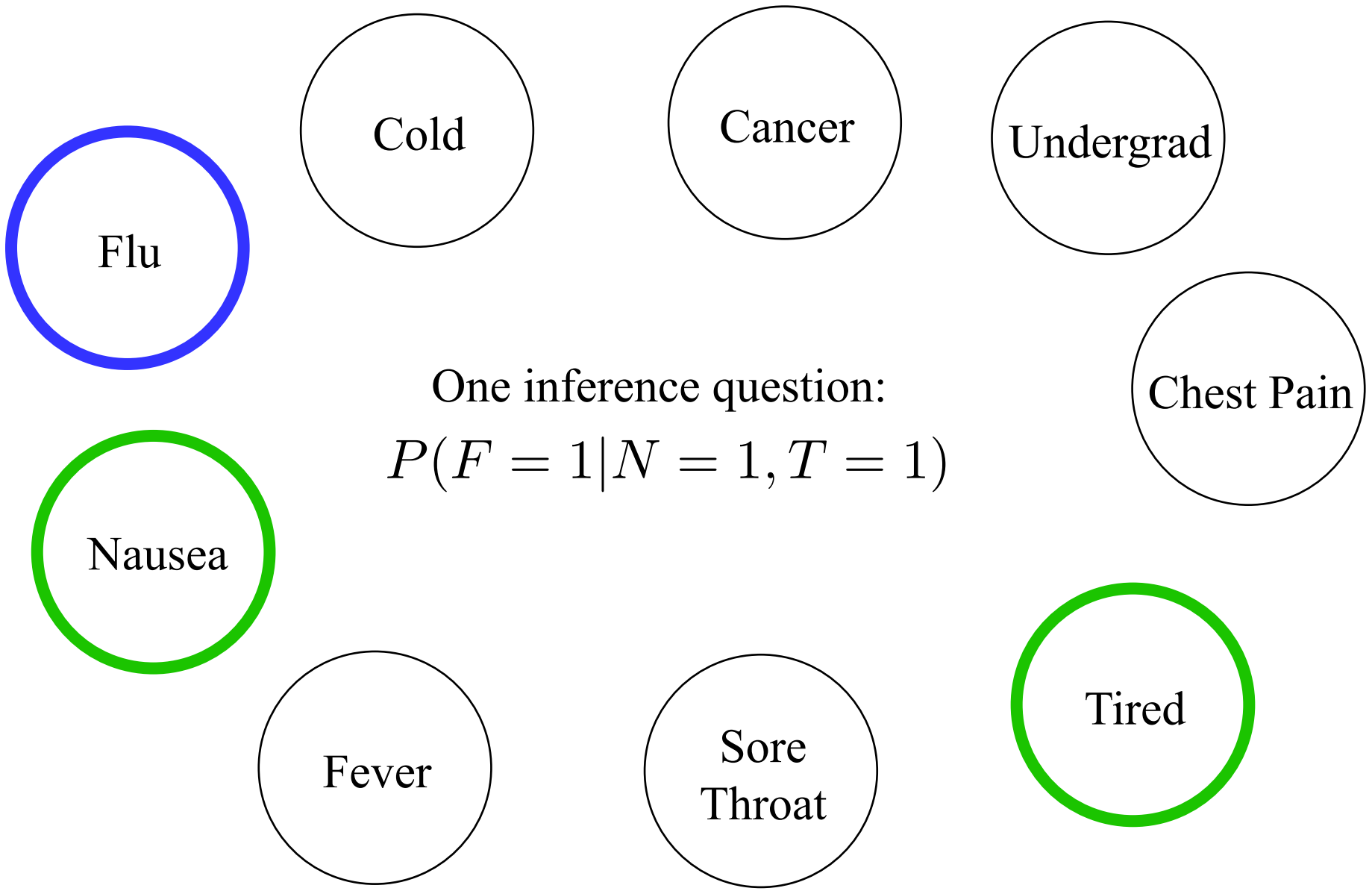
General “Inference”



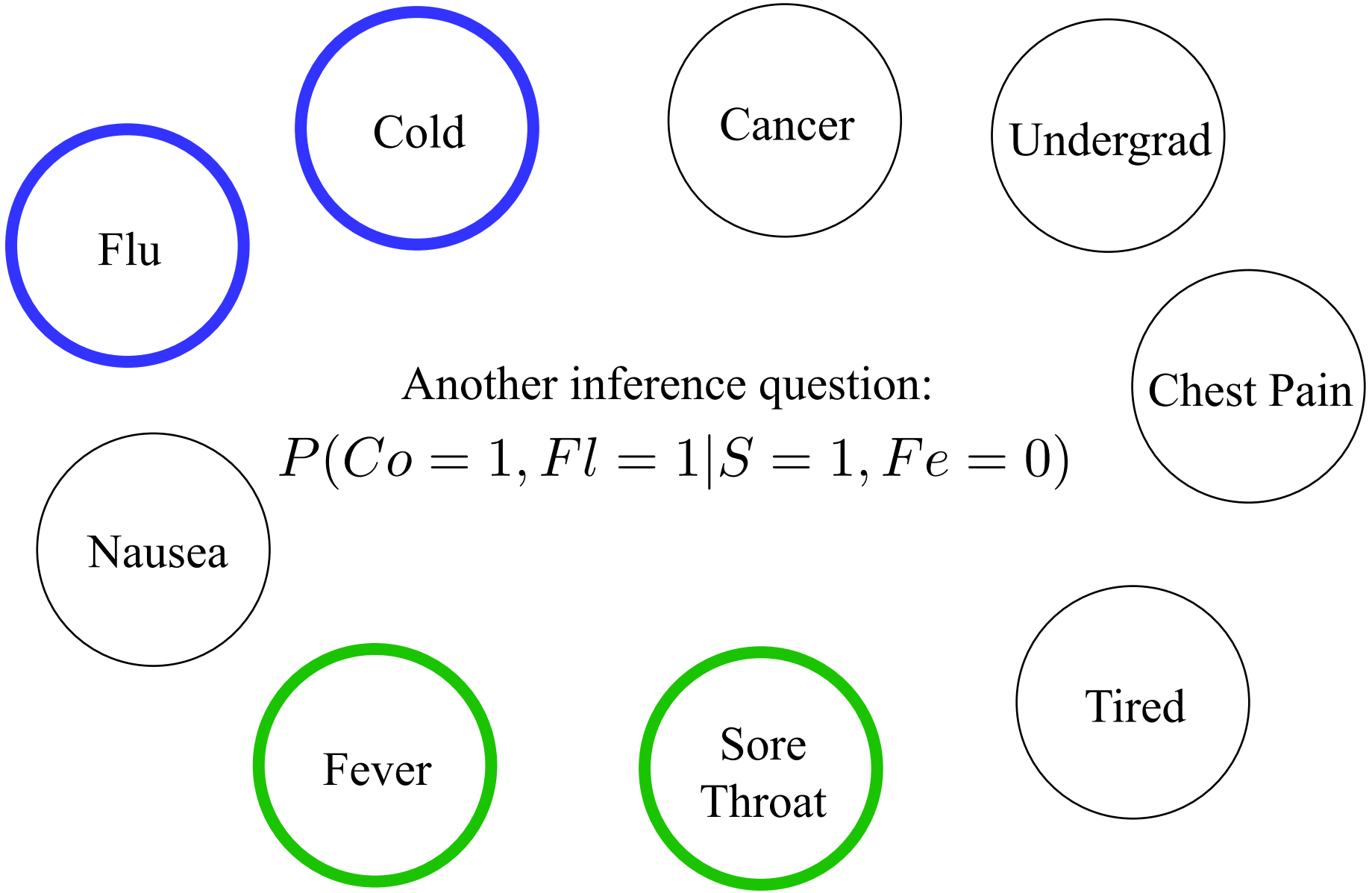
General "Inference"



General "Inference"



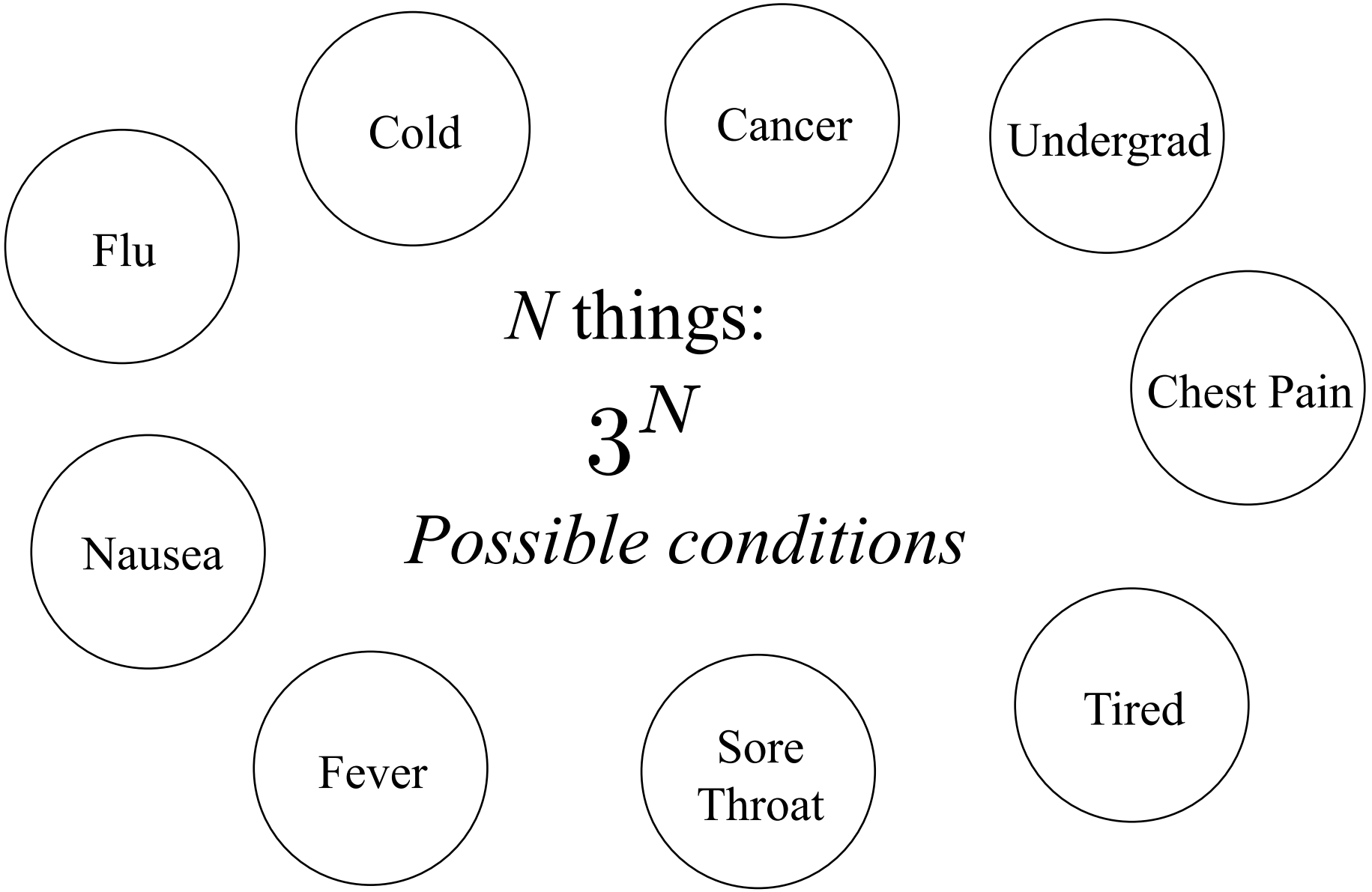
General "Inference"



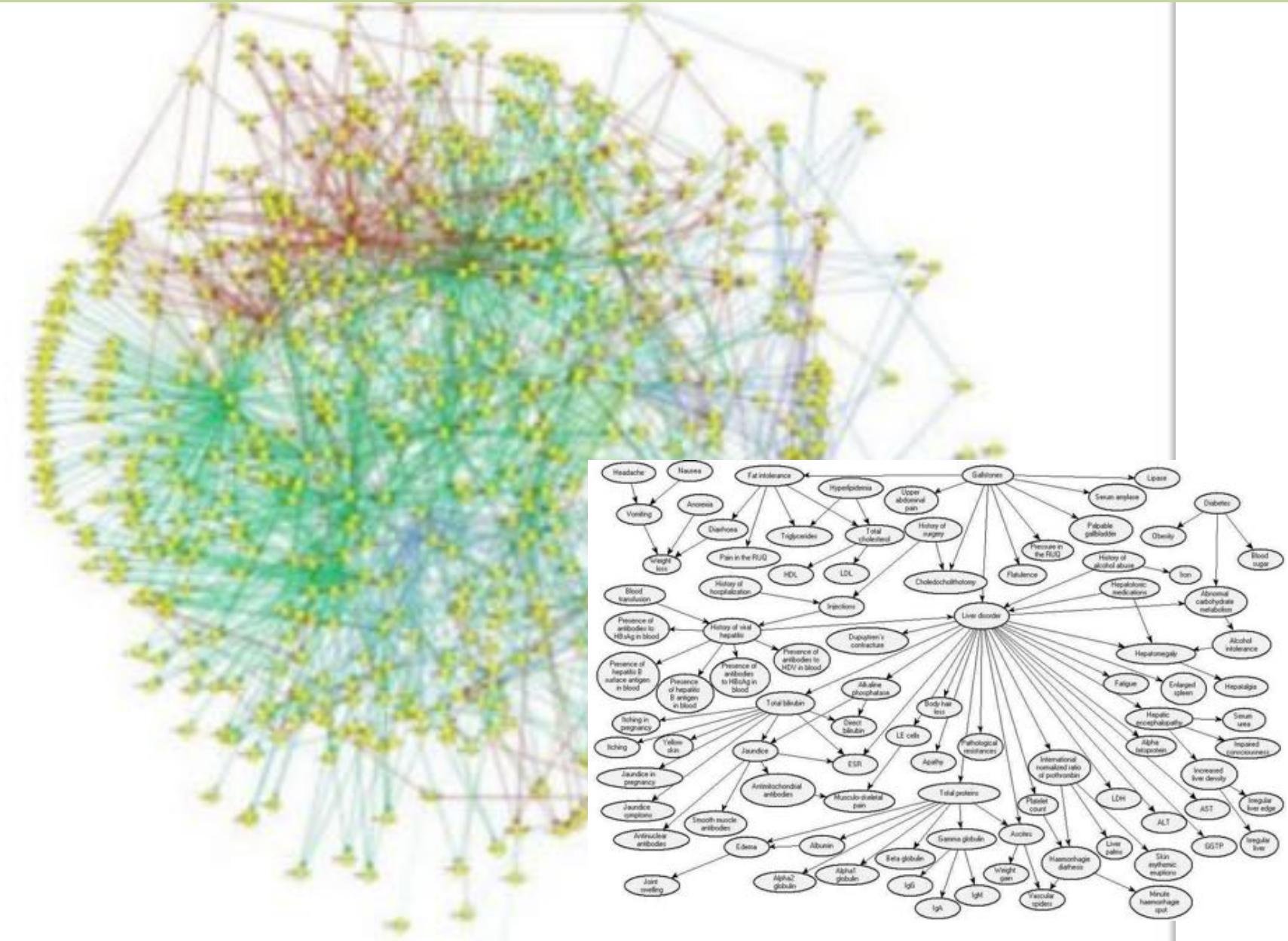
Another inference question:

$$P(Co = 1, Fl = 1 | S = 1, Fe = 0)$$

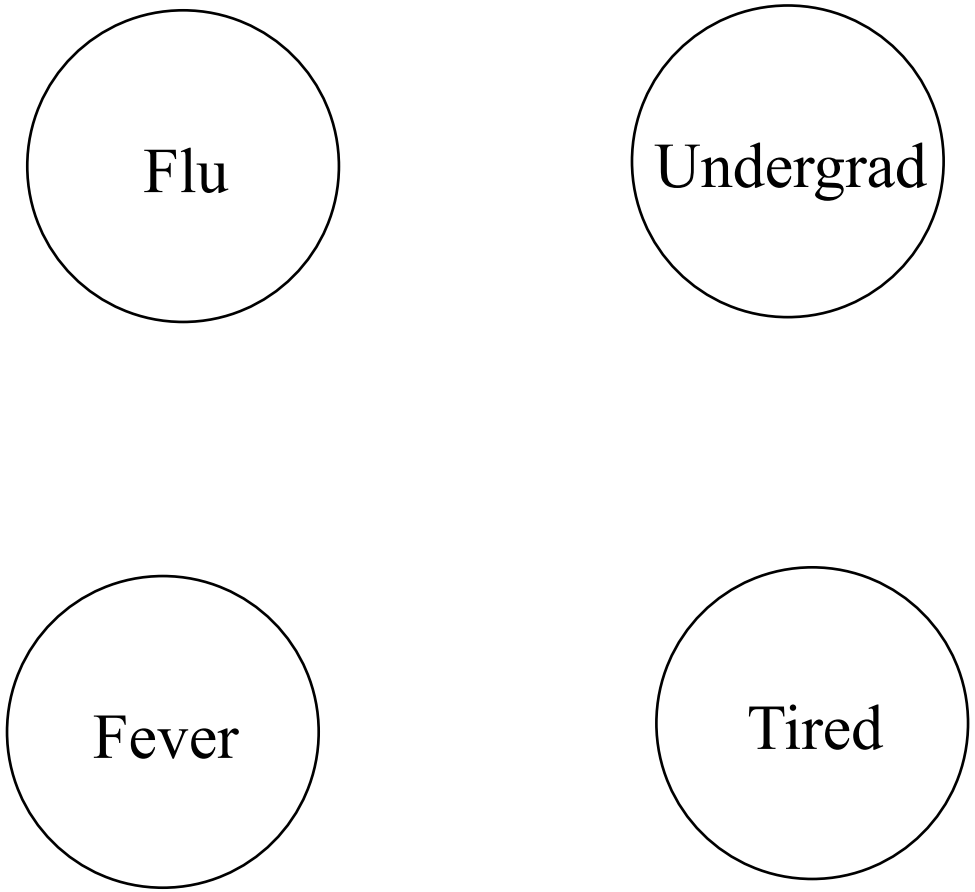
How Many Things Can You Condition On?



N is large…


Headache
Nausea
Vomiting
Anorexia
Diarrhoea
Weight loss
Fat intolerance
Triglycerides
Hyperlipidemia
Pain in the RUQ
HDL
LDL
Total cholesterol
Upper abdominal pain
History of surgery
Gallstones
Choledocholithotomy
Lipase
Serum amylase
Palpable gallbladder
Pressure in the RUQ
Flatulence
Diabetes
Obesity
Blood sugar
History of alcohol abuse
Iron
Hepatotoxic medications
Abnormal carbohydrate metabolism
Alcohol intolerance
Hepatomegaly
Fatigue
Enlarged spleen
Hepatalgia
Hepatic encephalopathy
Serum urea
Impaired consciousness
Alpha fetoprotein
Increased liver density
Irregular liver edge
AST
ALT
GGTP
Irregular liver
LDH
Liver palms
Skin erythemic eruptions
Minute haemorrhage spot
Haemorrhage diathesis
Ascites
Weight gain
Vascular spiders
Liver disorder
Blood transfusion
History of hospitalization
Injections
History of viral hepatitis
Presence of antibodies to HBsAg in blood
Presence of hepatitis B surface antigen in blood
Presence of hepatitis B antigen in blood
Presence of antibodies to HBcAg in blood
Presence of antibodies to HDV in blood
Dupuytren's contracture
Alkaline phosphatase
Total bilirubin
Direct bilirubin
Body hair loss
LE cells
Pathological resistances
Apathy
Itching in pregnancy
Itching
Yellow skin
Jaundice
ESR
Jaundice in pregnancy
Jaundice symptoms
Antimitochondrial antibodies
Musculo-skeletal pain
Total proteins
International normalized ratio of prothrombin
Platelet count
Antinuclear antibodies
Smooth muscle antibodies
Edema
Albumin
Gamma globulin
Beta globulin
Alpha2 globulin
Alpha1 globulin
IgG
IgM
IgA
Joint swelling

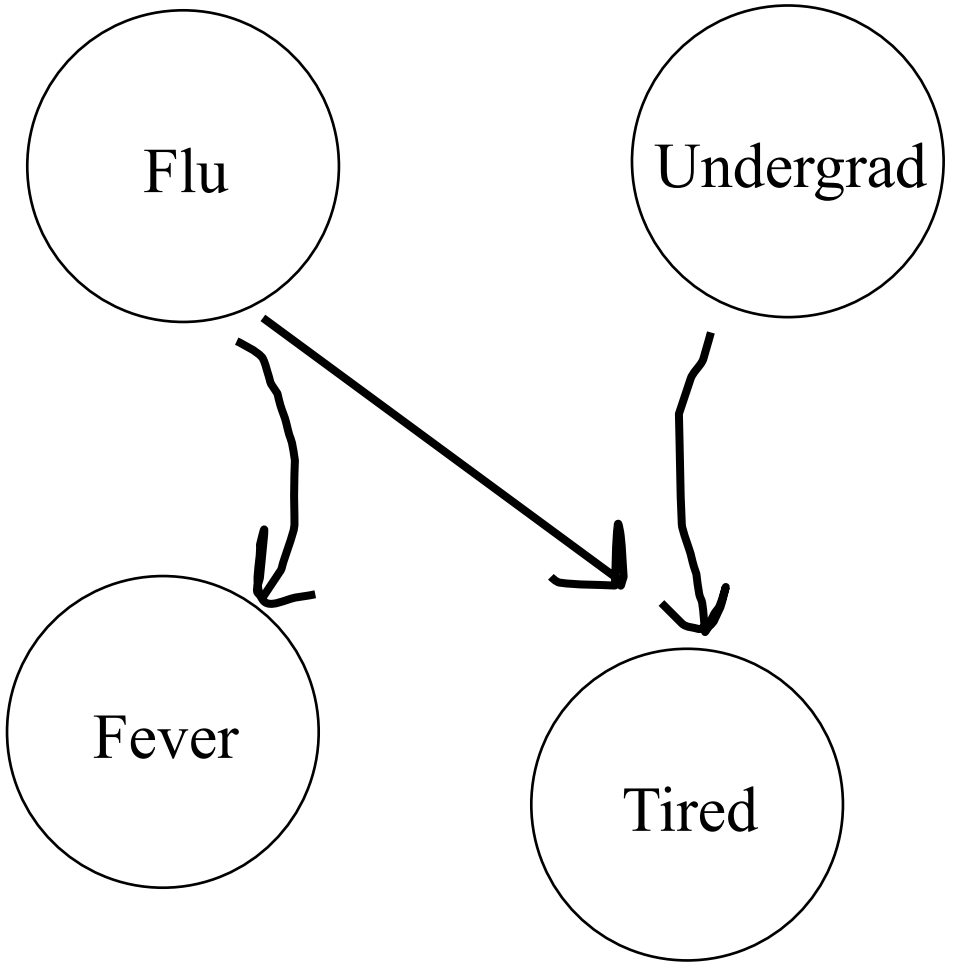
Simple WebMd



Naively specifying a joint is often impossible…

Probabilistic Model

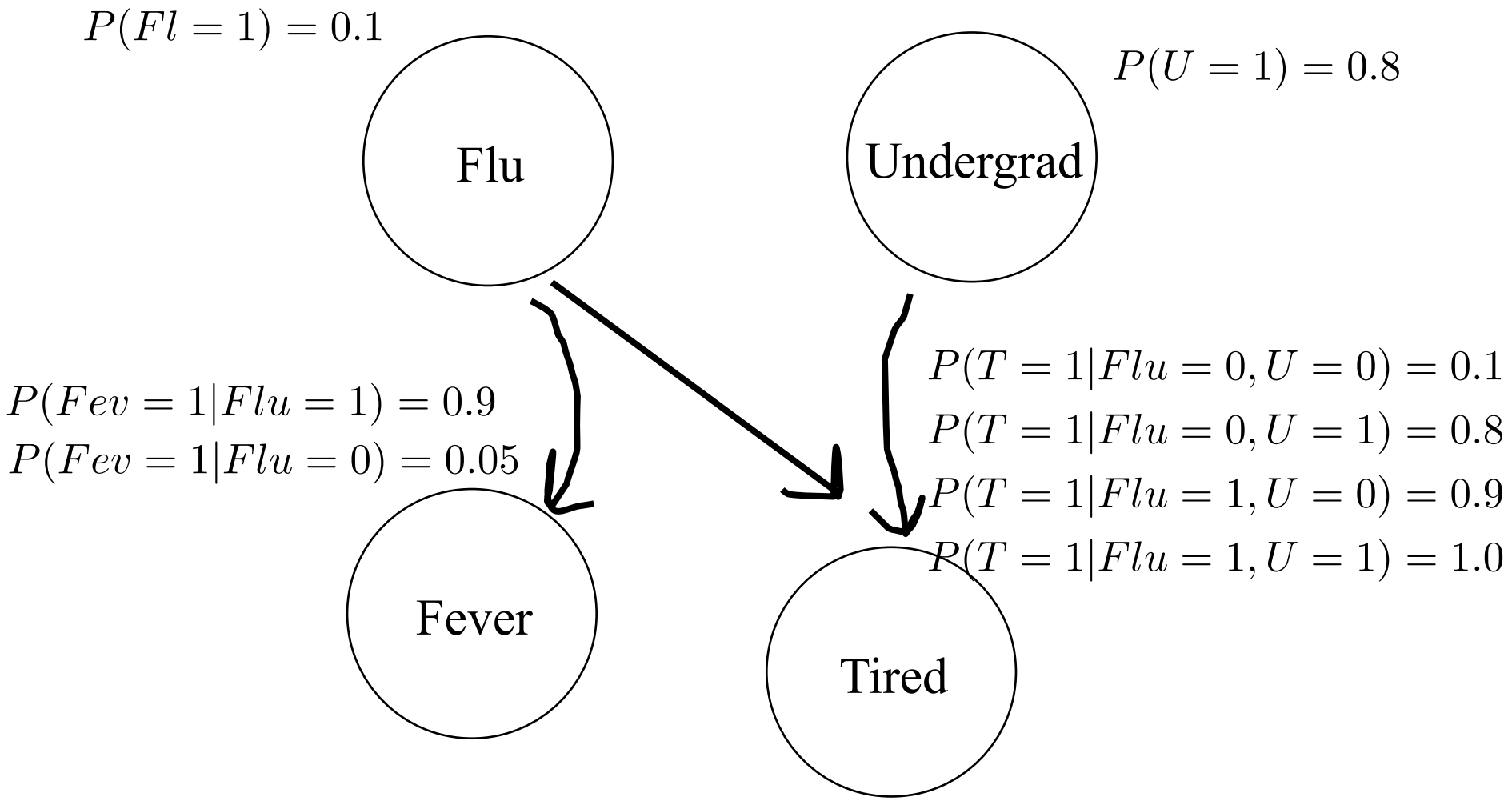
Describe the joint using causality!



$$P(Fl = a, Fe = b, U = c, T = d)?$$

Probabilistic Model

Describe the joint using causality!



$$P(Fl = a, Fe = b, U = c, T = d)?$$

If you know causality,

Make a network of causality for you random vars.

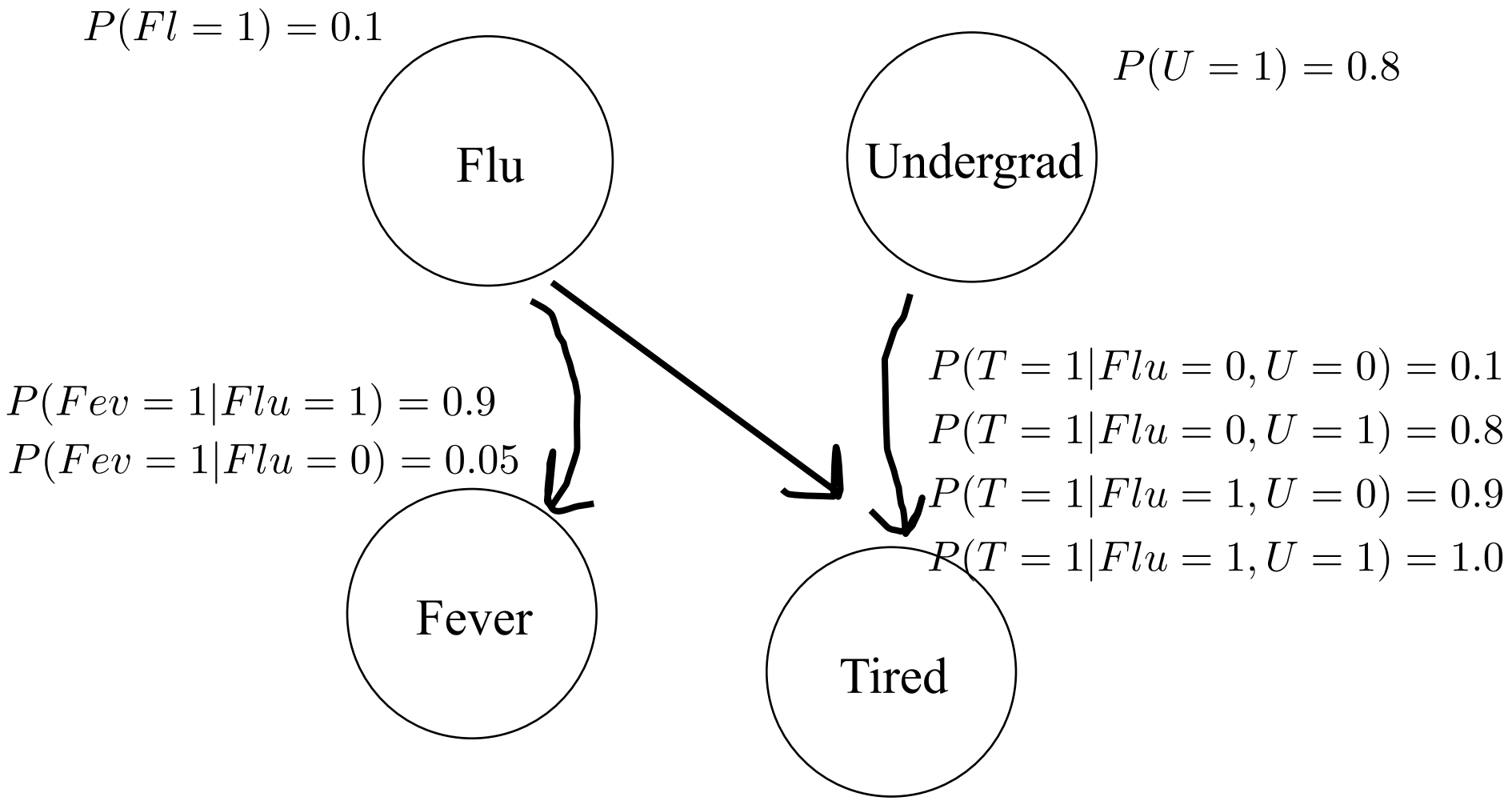
To define a joint you simply need give:

$P(\text{values} \mid \text{parents})$

for each random variable.

Prob can be a conditional probability table or an equation!

Probabilistic Model



Alg #0: Straight Math

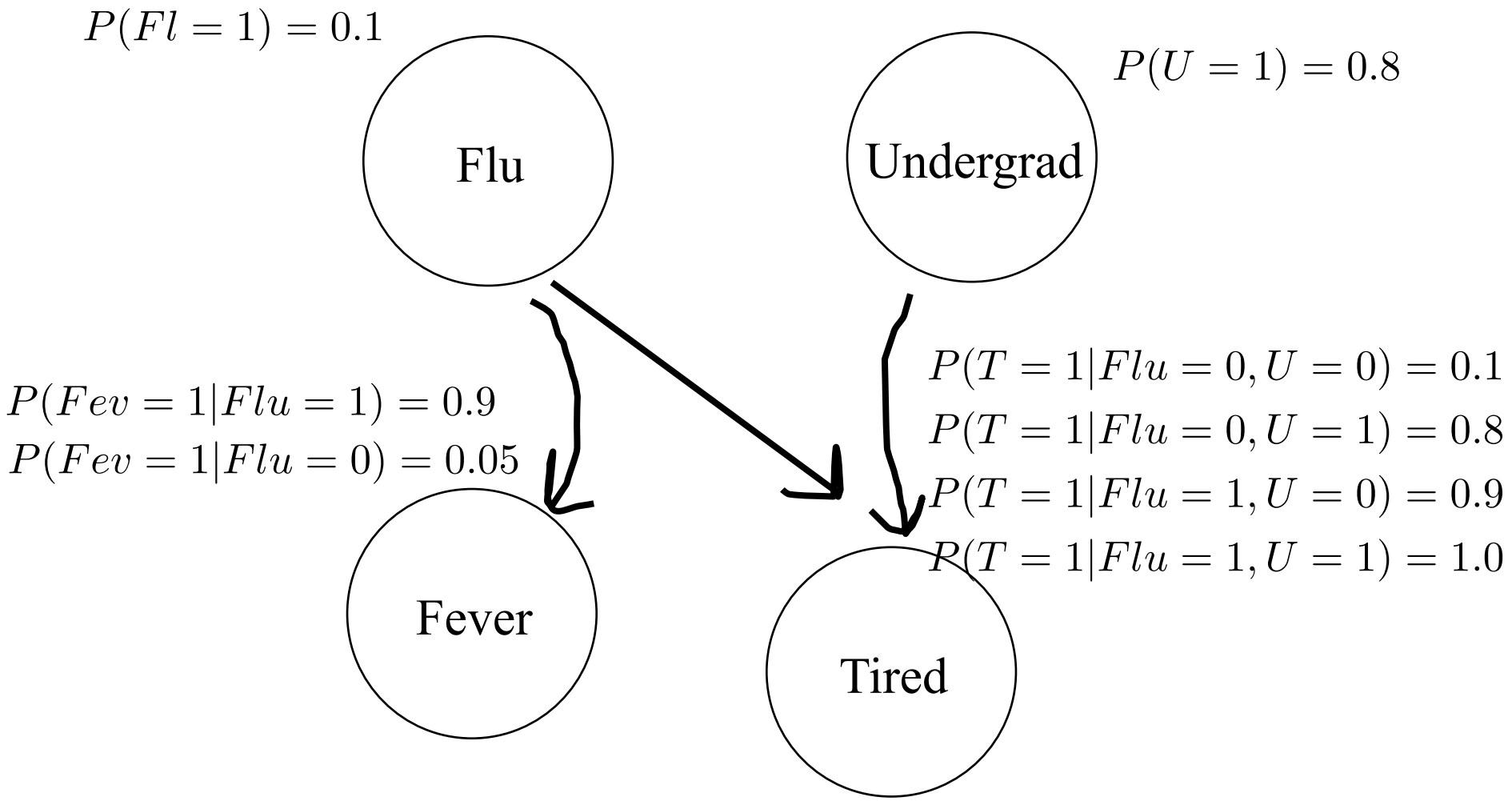
Too many possible **inference** questions one could ask…

Alg #1: Joint Sampling

```
3  N_SAMPLES = 100000
4
5  # Program: Joint Sample
6  # ------------------------
7  # we can answer any probability question
8  # with multivariate samples from the joint,
9  # where conditioned variables match
10 def main():
11     obs = getObservation()
12     print 'Observation = ', obs
13
14     samples = sampleATon()
15     prob = probFluGivenObs(samples, obs)
16     print 'Pr(Flu) = ', prob
```

```
71  # Method: Sample A Ton
72  # ------------------------
73  # chose N_SAMPLES with likelhood proportional
74  # to the joint distribution
75  def sampleATon():
76      samples = []
77      for i in range(N_SAMPLES):
78          sample = makeSample()
79          samples.append(sample)
80      return samples
```

Recall: Probabilistic Model



```
82  # Method: Make Sample
83  # ---------------------
84  # chose a single sample from the joint distribut
85  # based on the medical "Probabilistic Graphical I
86  def makeSample():
87      # prior on causal factors
88      flu = bern(0.1)
89      und = bern(0.8)
90
91      # choose fever based on flue
92      if flu == 1: fev = bern(0.9)
93      else:        fev = bern(0.05)
94
95      # choose tired based on (undergrade and flu)
96      if und == 1 and flu == 1:   tir = bern(1.0)
97      elif und == 1 and flu == 0: tir = bern(0.8)
98      elif und == 0 and flu == 1: tir = bern(0.9)
99      else:                       tir = bern(0.1)
100
101     # a sample from the joint has an
102     # assignment to *all* random variables
103     return [flu, und, fev, tir]
```

```
82  # Method: Make Sample
83  # ---------------------
84  # chose a single sample from the joint distribut
85  # based on the medical "Probabilistic Graphical I
86  def makeSample():
87      # prior on causal factors
88      flu = bern(0.1)
89      und = bern(0.8)
90
91      # choose fever based on flue
92      if flu == 1: fev = bern(0.9)
93      else:        fev = bern(0.05)
94
95      # choose tired based on (undergrade and flu)
96      if und == 1 and flu == 1:   tir = bern(1.0)
97      elif und == 1 and flu == 0: tir = bern(0.8)
98      elif und == 0 and flu == 1: tir = bern(0.9)
99      else:                       tir = bern(0.1)
100
101     # a sample from the joint has an
102     # assignment to *all* random variables
103     return [flu, und, fev, tir]
```

```
82  # Method: Make Sample
83  # ---------------------
84  # chose a single sample from the joint distribut
85  # based on the medical "Probabilistic Graphical I
86  def makeSample():
87      # prior on causal factors
88      flu = bern(0.1)
89      und = bern(0.8)
90
91      # choose fever based on flue
92      if flu == 1: fev = bern(0.9)
93      else:        fev = bern(0.05)
94
95      # choose tired based on (undergrade and flu)
96      if und == 1 and flu == 1:   tir = bern(1.0)
97      elif und == 1 and flu == 0: tir = bern(0.8)
98      elif und == 0 and flu == 1: tir = bern(0.9)
99      else:                       tir = bern(0.1)
100
101     # a sample from the joint has an
102     # assignment to *all* random variables
103     return [flu, und, fev, tir]
```

```
82  # Method: Make Sample
83  # ---------------------
84  # chose a single sample from the joint distribut
85  # based on the medical "Probabilistic Graphical I
86  def makeSample():
87      # prior on causal factors
88      flu = bern(0.1)
89      und = bern(0.8)
90
91      # choose fever based on flue
92      if flu == 1: fev = bern(0.9)
93      else:        fev = bern(0.05)
94
95      # choose tired based on (undergrade and flu)
96      if und == 1 and flu == 1:   tir = bern(1.0)
97      elif und == 1 and flu == 0: tir = bern(0.8)
98      elif und == 0 and flu == 1: tir = bern(0.9)
99      else:                       tir = bern(0.1)
100
101     # a sample from the joint has an
102     # assignment to *all* random variables
103     return [flu, und, fev, tir]
```

Alg #1: Joint Sampling

```
3  N_SAMPLES = 100000
4
5  # Program: Joint Sample
6  # ------------------------
7  # we can answer any pro
8  # with multivariate sam
9  # where conditioned var
10 def main():
11     obs = getObservatio
12     print 'Observation
13
14     samples = sampleATo
15     prob = probFluGiven
16     print 'Pr(Flu) = ',
```

webMd — -bash — 30×20

```
[0, 1, 0, 1]
[1, 1, 1, 1]
[0, 1, 0, 1]
[0, 1, 0, 0]
[0, 1, 0, 0]
[0, 1, 0, 1]
[0, 1, 0, 1]
[0, 0, 0, 0]
[0, 0, 0, 0]
[0, 1, 0, 1]
[0, 1, 0, 1]
[0, 1, 0, 1]
[0, 1, 0, 1]
[0, 1, 0, 1]
[0, 1, 0, 1]
[0, 1, 0, 0]
[0, 0, 0, 0]
[0, 1, 0, 1]
[0, 1, 0, 1]
[0, 1, 0, 1]
[1, 1, 0, 1]
```

Alg #1: Joint Sampling

```
3  N_SAMPLES = 100000
4
5  # Program: Joint Sample
6  # ----------------------------
7  # we can answer any probability question
8  # with multivariate samples from the joint,
9  # where conditioned variables match
10 def main():
11     obs = getObservation()
12     print 'Observation = ', obs
13
14     samples = sampleATon()
15     prob = probFluGivenObs(samples, obs)
16     print 'Pr(Flu) = ', prob
```

```
25 # Method: Probability of Flu Given Observation
26 # ----------------------------------
27 # Calculate the probability of flu given many
28 # samples from the joint distribution and a set
29 # of ovservations to condition on.
30 def probFluGivenObs(samples, obs):
31     # reject all samples which don't align
32     # with condition
33     keepSamples = []
34     for sample in samples:
35         if checkObsMatch(sample, obs):
36             keepSamples.append(sample)
37
38     # from remaining, simply count...
39     fluCount = 0
40     for sample in keepSamples:
41         [flu, und, fev, tir] = sample
42         if flu == 1:
43             fluCount += 1
44
45     # counting can be so sweet...
46     return float(fluCount) / len(keepSamples)
```

```
25 # Method: Probability of Flu Given Observation
26 # ---------------------------------
27 # Calculate the probability of flu given many
28 # samples from the joint distribution and a set
29 # of ovservations to condition on.
30 def probFluGivenObs(samples, obs):
31     # reject all samples which don't align
32     # with condition
33     keepSamples = []
34     for sample in samples:
35         if checkObsMatch(sample, obs):
36             keepSamples.append(sample)
37
38     # from remaining, simply count...
39     fluCount = 0
40     for sample in keepSamples:
41         [flu, und, fev, tir] = sample
42         if flu == 1:
43             fluCount += 1
44
45     # counting can be so sweet...
46     return float(fluCount) / len(keepSamples)
```

```
25 # Method: Probability of Flu Given Observation
26 # ----------------------------
27 # Calculate the probability of flu given many
28 # samples from the joint distribution and a set
29 # of ovservations to condition on.
30 def probFluGivenObs(samples, obs):
31     # reject all samples which don't align
32     # with condition
33     keepSamples = []
34     for sample in samples:
35         if checkObsMatch(sample, obs):
36             keepSamples.append(sample)
37
38     # from remaining, simply count...
39     fluCount = 0
40     for sample in keepSamples:
41         [flu, und, fev, tir] = sample
42         if flu == 1:
43             fluCount += 1
44
45     # counting can be so sweet...
46     return float(fluCount) / len(keepSamples)
```

Alg #1: Joint Sampling

```
3  N_SAMPLES = 100000
4
5  # Program: Joint Sample
6  # ------------------------
7  # we can answer any pro
8  # with multivariate sam
9  # where conditioned var
10 def main():
11     obs = getObservatio
12     print 'Observation
13
14     samples = sampleATo
15     prob = probFluGiven
16     print 'Pr(Flu) = ',
```

```
webMd — -bash — 30×20
[0, 1, 0, 1]
[0, 1, 0, 0]
[0, 1, 0, 0]
[0, 1, 0, 1]
[0, 1, 0, 1]
[0, 0, 0, 0]
[0, 0, 0, 0]
[0, 1, 0, 1]
[0, 1, 0, 1]
[0, 1, 0, 1]
[0, 1, 0, 1]
[0, 1, 0, 1]
[0, 1, 0, 1]
[0, 1, 0, 0]
[0, 0, 0, 0]
[0, 1, 0, 1]
[0, 1, 0, 1]
[1, 1, 0, 1]
Pr(Flu) =  0.141503173687
>
```

The Magic School Bus™

webMd — -bash — 39×20

```
[0, 1, 1, 0]
[1, 0, 1, 1]
[0, 1, 0, 1]
[0, 1, 0, 0]
[0, 1, 0, 0]
[0, 1, 1, 0]
[1, 1, 1, 1]
[0, 1, 0, 0]
[0, 0, 0, 1]
[0, 1, 0, 1]
[0, 1, 0, 1]
[0, 1, 0, 1]
[0, 1, 0, 0]
[0, 1, 0, 1]
[0, 1, 0, 0]
[0, 0, 0, 0]
[0, 0, 0, 1]
Observation =  [None, None, None, None]
Pr(Flu | Obs) =  0.10164
>
```

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

Each one of these is one joint sample:
[Flu, Undergrad, Fever, Tired]

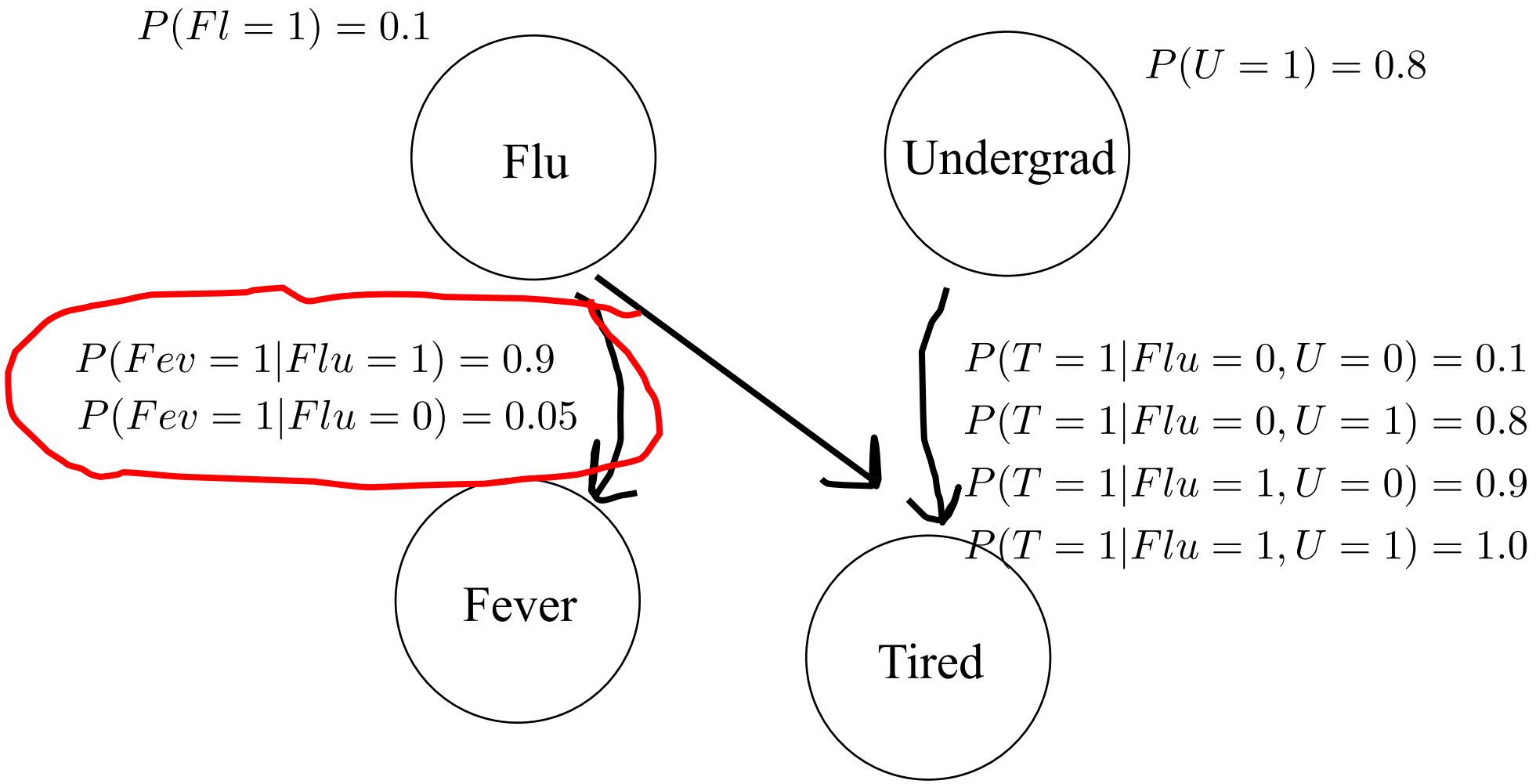
Alg #1: Joint Sampling

With enough samples:

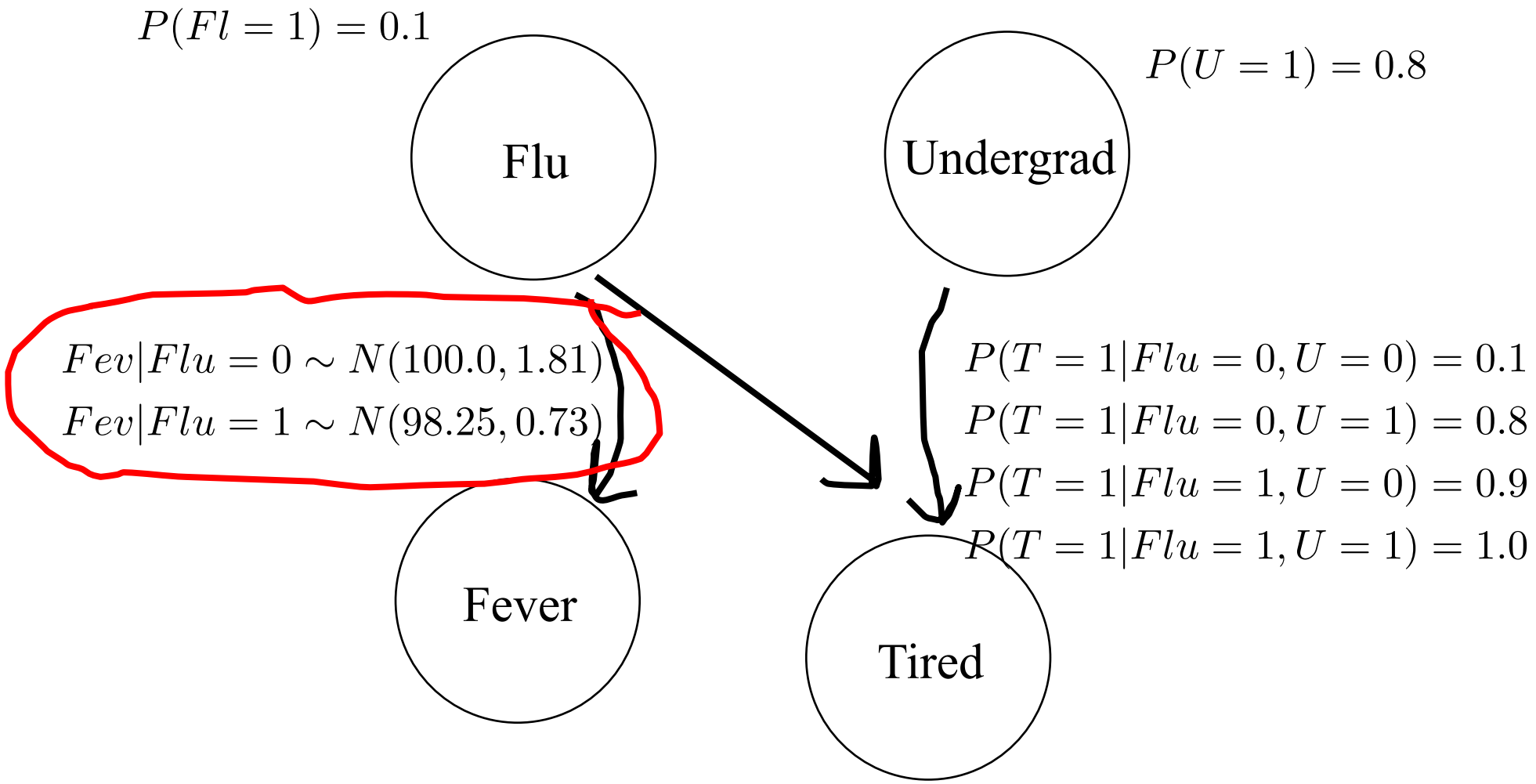
- Probability estimates will be correct
- Conditional probability estimates will be correct
- Expectation estimations will be correct

What's the matter with joint sampling?

Probabilistic Model



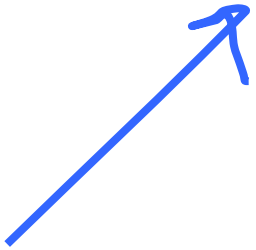
Probabilistic Model



The Magic School Bus™

Markov Chain

MCMC



Monte Carlo

Alg #2: MCMC

```
[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 joint sample:

[Flu, Undergrad, Fever, Tired]

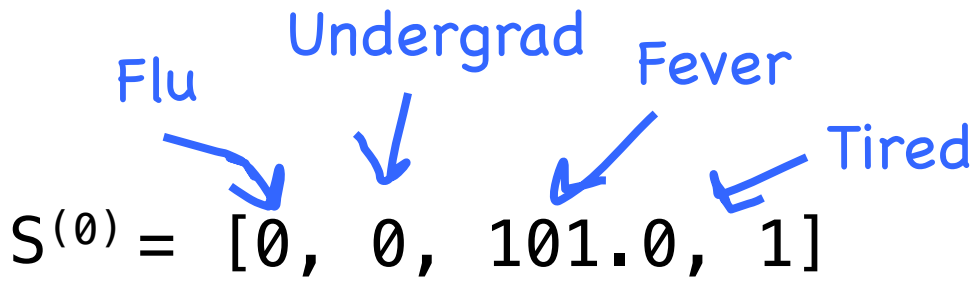
Alg #2: MCMC

```
All Samples = []
```


```
Initial Sample = [0, 0, 101.0, 1]
```

Alg #2: MCMC

```
All Samples = []
```



Alg #2: MCMC

All Samples = [$S^{(0)}$]

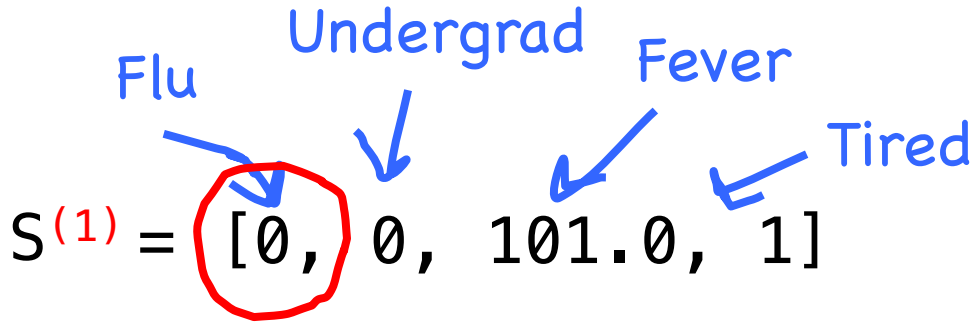
Flu, Undergrad, Fever, Tired

$S^{(0)}$ = [0, 0, 101.0, 1]

From S_t make S_{t+1}

Alg #2: MCMC

All Samples = [$S^{(0)}$]

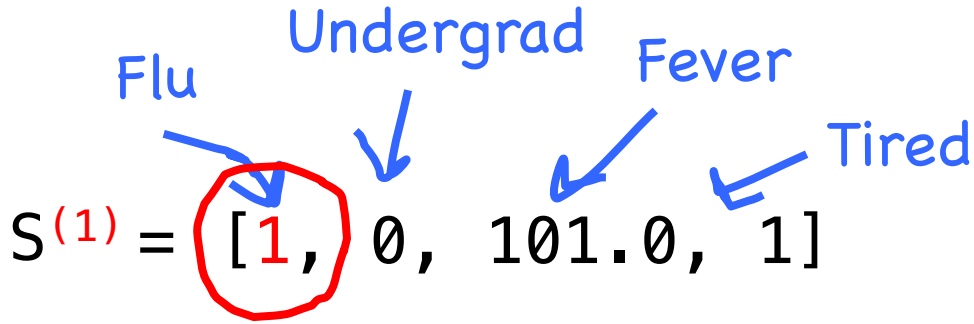


$$P(Flu = 1 | \text{All others})$$
$$= P(Flu = 1 | Und = 0, Fev = 98.3, Tir = 1)$$
$$= 0.21$$

$$Flu_1 = \text{Sample}\Big[P(Flu = 1 | \text{All others})\Big]$$

Alg #2: MCMC

All Samples = [$S^{(0)}$]

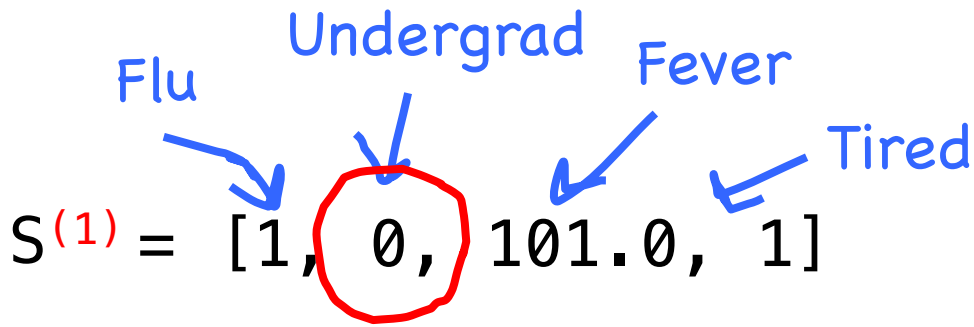


$$P(Flu = 1 | \text{All others})$$
$$= P(Flu = 1 | Und = 0, Fev = 98.3, Tir = 1)$$
$$= 0.21$$

$$Flu_1 = \text{Sample}\Big[P(Flu = 1 | \text{All others})\Big]$$

Alg #2: MCMC

All Samples = [$S^{(0)}$]

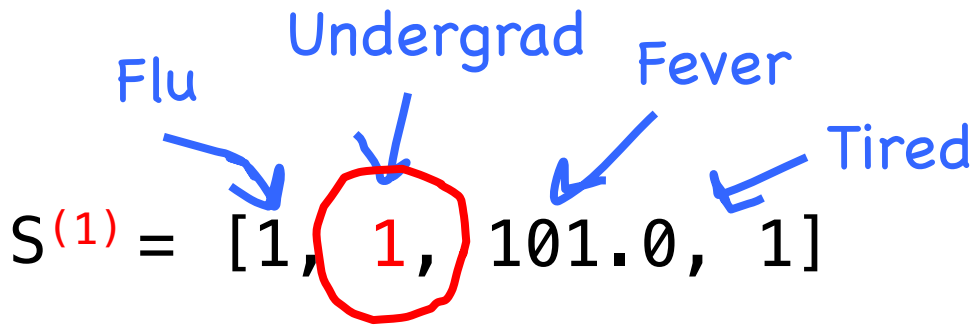


$$P(Und = 1 | \text{All others})$$
$$= P(Und = 1 | Flu = 1, Fev = 98.3, Tir = 1)$$
$$= 0.91$$

$$Und_1 = \text{Sample}\Big[P(Und = 1 | \text{All others})\Big]$$

Alg #2: MCMC

All Samples = [$S^{(0)}$]

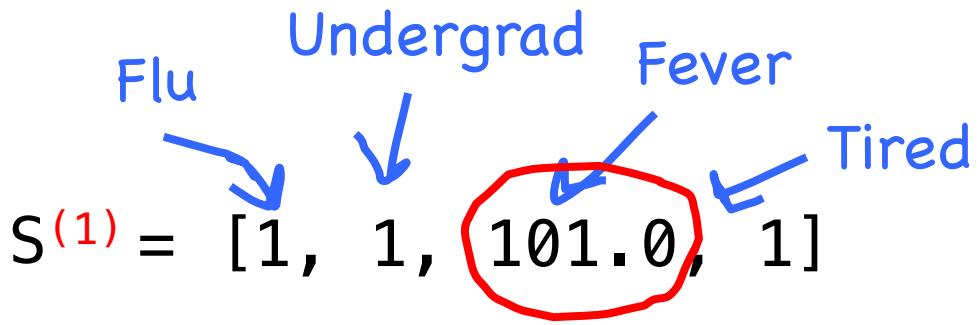


$$P(Und = 1 | \text{All others})$$
$$= P(Und = 1 | Flu = 1, Fev = 98.3, Tir = 1)$$
$$= 0.91$$

$$Und_1 = \text{Sample}\Big[P(Und = 1 | \text{All others})\Big]$$

Alg #2: MCMC

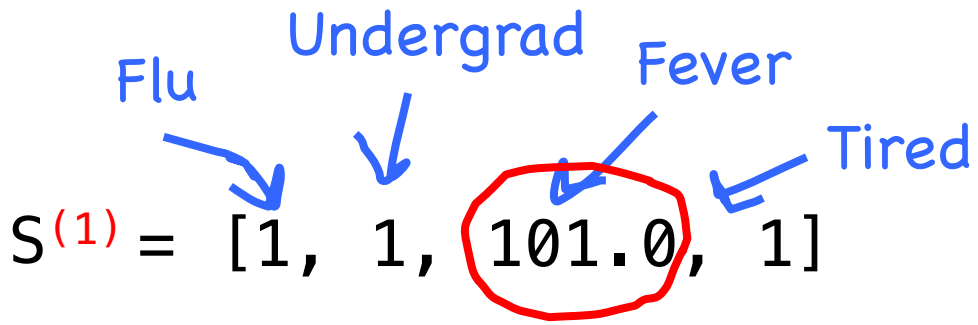
All Samples = [$S^{(0)}$]



Let's say you are conditioning on fever being 101.0...
then don't change that value

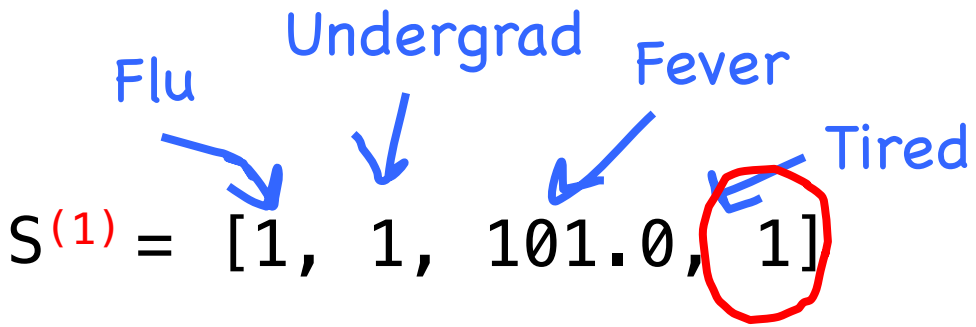
Alg #2: MCMC

All Samples = [$S^{(0)}$]



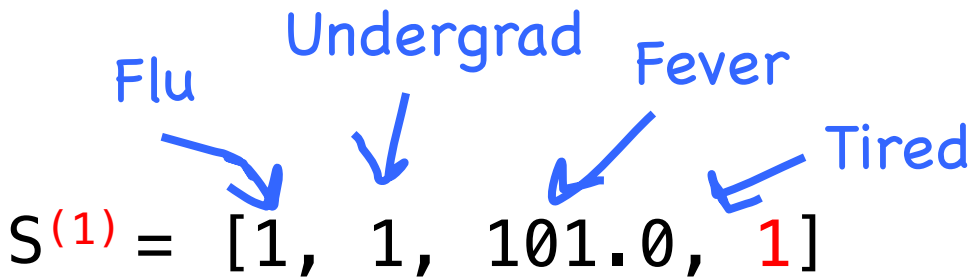
Alg #2: MCMC

All Samples = [$S^{(0)}$]



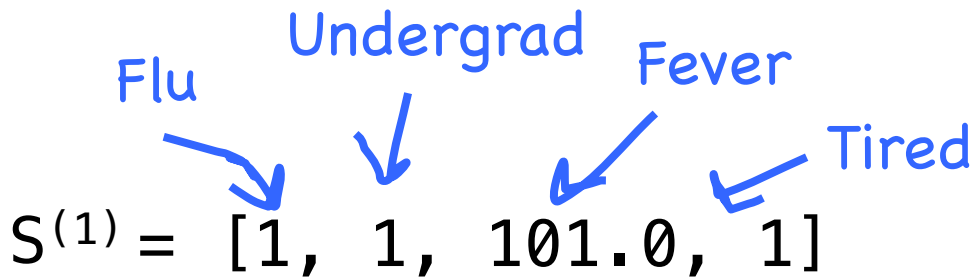
Alg #2: MCMC

All Samples = [$S^{(0)}$]



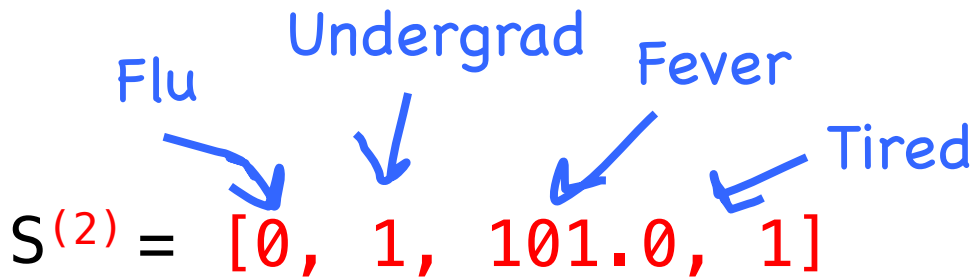
Alg #2: MCMC

All Samples = $[S^{(0)}, S^{(1)}]$



Alg #2: MCMC

$$\text{All Samples} = [S^{(0)}, S^{(1)}]$$



Alg #2: MCMC

All Samples = $[S^{(0)}, S^{(1)}, S^{(2)}]$

Repeat at least 10,000 times

Alg #2: MCMC

All Samples = [$S^{(0)}$, $S^{(1)}$, $S^{(2)}$, … , $S^{(10000)}$]

Repeat at least 10,000 times

Alg #2: MCMC

```
[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 joint sample:

[Flu, Undergrad, Fever, Tired]

Alg #2: MCMC

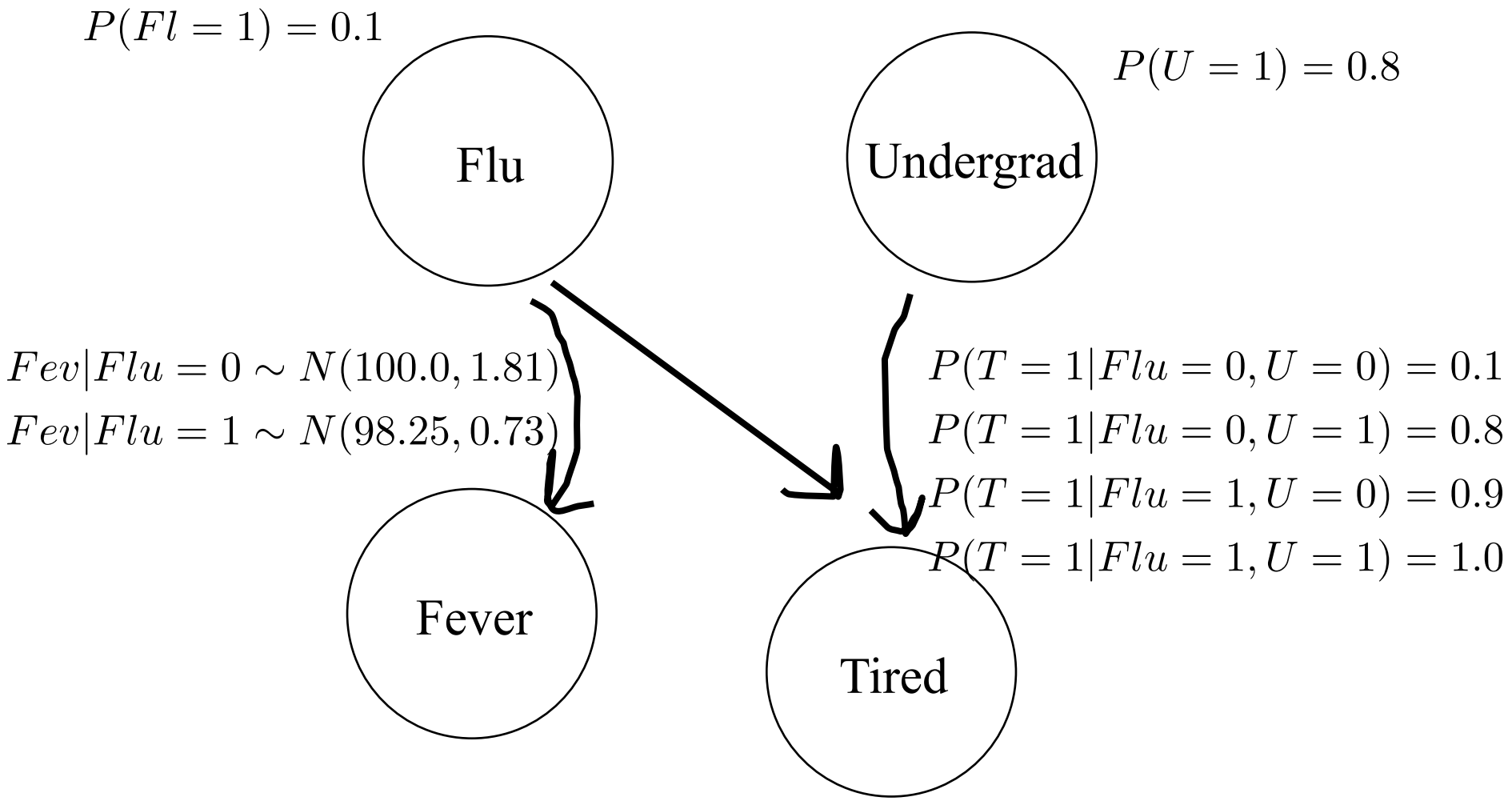
For each random variable you must specify a way to sample from

Current random variable

$$P(X_j^{(t+1)} | X_1^{(t+1)}, \ldots, X_{j-1}^{(t+1)}, X_{j+1}^{(t)} \ldots X_n^{(t)})$$

All the other values in the sample

Probabilistic Model




The Magic
School Bus

Alg #1: Joint Sampling

With enough samples:

- Probability estimates will be correct
- Conditional probability estimates will be correct
- Expectation estimations will be correct