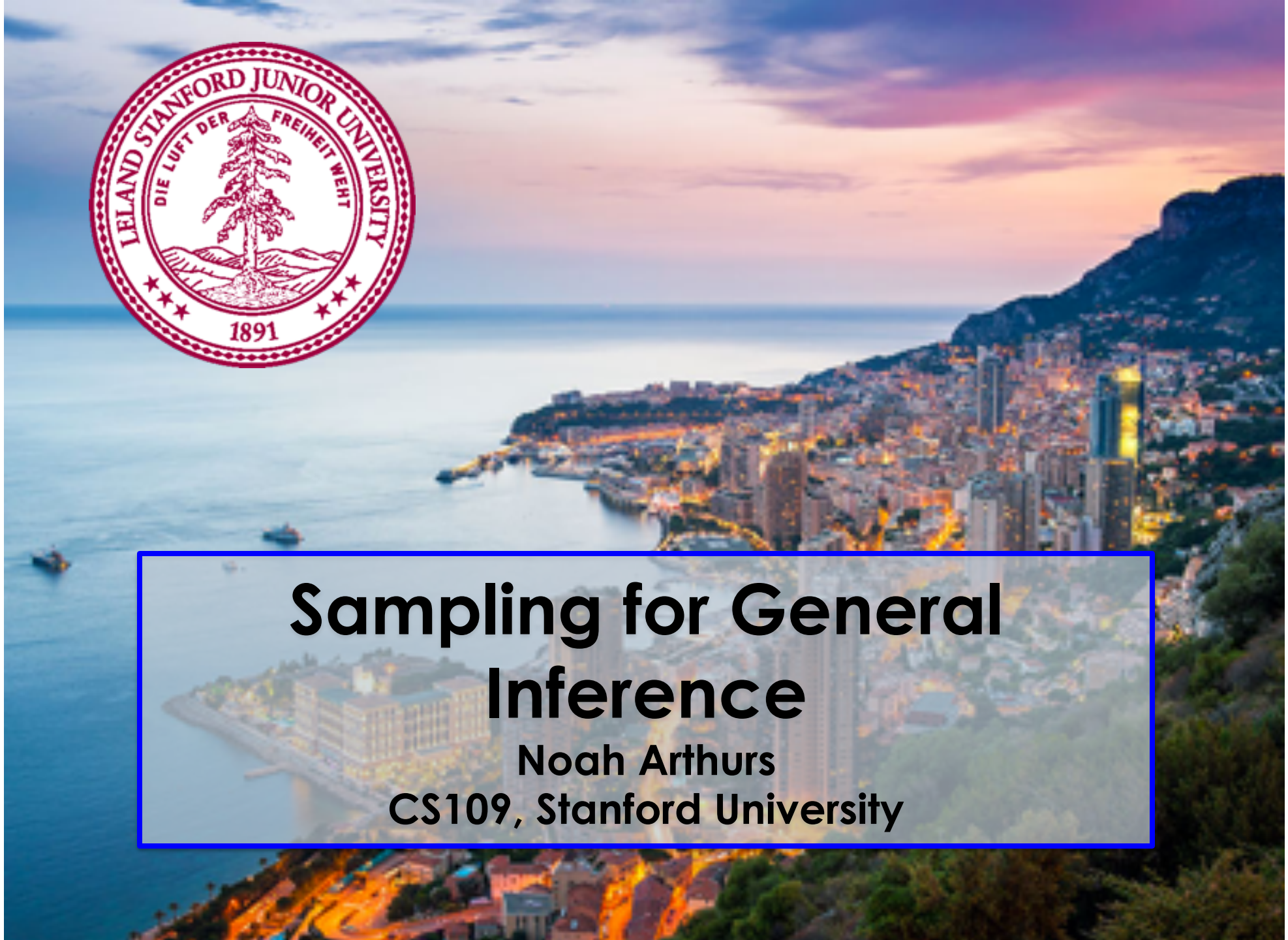
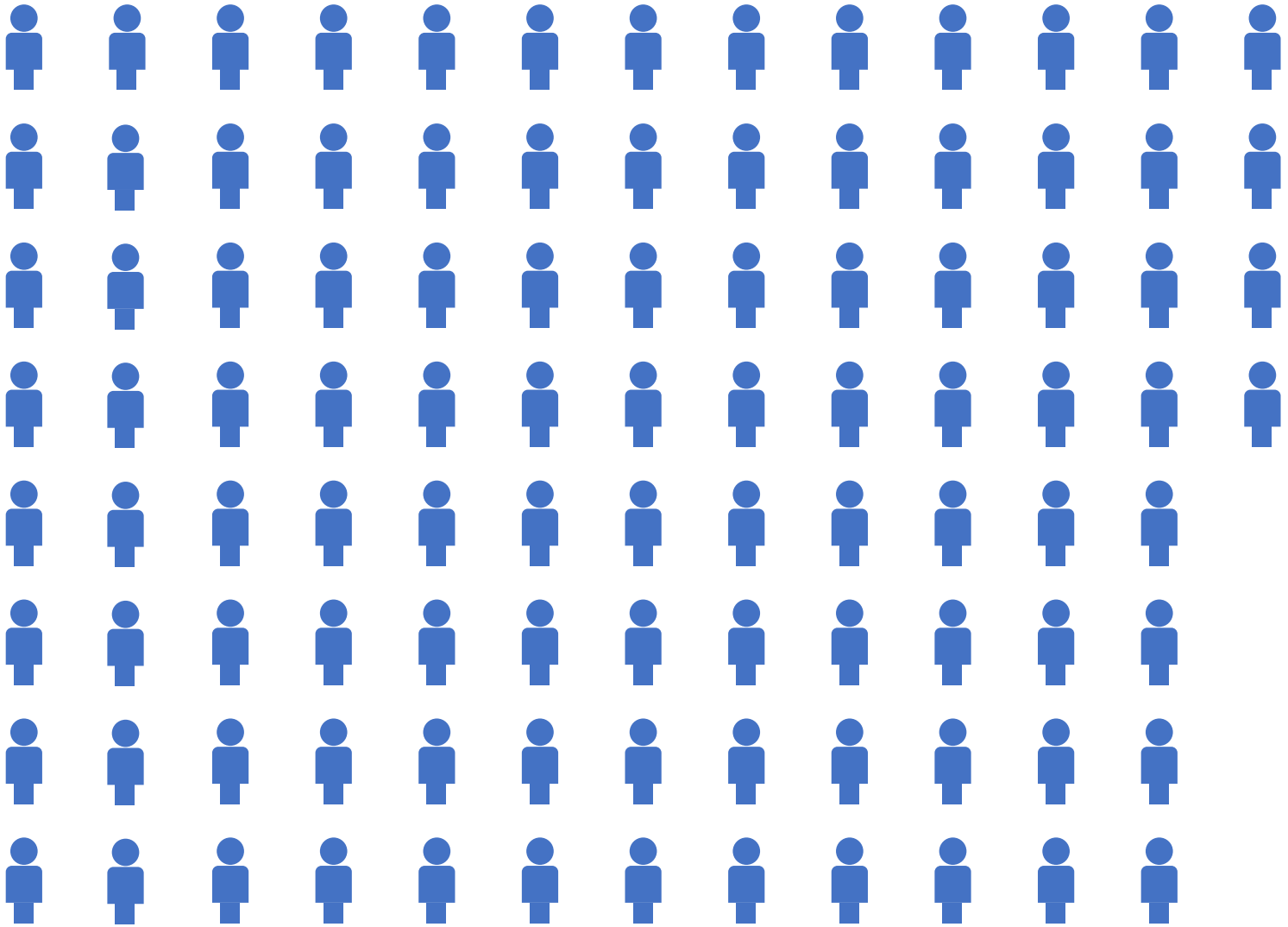


Sampling for General Inference

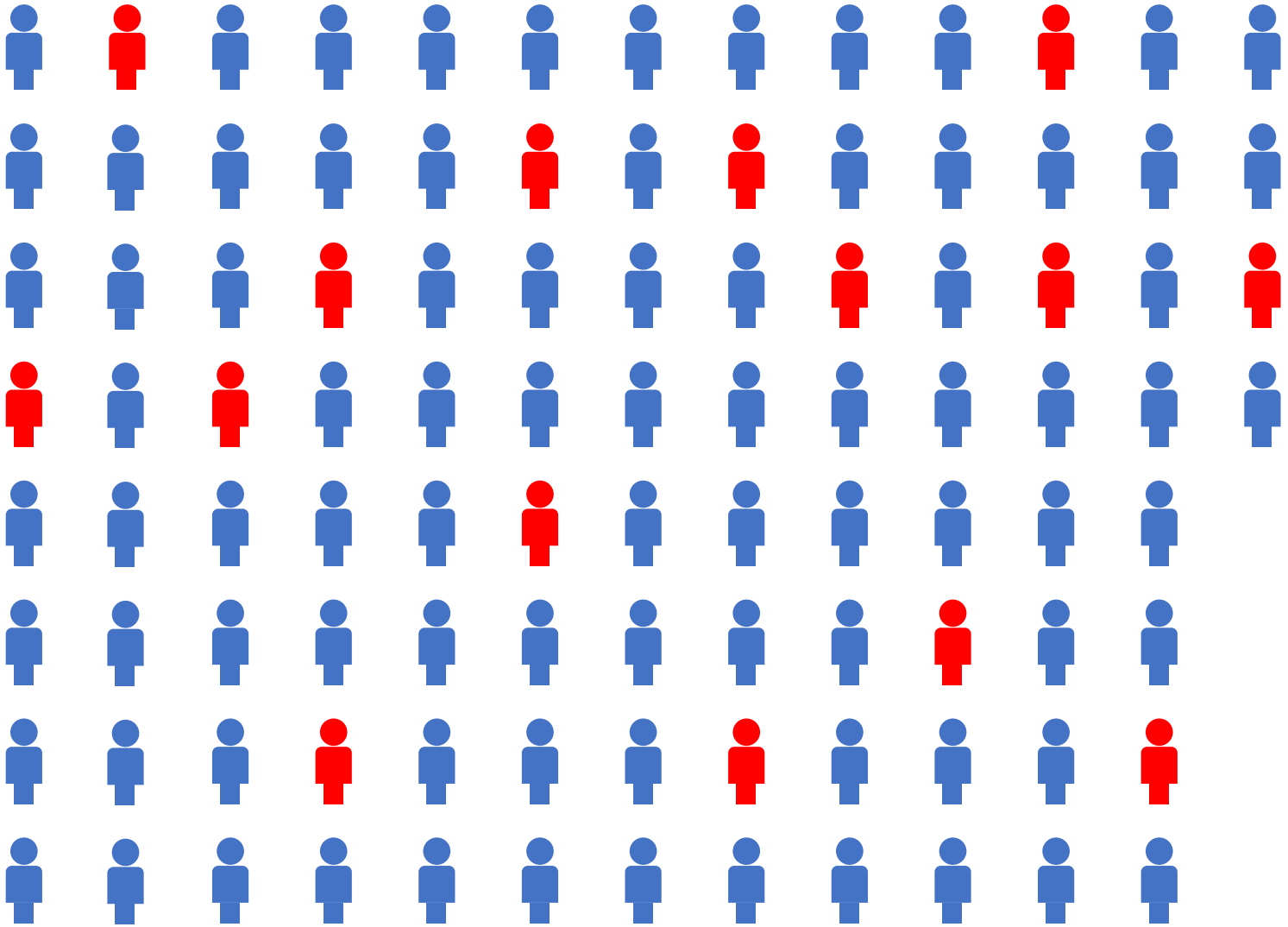
Noah Arthurs
CS109, Stanford University

Review

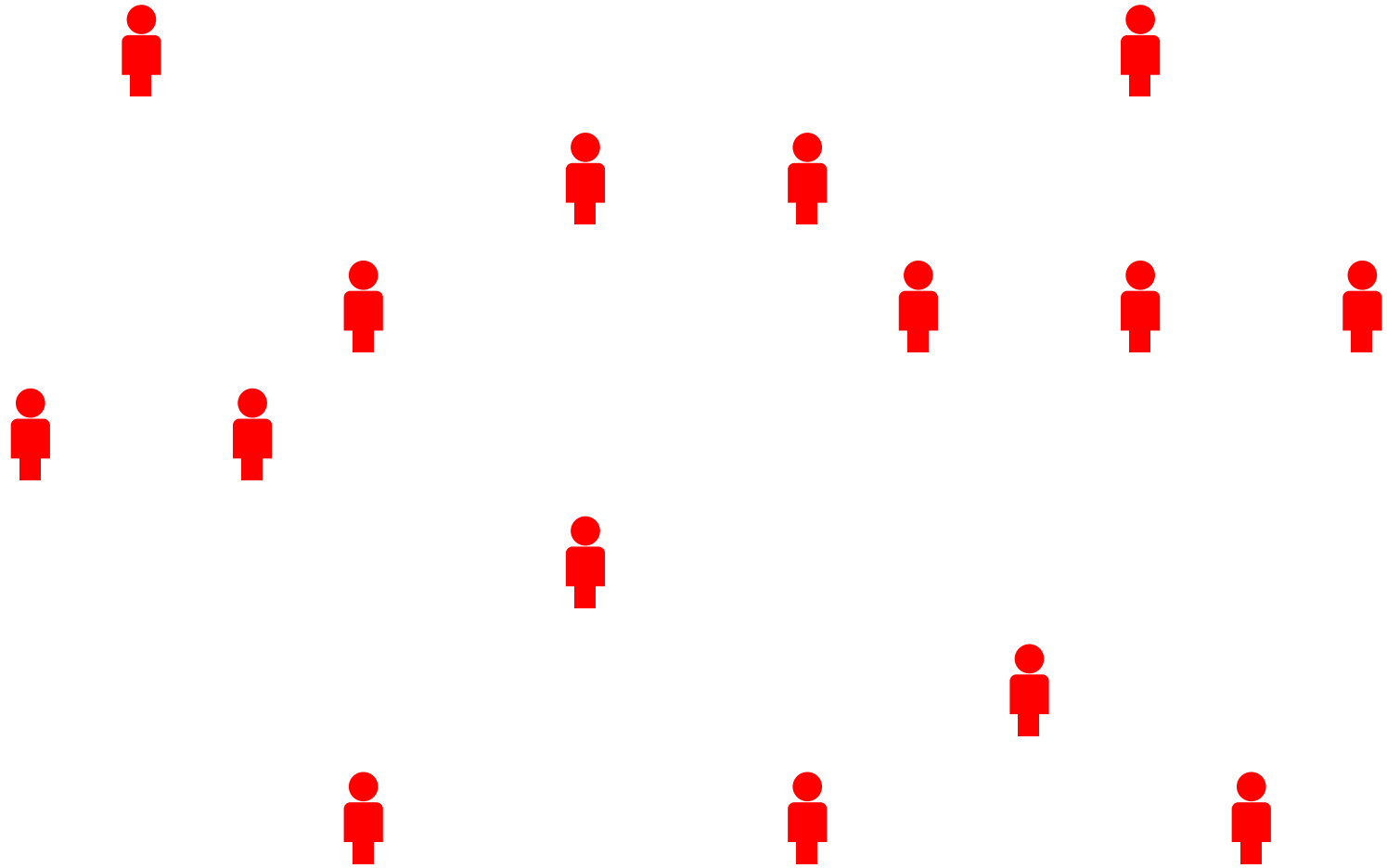
Population



Sample

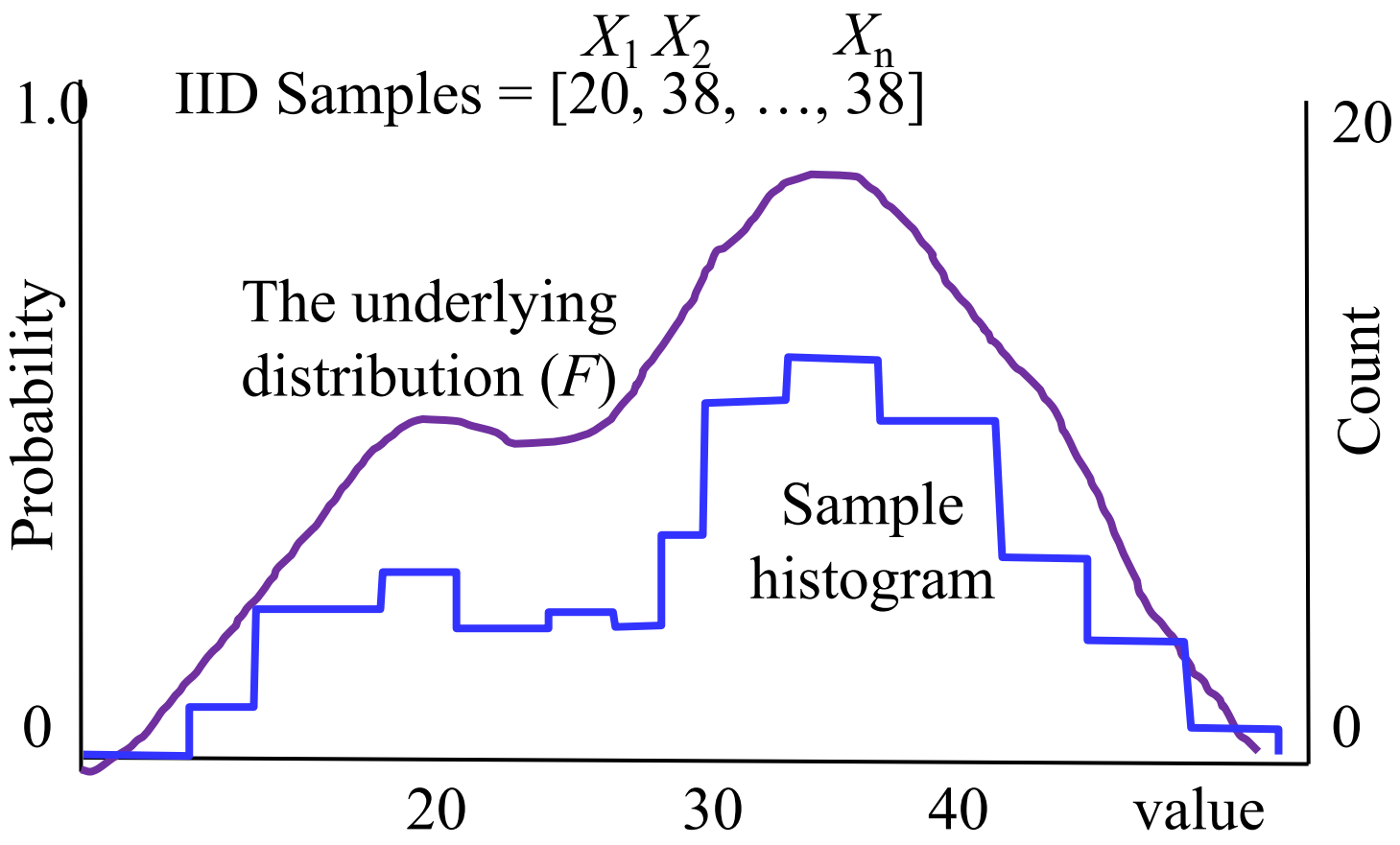


Sample

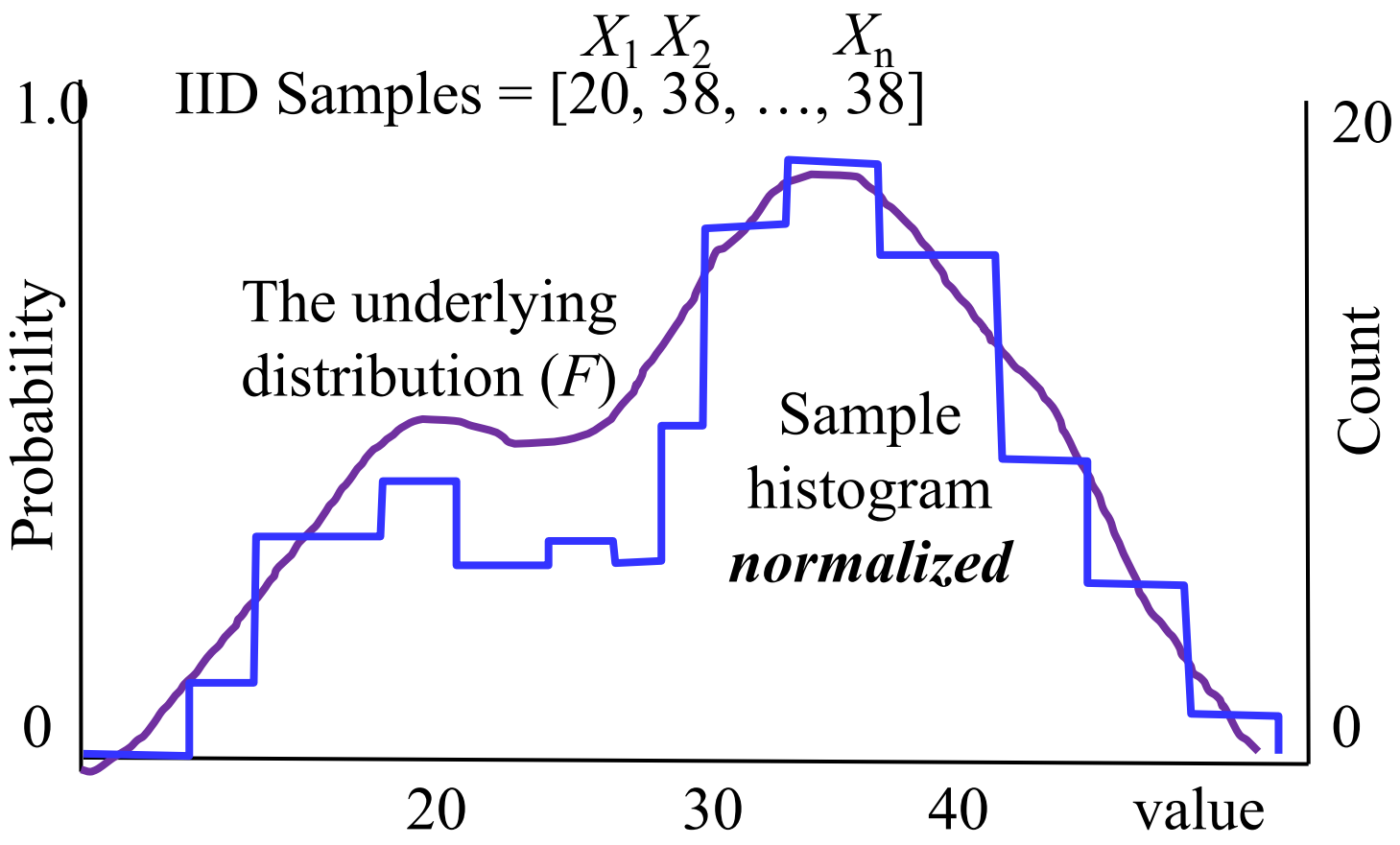


Collect one (or more) numbers from each person

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

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

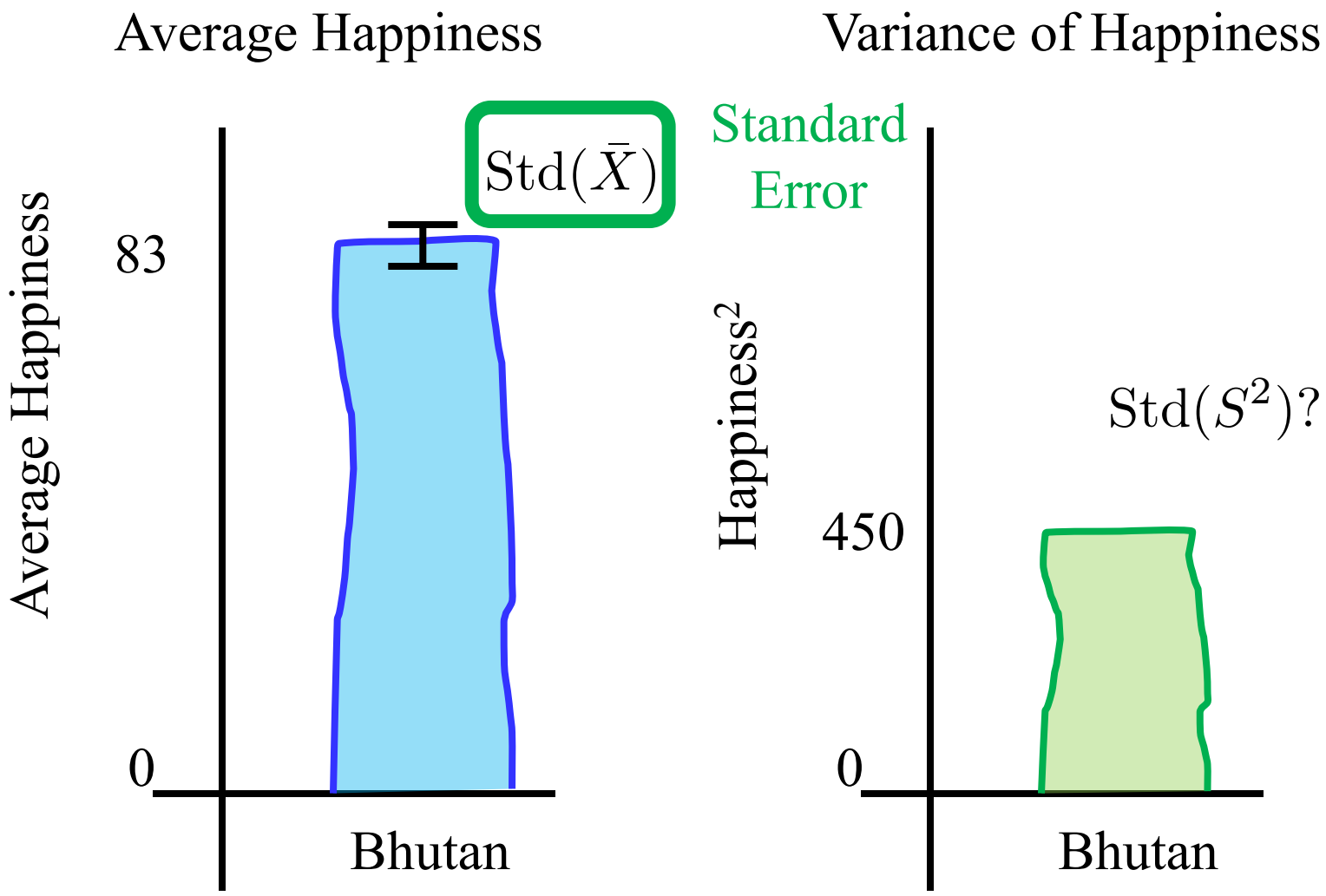
These can be thought of as random variables :O

Var of Sample Mean

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

Square of Standard Error

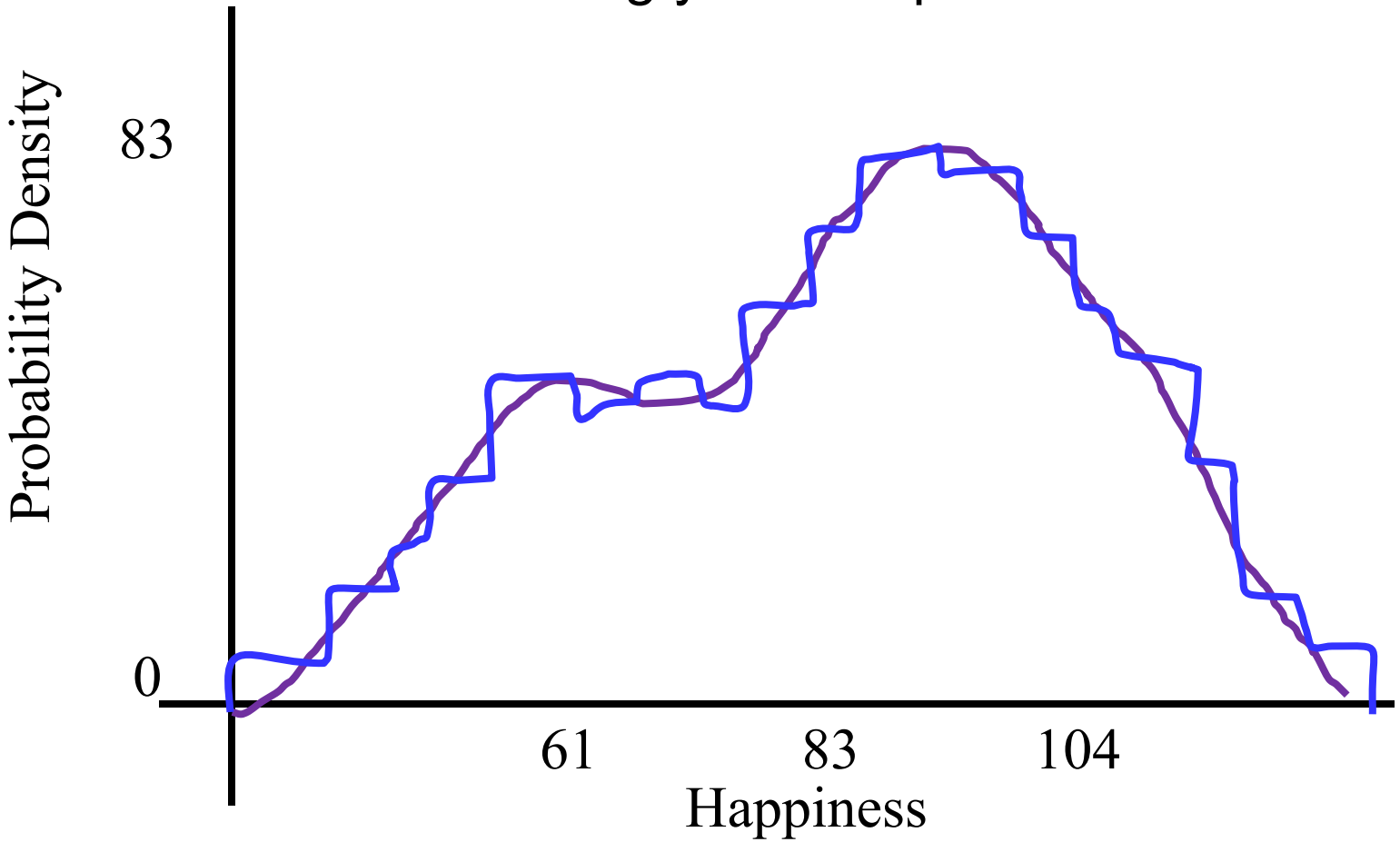
Sample Mean



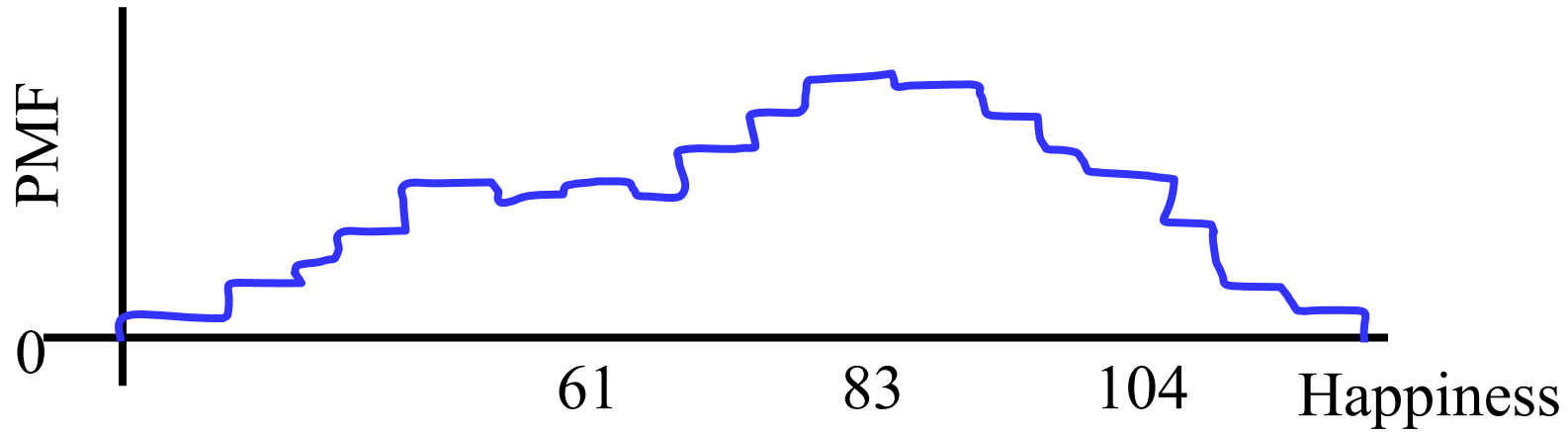
Claim: The average happiness of Bhutan is 83 ± 2

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**

A **non-parametric** continuous distribution can be represented in a **computer** as a **list** of numbers sampled from the distribution

$$\text{Vars} = [472.7, 478.4, 469.2, \ldots, 476.2]$$

End Review

Bootstrapping 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
$\mu_1 = 3.1$	$\mu_2 = 2.4$

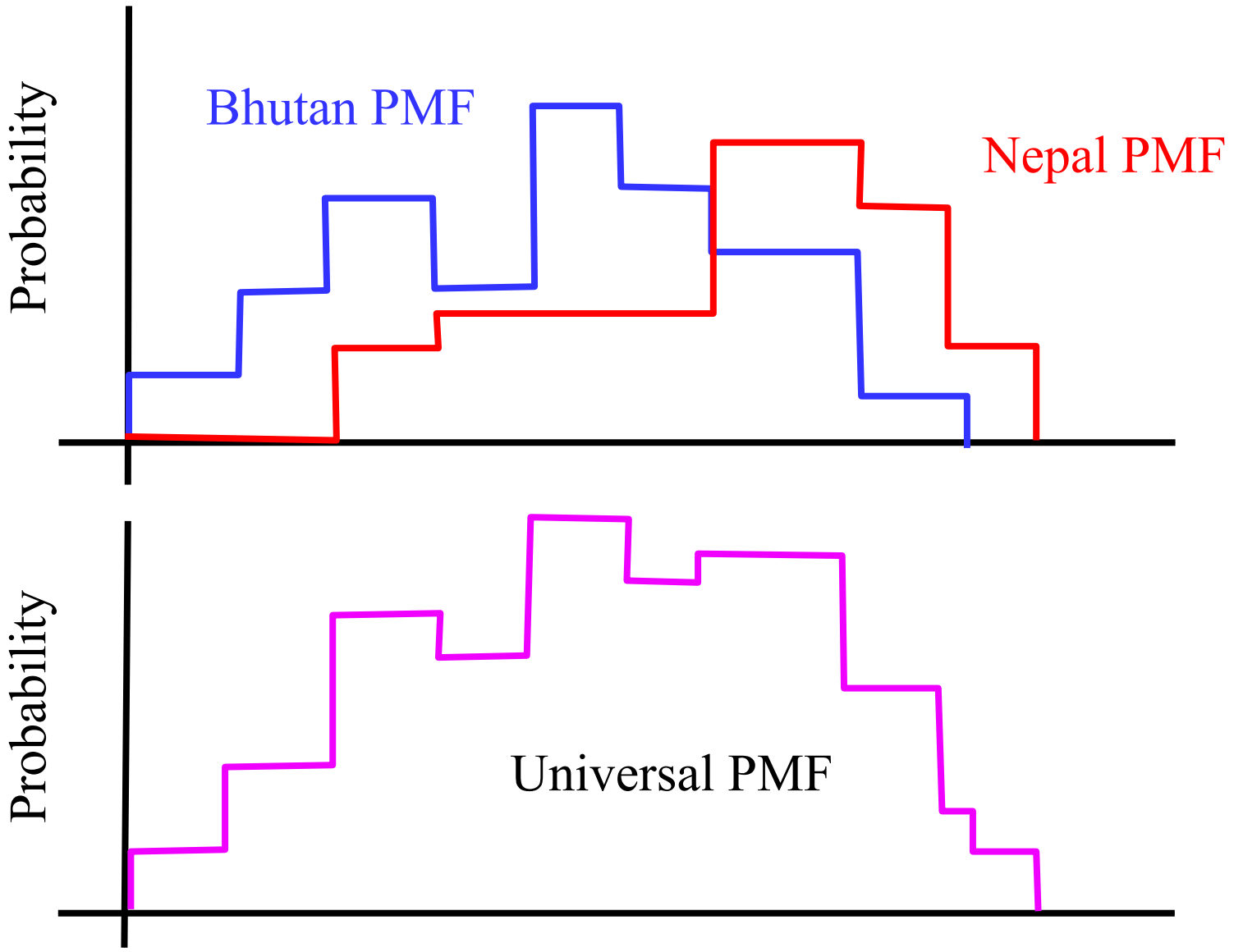
Null Hypothesis Test

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 claim 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!

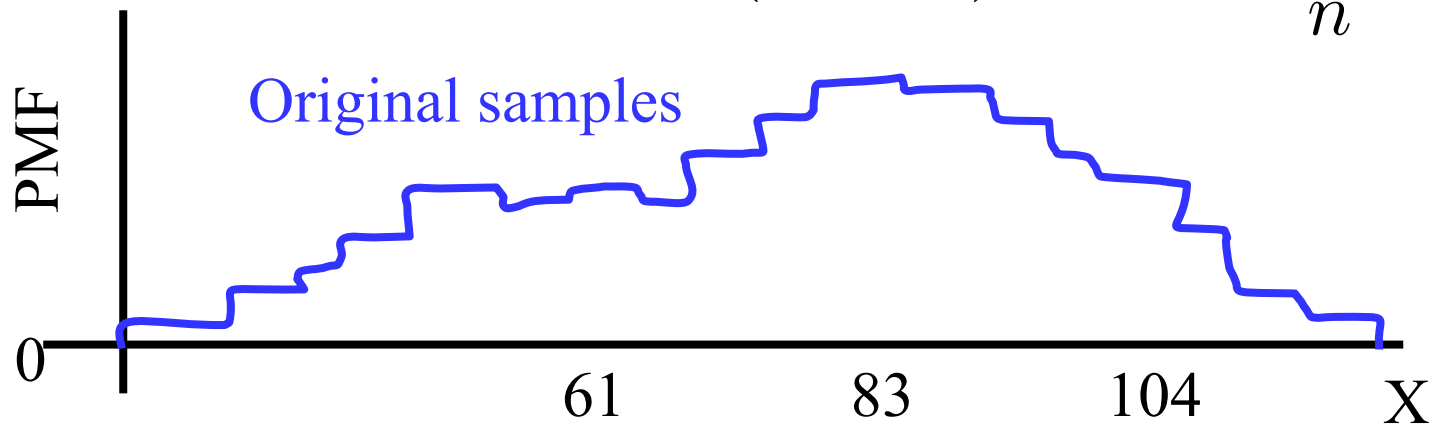
Algorithm in Practice

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

Algorithm in Practice

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

$$P(X = k) = \frac{\text{count}(X = k)}{n}$$



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 ($p < 0.008$).

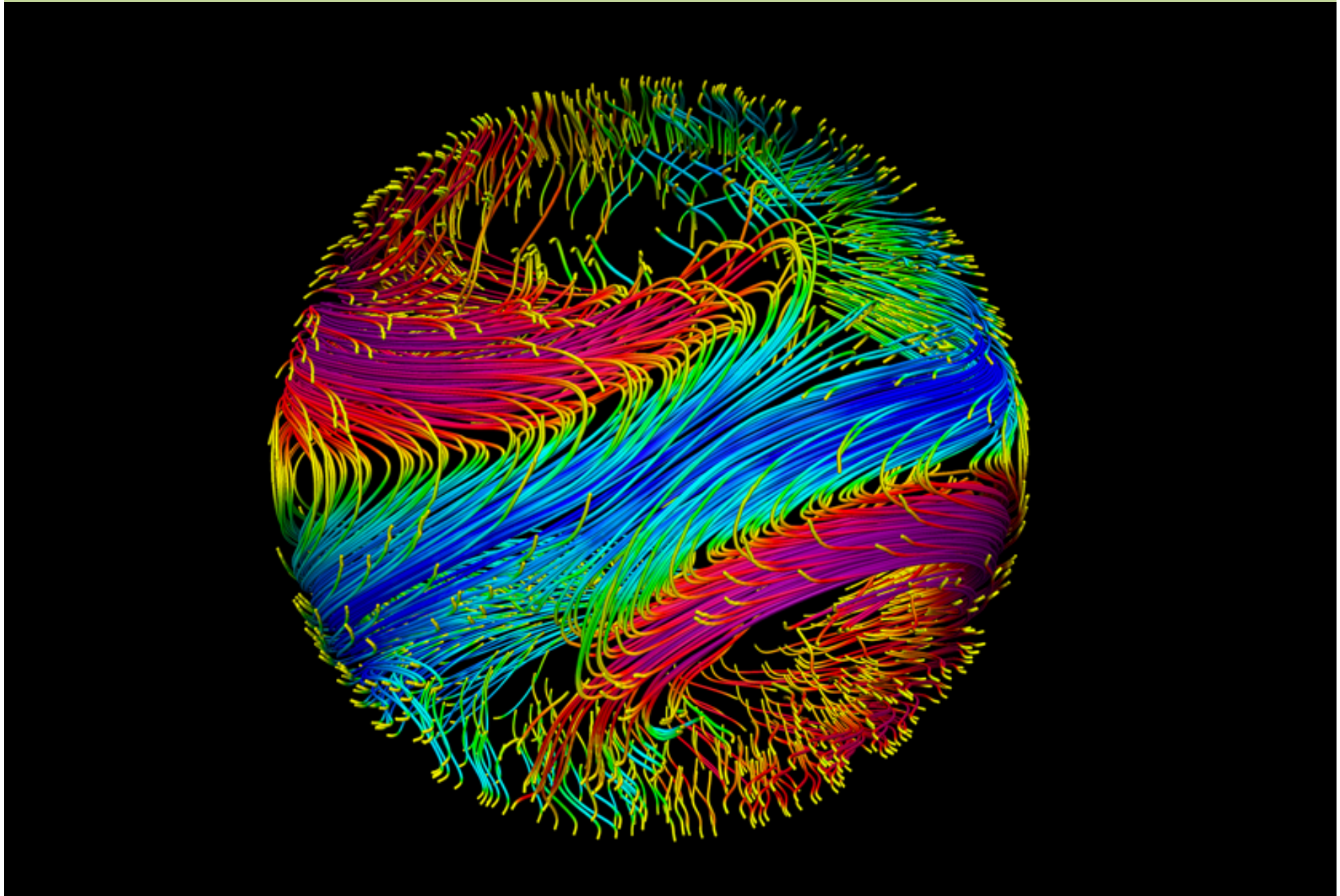
Null Hypothesis Test

Nepal Happiness
4.44
3.36
5.87
2.31
...
3.70
$\mu_1 = 3.1$

Bhutan Happiness
2.15
3.01
2.02
What about your confidence in this number?
$\mu_2 = 2.4$

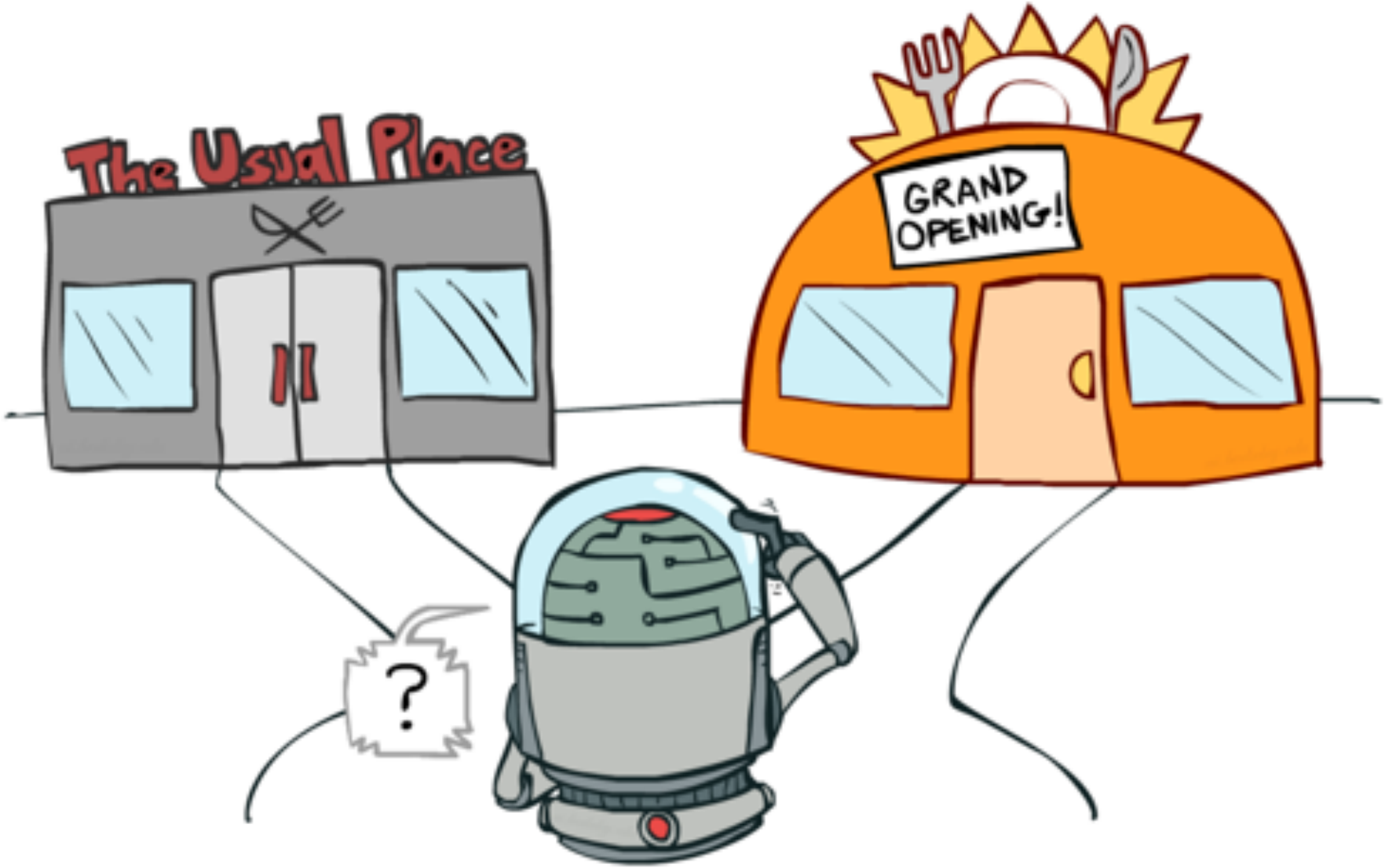
Claim: The difference in happiness between Nepal and Bhutan is 0.7 happiness points ($p < 0.008$).

Randomized Algorithms



Bootstrapping

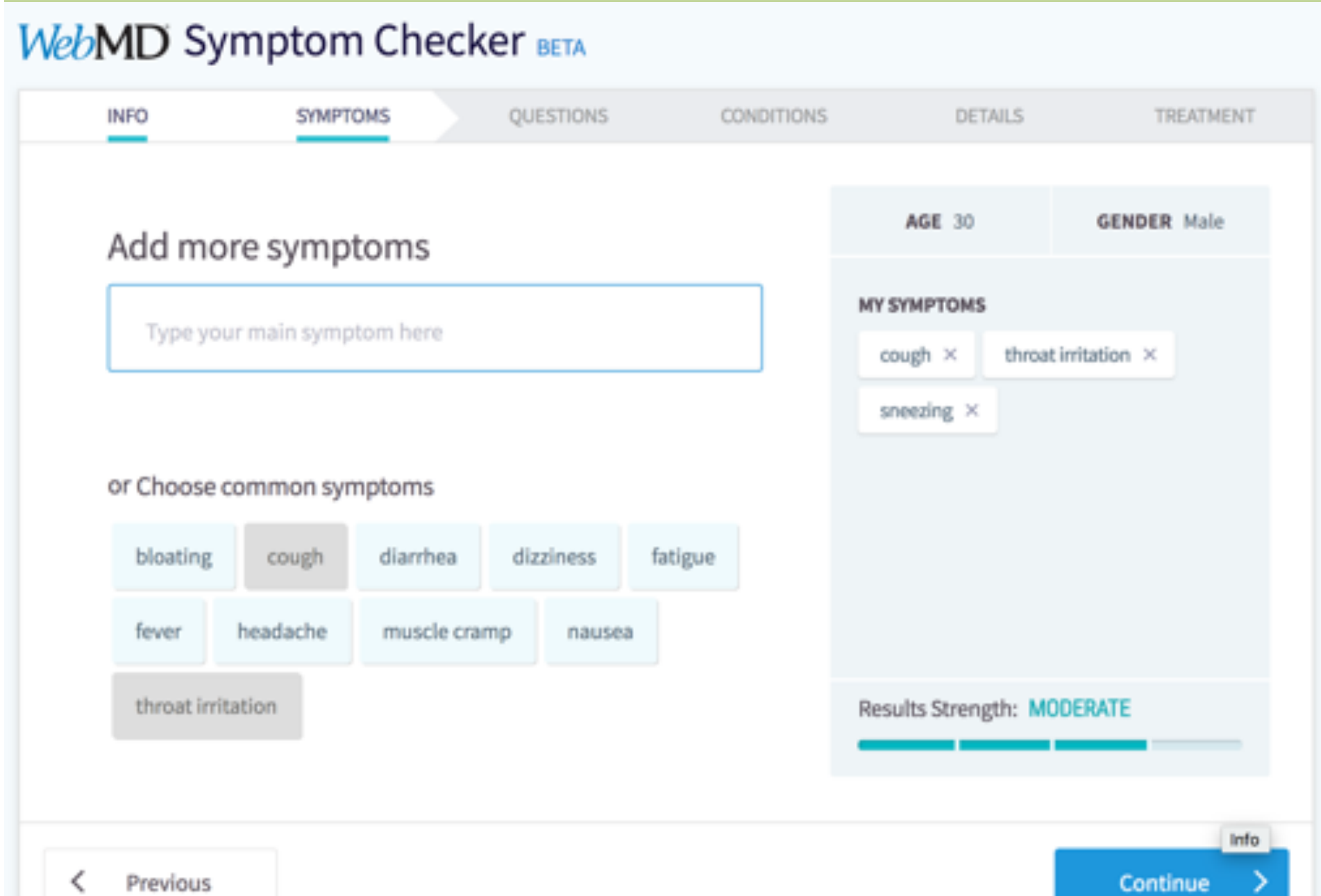
Thompson Sampling



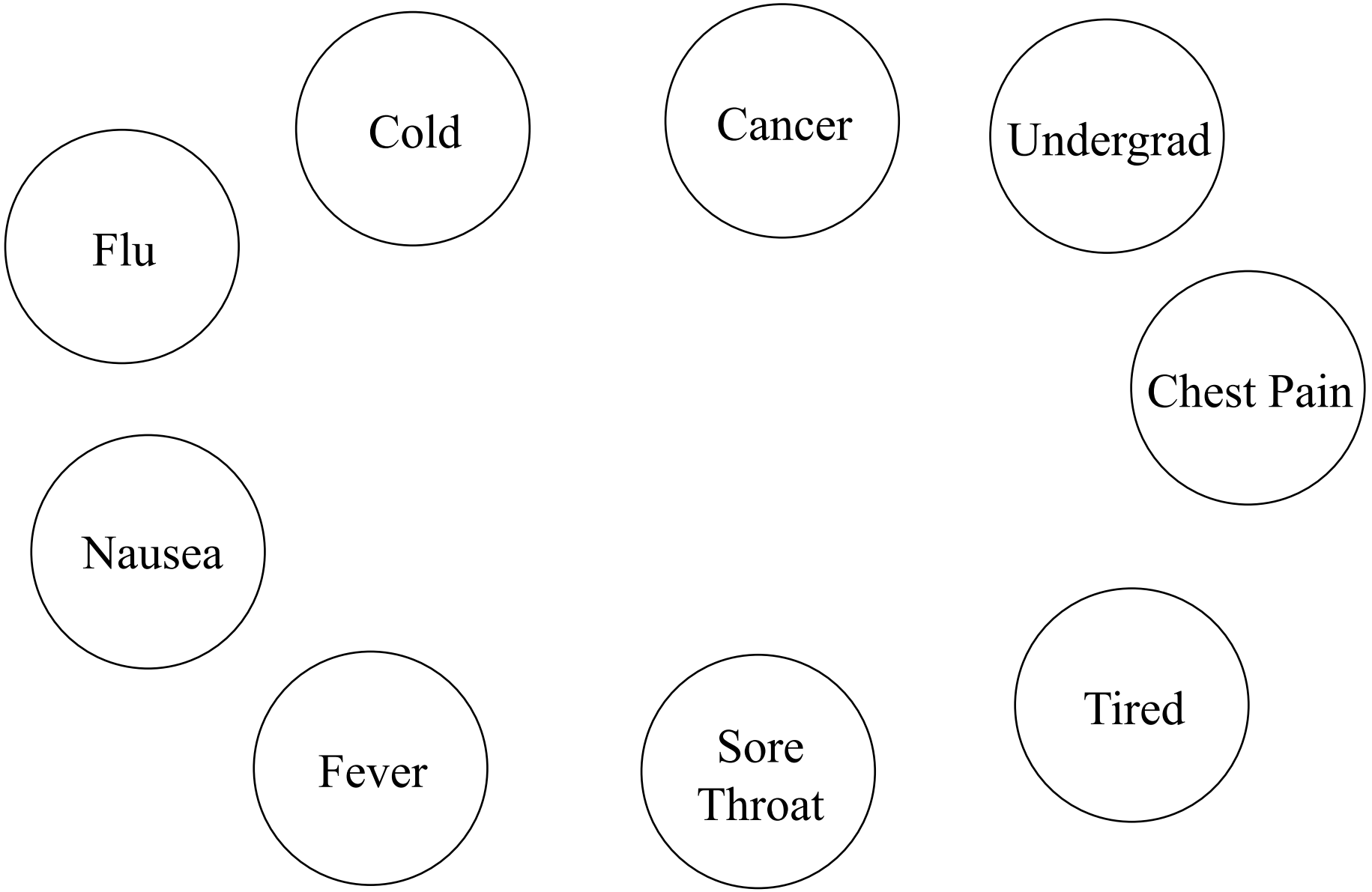
Something brand **new**…

General “Inference”

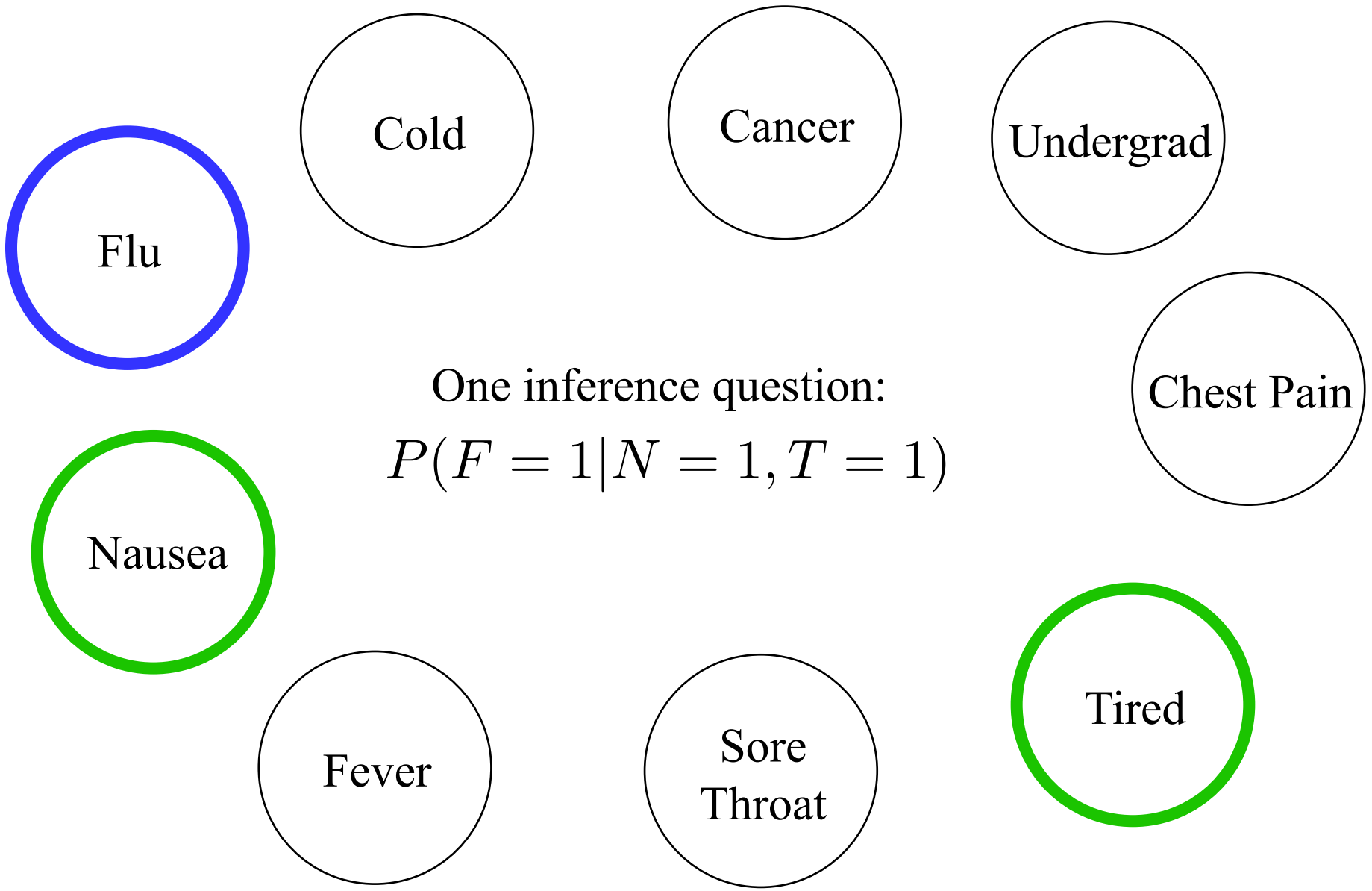
General “Inference”



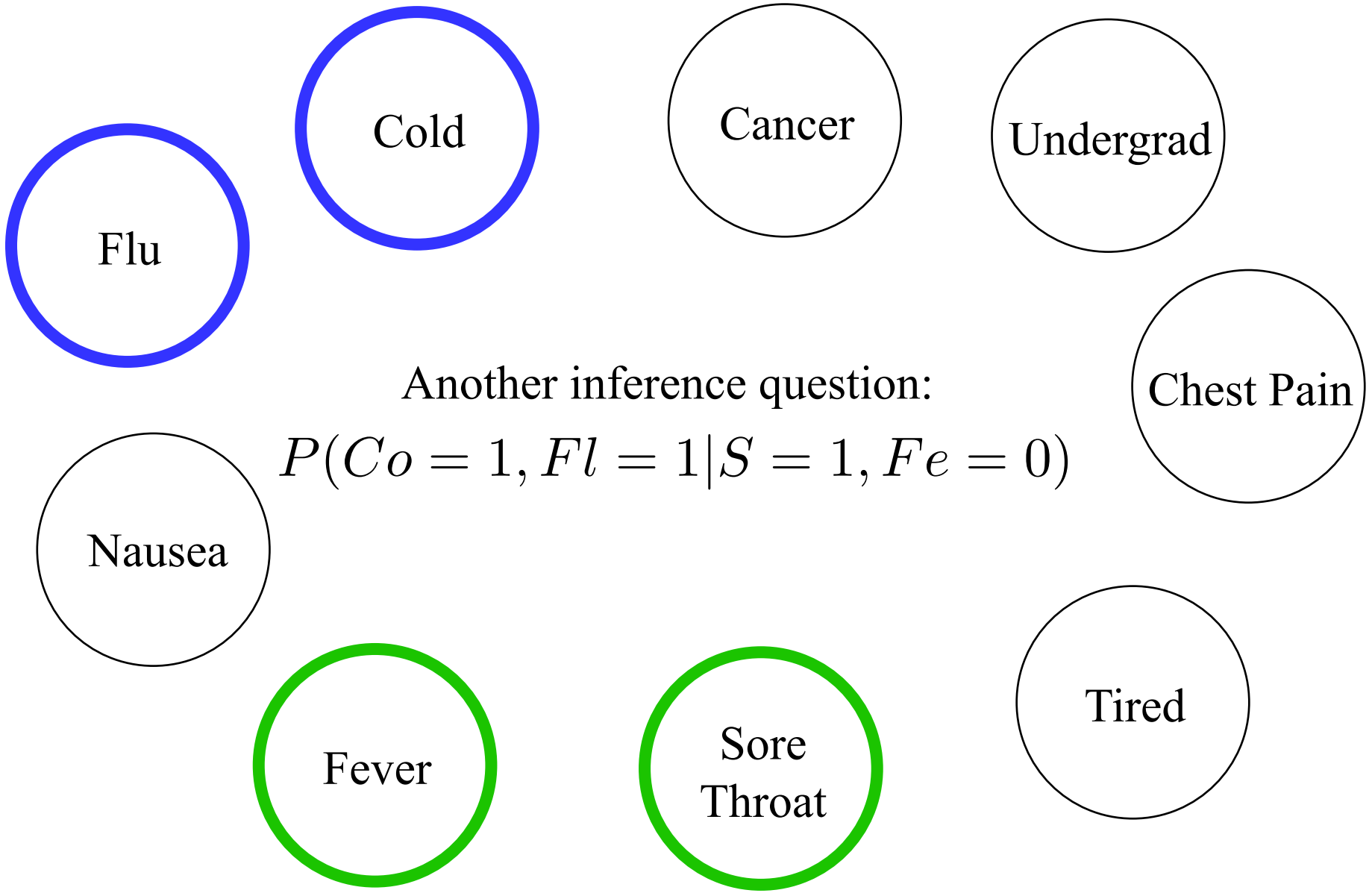
General "Inference"



General "Inference"



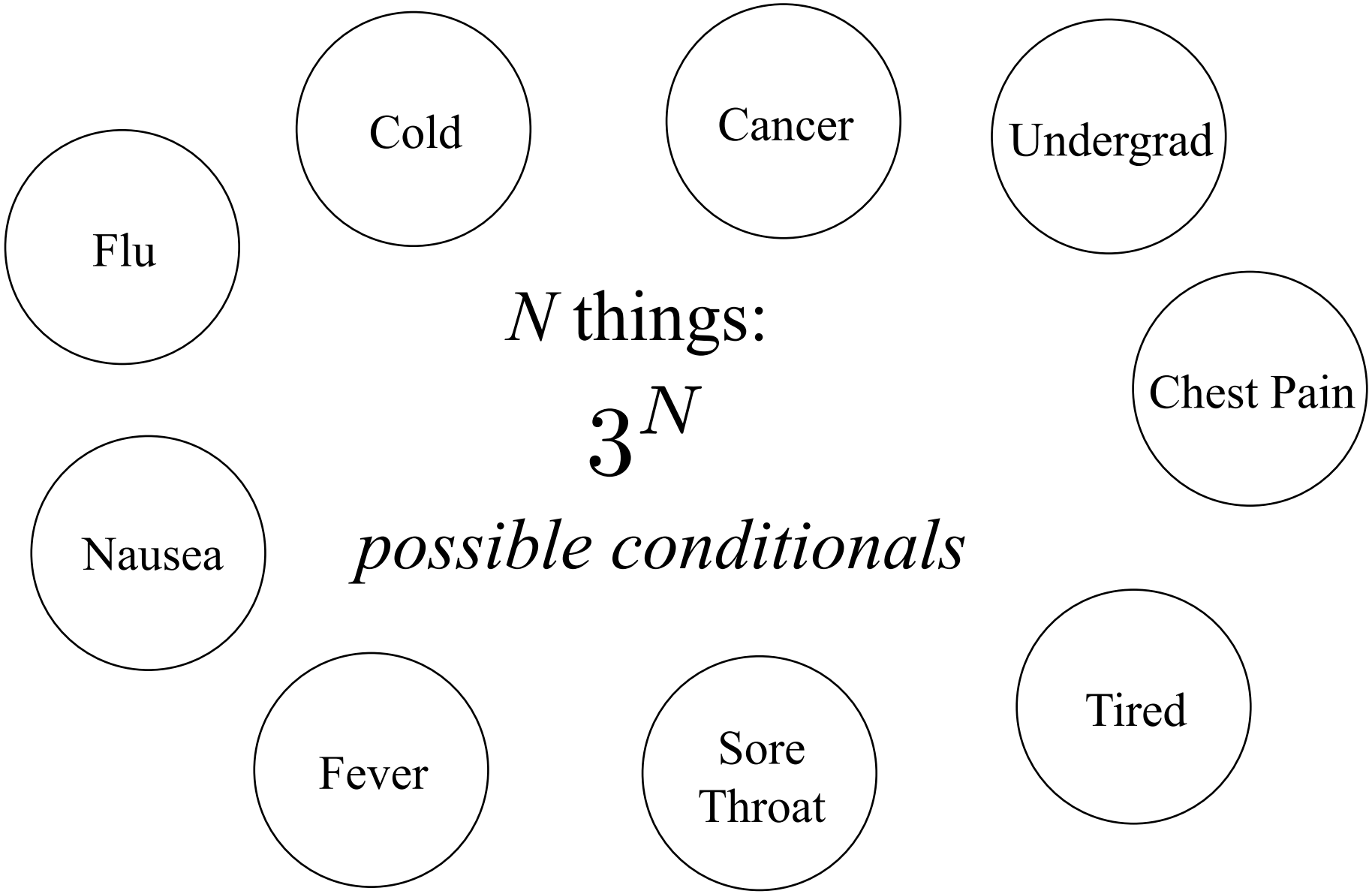
General "Inference"



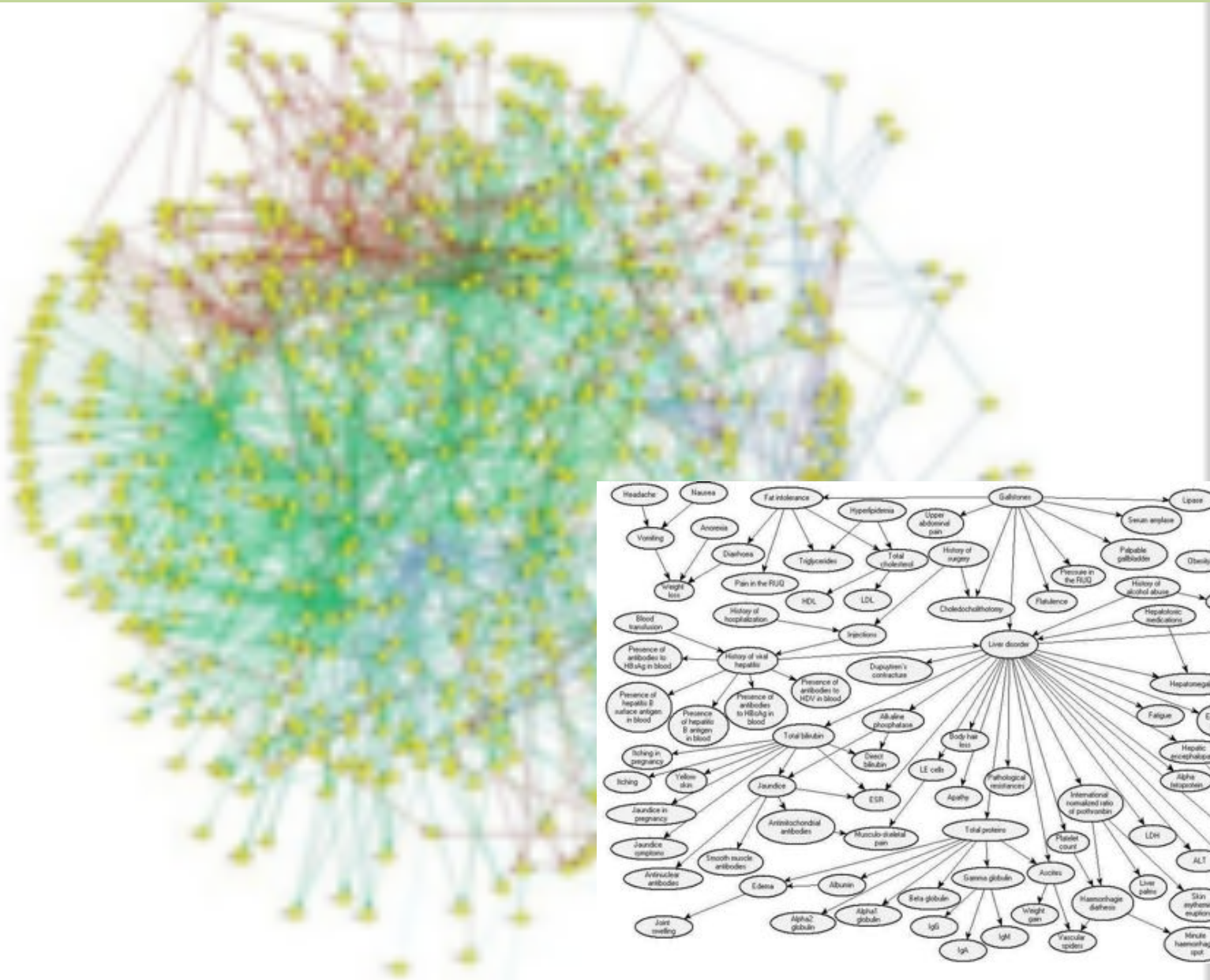
Another inference question:

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

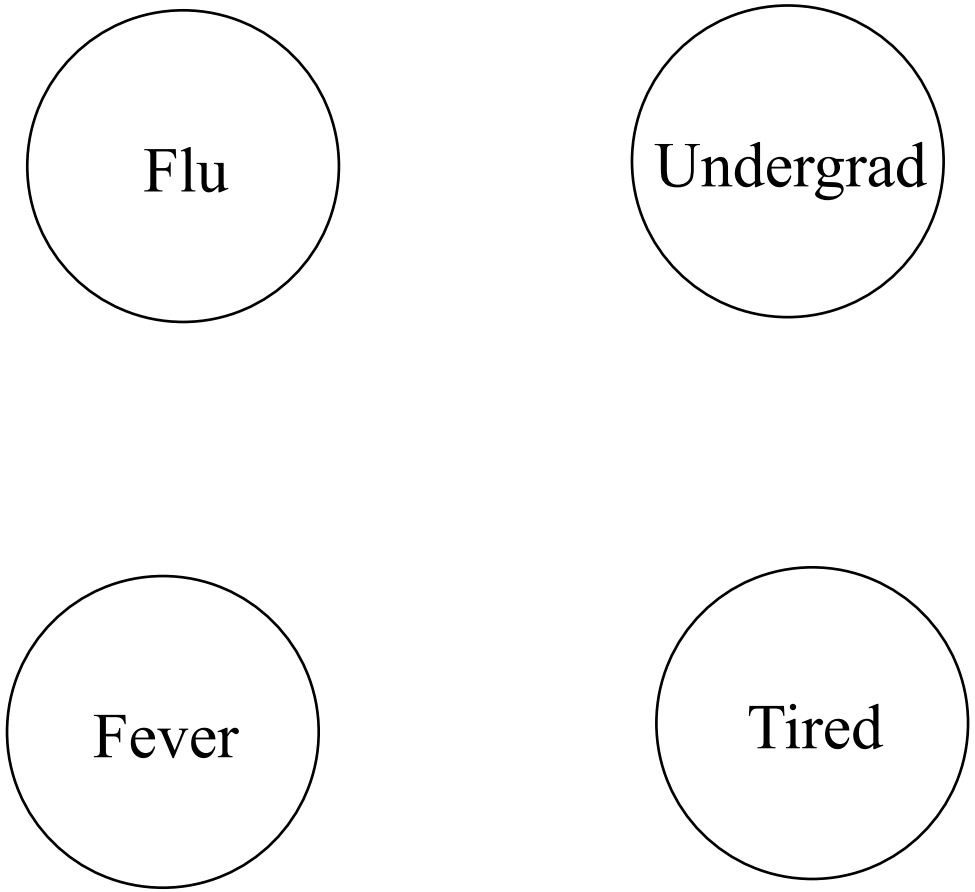
How Many Things Can You Condition On?



N is large…


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

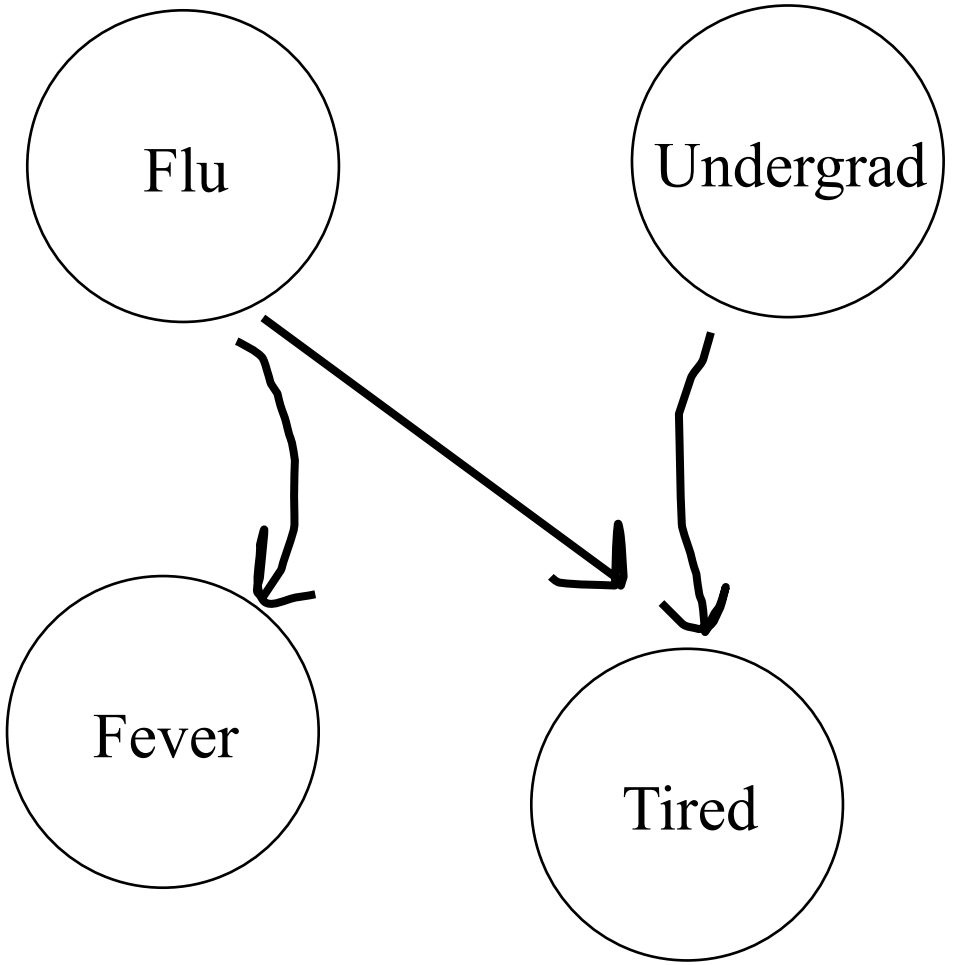
Simple WebMd



Naively specifying a joint is often impossible…

Probabilistic Model

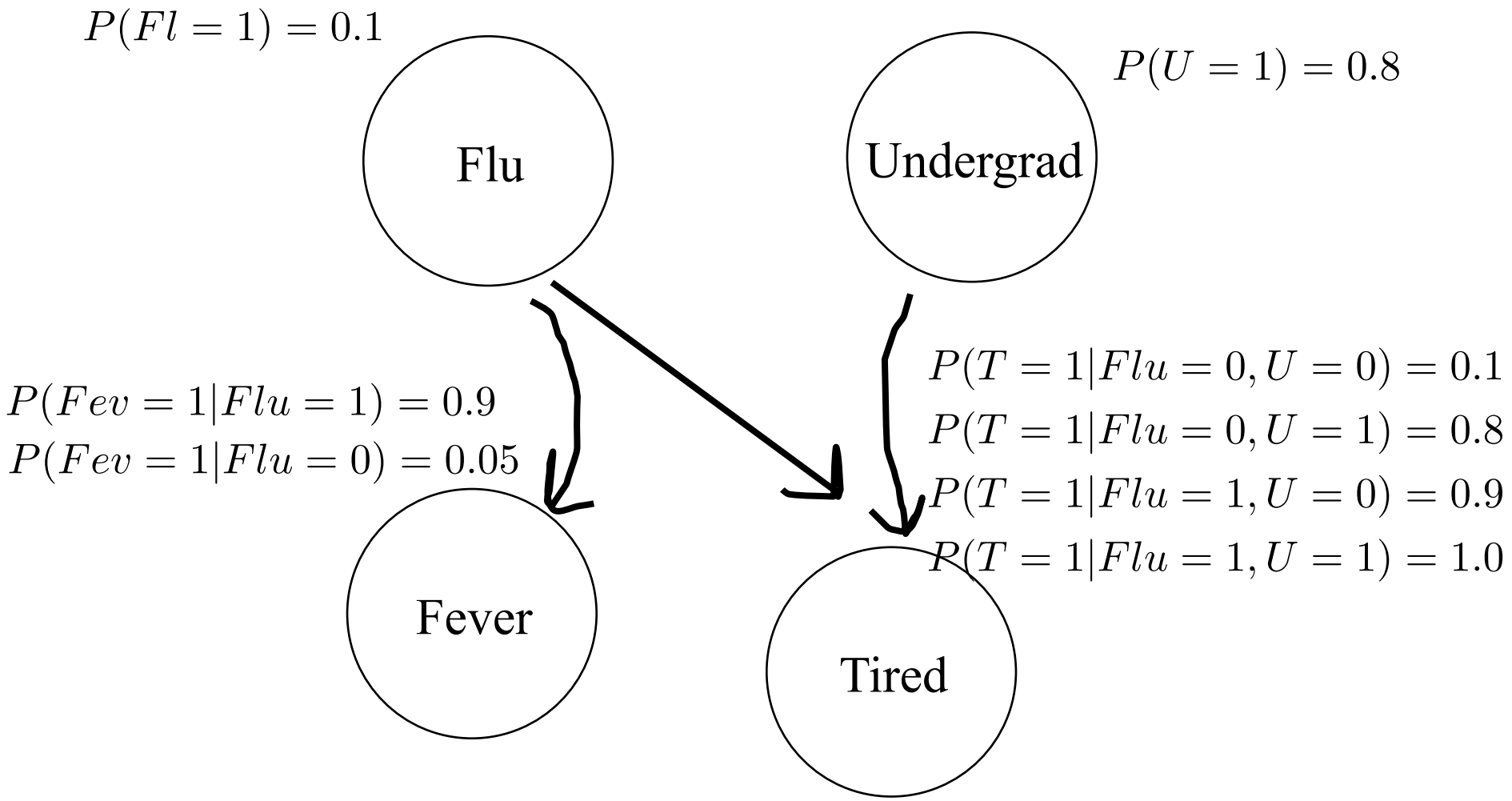
Describe the joint using causality!



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

Probabilistic Model

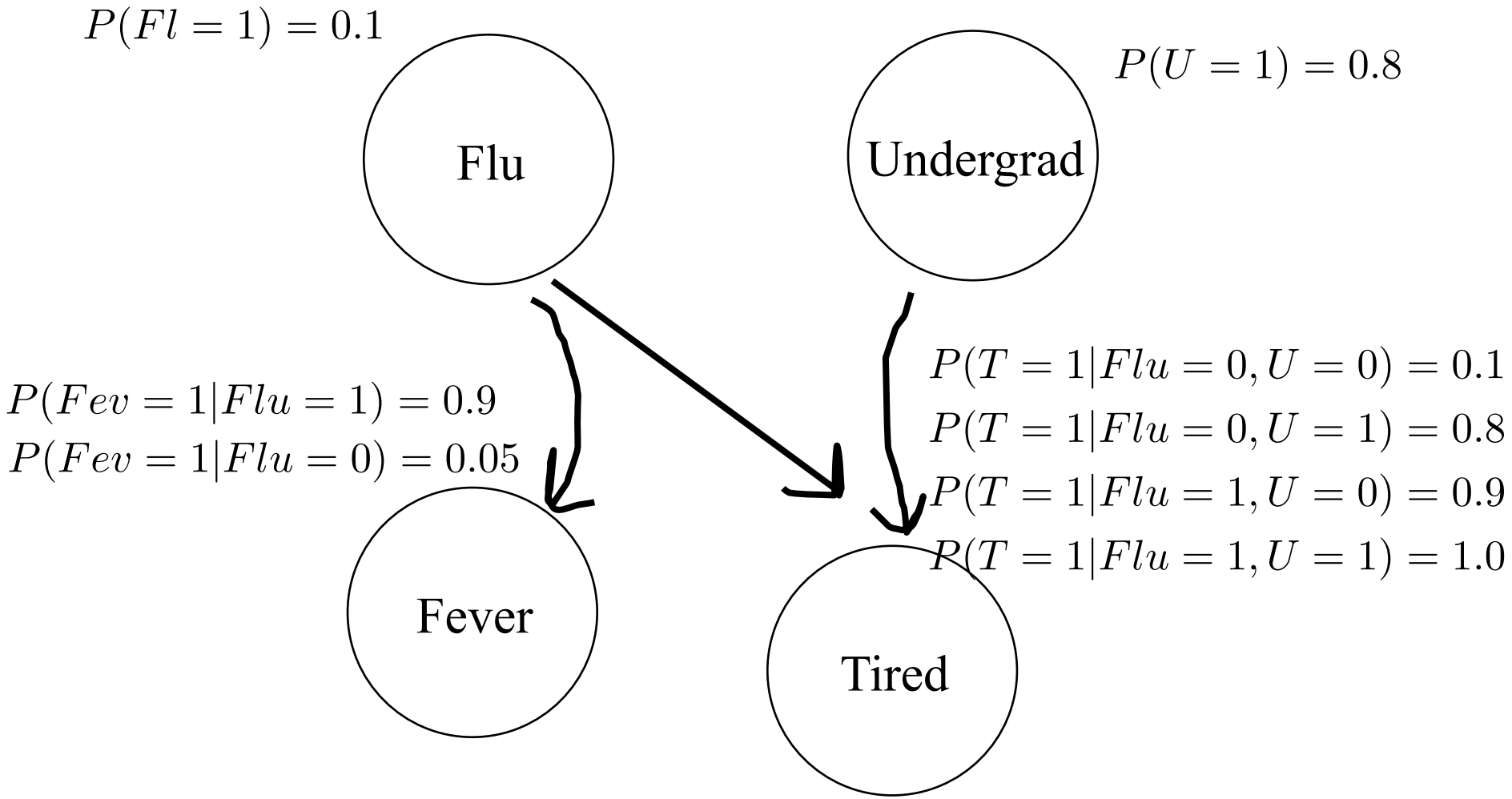
Describe the joint using causality!



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

Bayesian Network

Describe the joint using causality!



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

Bayesian Network:

If you know causality,

Make a network of assumed direct causality for your random vars.

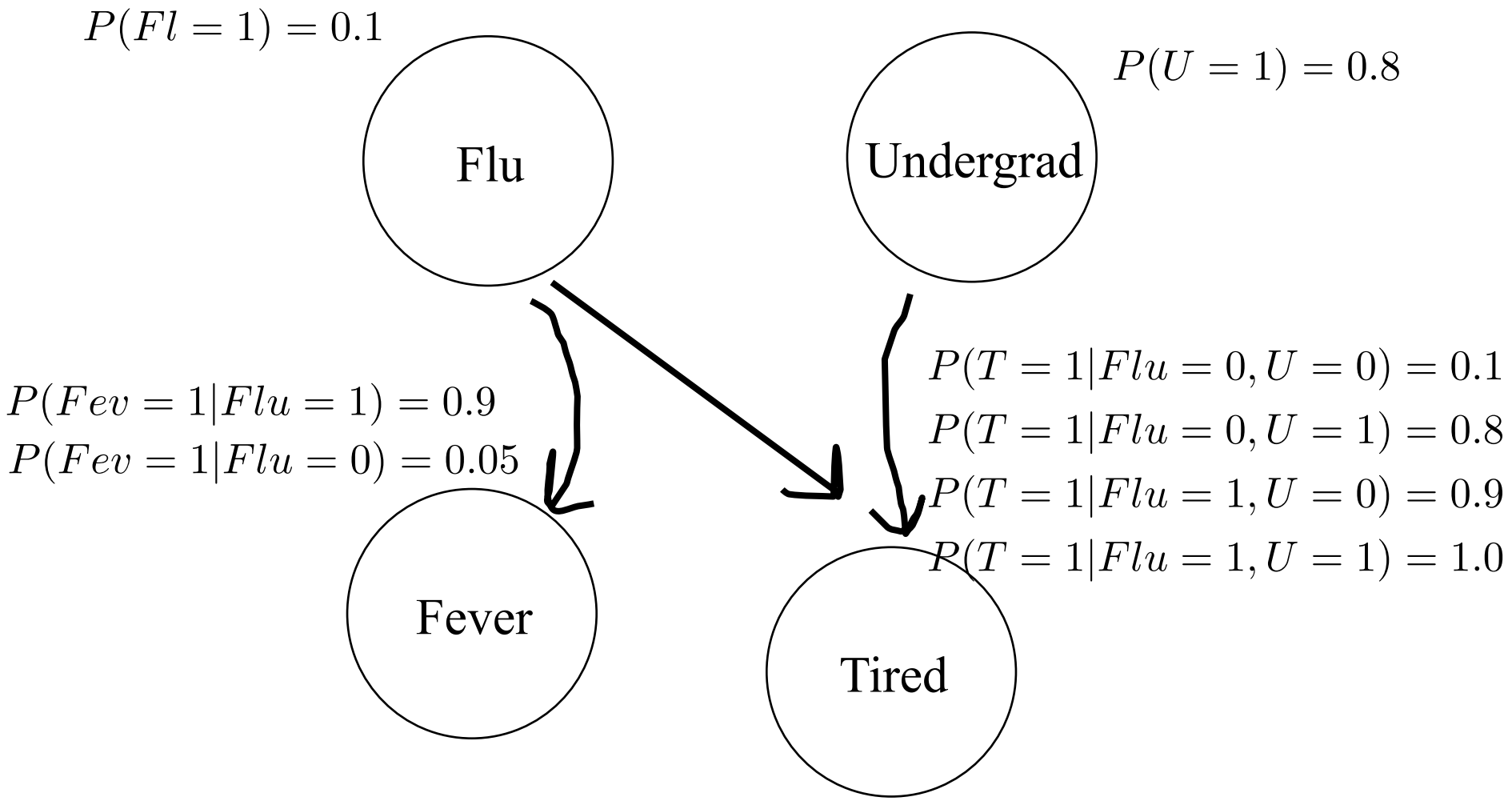
You simply need to give:

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

for each random variable.

Prob can be a conditional probability table or an equation!

Probabilistic Model

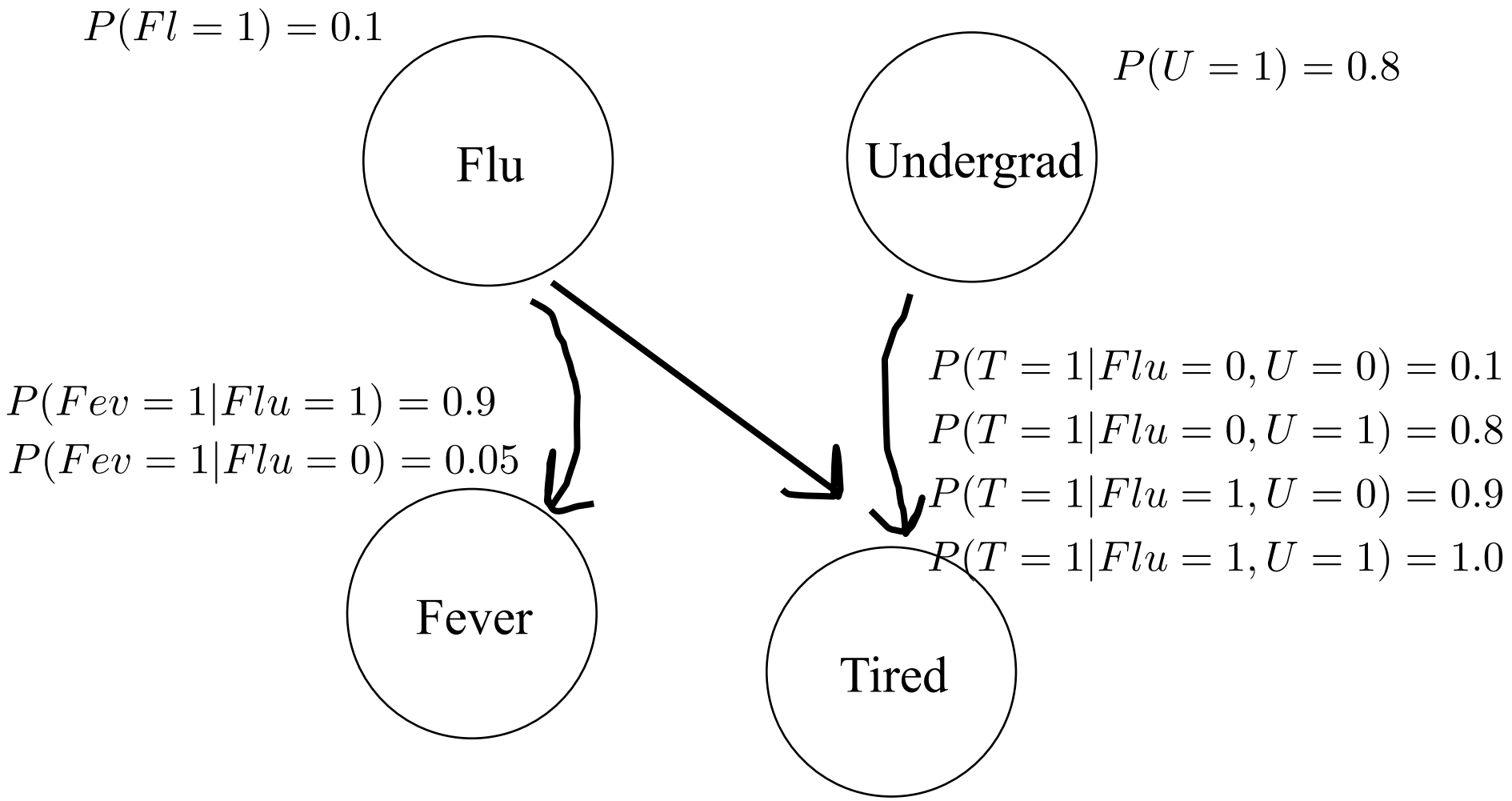


Joint Sampling

```
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 likelihood 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 M
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 M
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 M
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 M
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
```

```
# Method: Probability of Flu Given Observation
# ---------------------------
# Calculate the probability of flu given many
# samples from the joint distribution and a set
# of ovservations to condition on.
def probFluGivenObs(samples, obs):
    # reject all samples which don't align
    # with condition
    keepSamples = []
    for sample in samples:
        if checkObsMatch(sample, obs):
            keepSamples.append(sample)

    # from remaining, simply count...
    fluCount = 0
    for sample in keepSamples:
        [flu, und, fev, tir] = sample
        if flu == 1:
            fluCount += 1

    # counting can be so sweet...
    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
>
```

Lets try it!

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

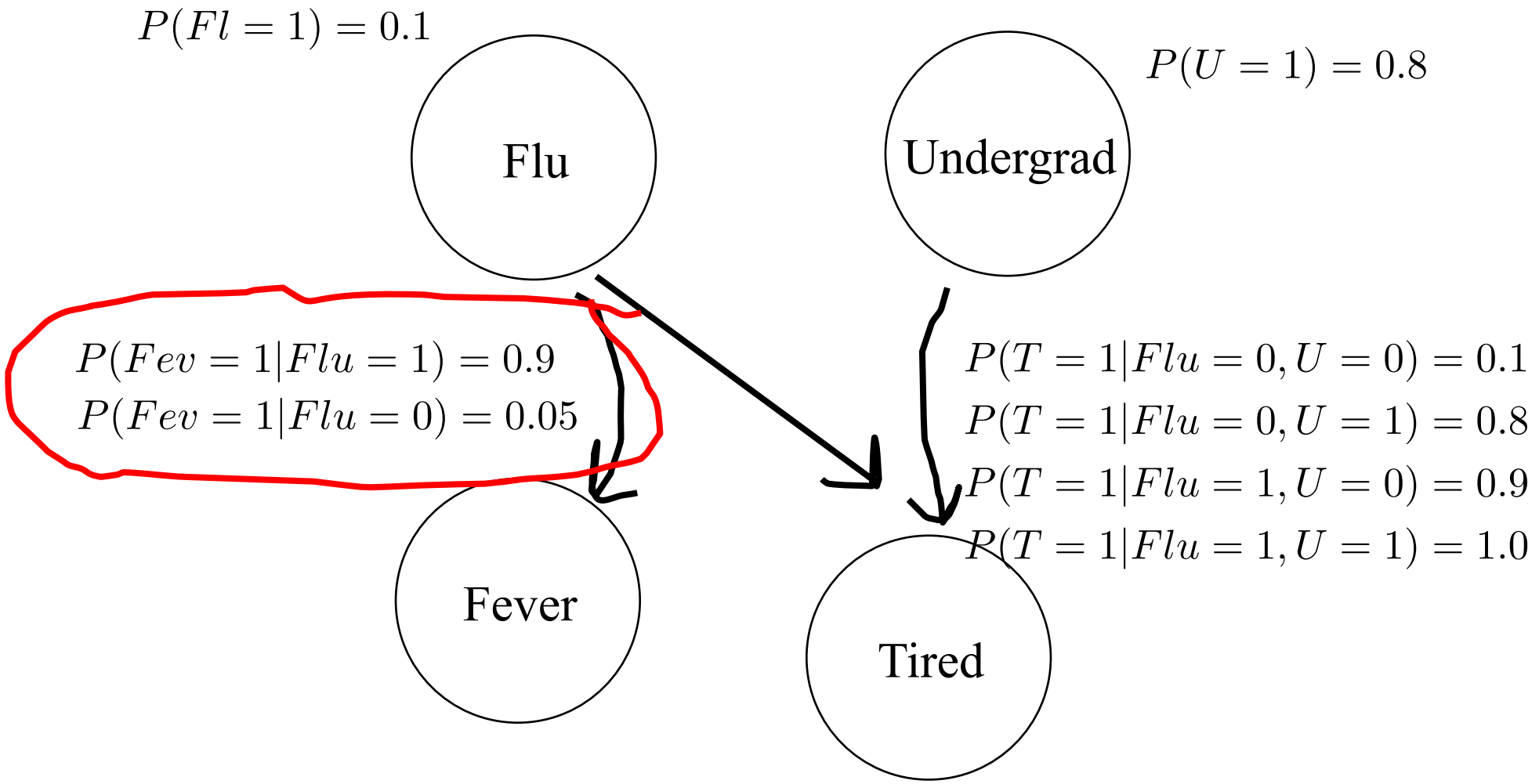
With enough samples:

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

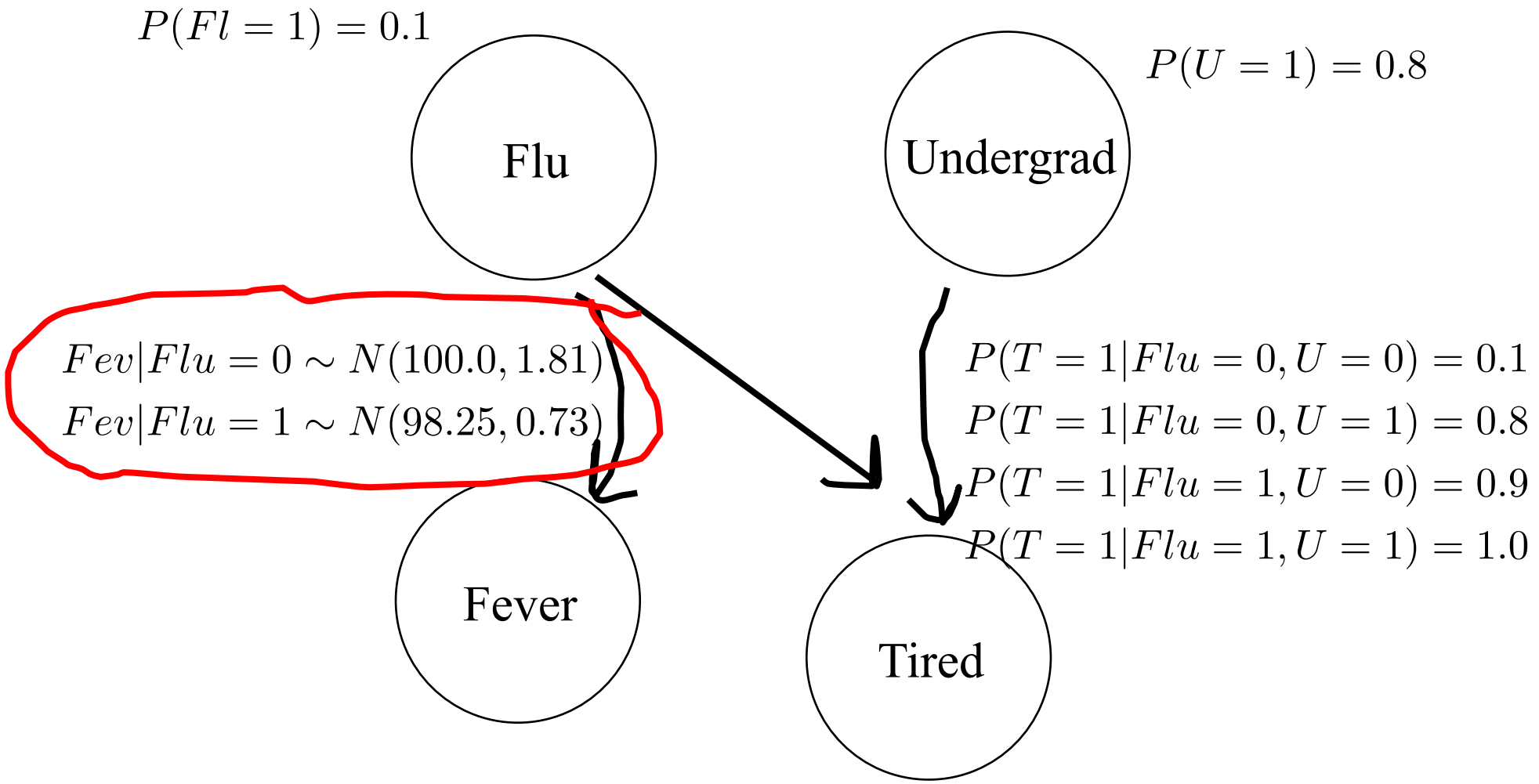
Because your samples are a representation of the joint distribution…

What’s the matter with joint sampling?

Probabilistic Model



Probabilistic Model



BAE's Theorem?

$$P(A \mid B\ E) = \frac{P(B \mid A\ E)\ P(A \mid E)}{P(B \mid E)}$$

$P(F = 1 \mid \text{all other rvs})$

Know: P(symptom | flu, undergrad) P(flu) P(undergrad)

Flu is independent of undergrad

Tired and fever are conditionally independent given flu, undergrad

---

$$P(F = 1|\text{symptoms}, U = u)$$

$$= \frac{P(\text{symptoms}|F = 1, U = u)P(F = 1|U = u)}{P(\text{symptoms}|U = u)}$$

$$\propto P(\text{symptoms}|F = 1, U = u)P(F = 1|U = u)$$

$$\propto P(F = 1)P(\text{symptoms}|F = 1, U = u)$$

$$\propto P(F = 1)\prod_i P(\text{symptom}_i|F = 1, U = u)$$

$$P(F = 1|\text{all other rvs}) \propto P(F = 1) \prod_i P(\text{symptom}_i|F = 1, U = u)$$

```
120 def sampleFlu(sample):
121     f1 = getFluPr1(sample)
122     f0 = getFluPr0(sample)
123     p1 = f1 / (f1 + f0)
124     return bern(p1)
125
126 def getFluPr0(sample):
127     [_, und, fev, tir] = sample
128     pFlu0 = 0.9
129     pFev = getPrFeverX(fev, flu=0)
130     pTir = getPrTiredX(tir, und=und, flu=0)
131     return pFlu0 * pFev * pTir
132
133 def getFluPr1(sample):
134     [_, und, fev, tir] = sample
135     pFlu1 = 0.1
136     pFev = getPrFeverX(fev, flu=1)
137     pTir = getPrTiredX(tir, und=und, flu=1)
138     return pFlu1 * pFev * pTir
```

$$P(F = 1|\text{all other rvs}) \propto P(F = 1) \prod_i P(\text{symptom}_i|F = 1, U = u)$$

```
120 def sampleFlu(sample):
121     f1 = getFluPr1(sample)
122     f0 = getFluPr0(sample)
123     p1 = f1 / (f1 + f0)
124     return bern(p1)
125
126 def getFluPr0(sample):
127     [_, und, fev, tir] = sample
128     pFlu0 = 0.9
129     pFev = getPrFeverX(fev, flu=0)
130     pTir = getPrTiredX(tir, und=und, flu=0)
131     return pFlu0 * pFev * pTir
132
133 def getFluPr1(sample):
134     [_, und, fev, tir] = sample
135     pFlu1 = 0.1
136     pFev = getPrFeverX(fev, flu=1)
137     pTir = getPrTiredX(tir, und=und, flu=1)
138     return pFlu1 * pFev * pTir
```

$$P(F = 1|\text{all other rvs}) \propto P(F = 1) \prod_i P(\text{symptom}_i|F = 1, U = u)$$

```
120 def sampleFlu(sample):
121     f1 = getFluPr1(sample)
122     f0 = getFluPr0(sample)
123     p1 = f1 / (f1 + f0)
124     return bern(p1)
125
126 def getFluPr0(sample):
127     [_, und, fev, tir] = sample
128     pFlu0 = 0.9
129     pFev = getPrFeverX(fev, flu=0)
130     pTir = getPrTiredX(tir, und=und, flu=0)
131     return pFlu0 * pFev * pTir
132
133 def getFluPr1(sample):
134     [_, und, fev, tir] = sample
135     pFlu1 = 0.1
136     pFev = getPrFeverX(fev, flu=1)
137     pTir = getPrTiredX(tir, und=und, flu=1)
138     return pFlu1 * pFev * pTir
```

$$P(F = 1|\text{all other rvs}) \propto P(F = 1) \prod_i P(\text{symptom}_i|F = 1, U = u)$$

```
120 def sampleFlu(sample):
121     f1 = getFluPr1(sample)
122     f0 = getFluPr0(sample)
123     p1 = f1 / (f1 + f0)
124     return bern(p1)
125
126 def getFluPr0(sample):
127     [_, und, fev, tir] = sample
128     pFlu0 = 0.9
129     pFev = getPrFeverX(fev, flu=0)
130     pTir = getPrTiredX(tir, und=und, flu=0)
131     return pFlu0 * pFev * pTir
132
133 def getFluPr1(sample):
134     [_, und, fev, tir] = sample
135     pFlu1 = 0.1
136     pFev = getPrFeverX(fev, flu=1)
137     pTir = getPrTiredX(tir, und=und, flu=1)
138     return pFlu1 * pFev * pTir
```

$$P(F = 1|\text{all other rvs}) \propto P(F = 1) \prod_i P(\text{symptom}_i|F = 1, U = u)$$

```
120 def sampleFlu(sample):
121     f1 = getFluPr1(sample)
122     f0 = getFluPr0(sample)
123     p1 = f1 / (f1 + f0)
124     return bern(p1)
125
126 def getFluPr0(sample):
127     [_, und, fev, tir] = sample
128     pFlu0 = 0.9
129     pFev = getPrFeverX(fev, flu=0)
130     pTir = getPrTiredX(tir, und=und, flu=0)
131     return pFlu0 * pFev * pTir
132
133 def getFluPr1(sample):
134     [_, und, fev, tir] = sample
135     pFlu1 = 0.1
136     pFev = getPrFeverX(fev, flu=1)
137     pTir = getPrTiredX(tir, und=und, flu=1)
138     return pFlu1 * pFev * pTir
```

$$P(F = 1|\text{all other rvs}) \propto P(F = 1) \prod_i P(\text{symptom}_i|F = 1, U = u)$$

```
120 def sampleFlu(sample):
121     f1 = getFluPr1(sample)
122     f0 = getFluPr0(sample)
123     p1 = f1 / (f1 + f0)
124     return bern(p1)
125
126 def getFluPr0(sample):
127     [_, und, fev, tir] = sample
128     pFlu0 = 0.9
129     pFev = getPrFeverX(fev, flu=0)
130     pTir = getPrTiredX(tir, und=und, flu=0)
131     return pFlu0 * pFev * pTir
132
133 def getFluPr1(sample):
134     [_, und, fev, tir] = sample
135     pFlu1 = 0.1
136     pFev = getPrFeverX(fev, flu=1)
137     pTir = getPrTiredX(tir, und=und, flu=1)
138     return pFlu1 * pFev * pTir
```

General Inference Summary

- **Straight Math** is fast, but can be prohibitively hard for complex models (see hw).
- **Joint Sampling** is really easy to program but fails for continuous variables (and when what you are conditioning on is rare)
- **Take CS228** to learn about many, many more algorithms!


Early morning dip
in the paper
recycling

Stretch!

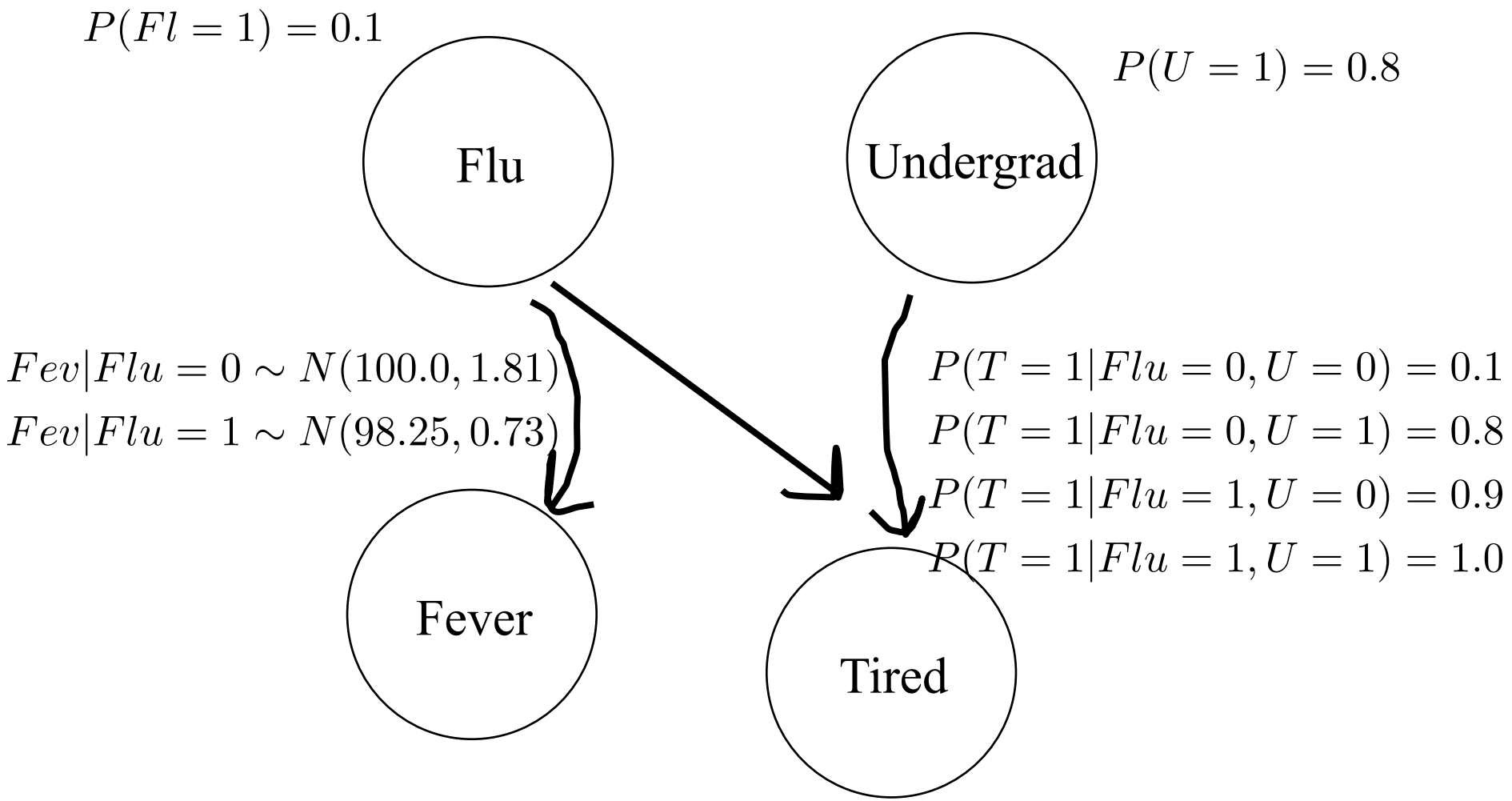

Trash Demon
is displeased
with your
offering

Parameter Estimation

Noah Arthurs
CS109, Stanford University

MACHINE LEARNING

Where Do The Numbers Come From?



Pedagogical Pause

At this point, if you are given a *model*, with all the involved probabilities, you can make predictions

But what if you want to *learn* the probabilities in the model?

But what if you want to *learn* the probabilities in the model?

Oh can we also learn the *structure* of the model too?

But what if you want to *learn* the probabilities in the model?

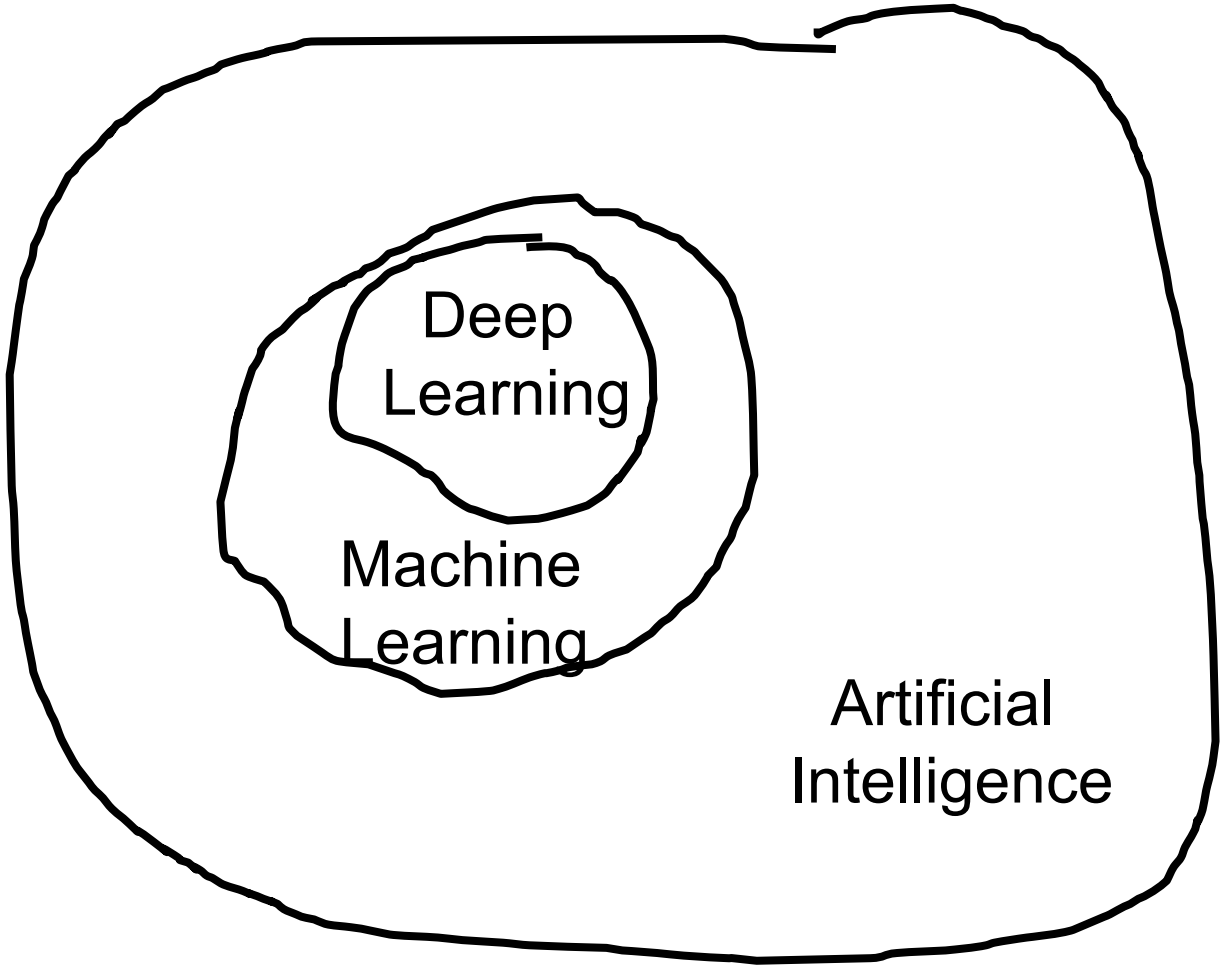
~~Oh can we also learn the *structure* of the model too?~~

I wish. Another day ☺

But what if you want to *learn* the probabilities in the model?

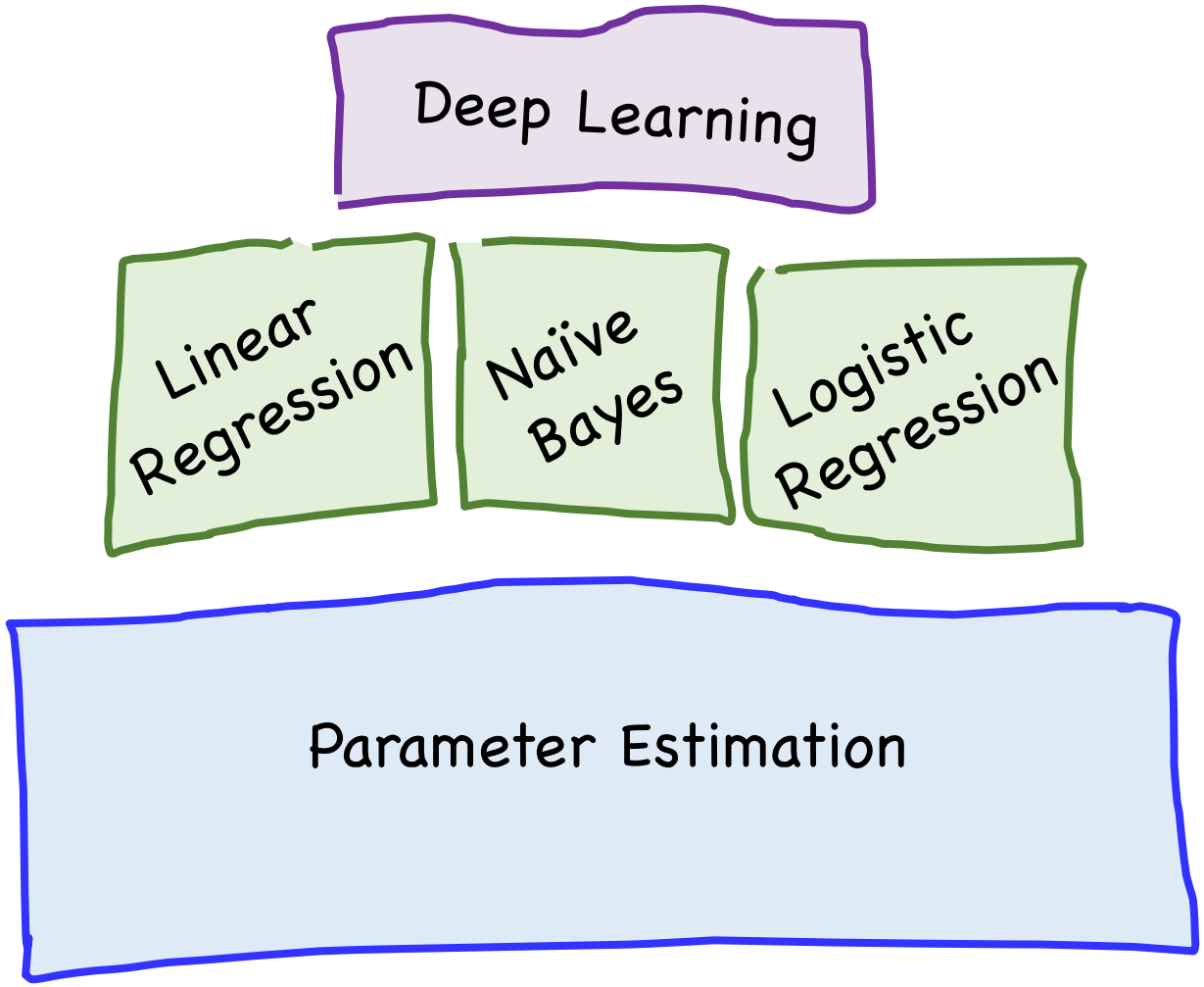
Machine Learning

AI and Machine Learning

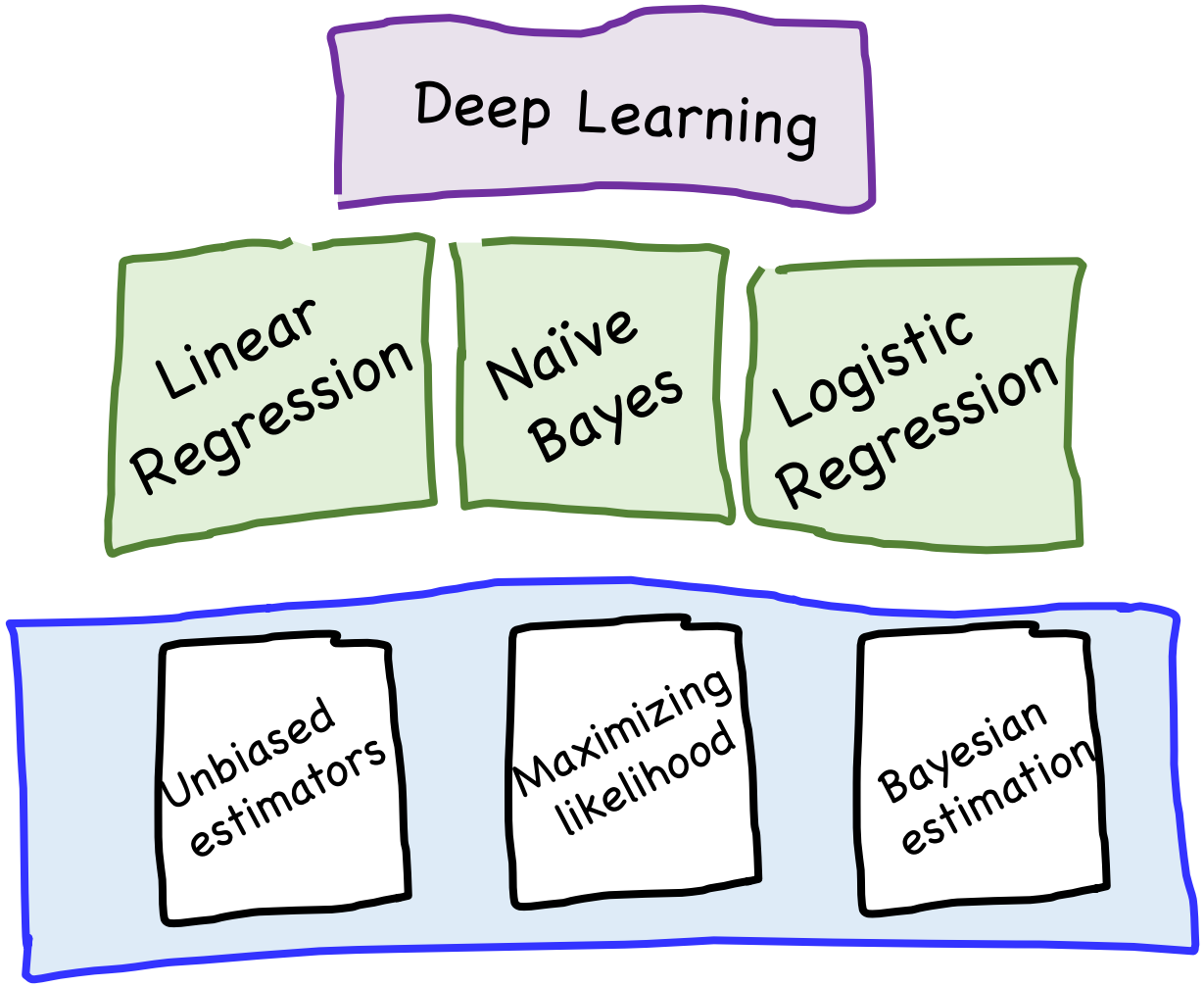


ML: Rooted in probability theory

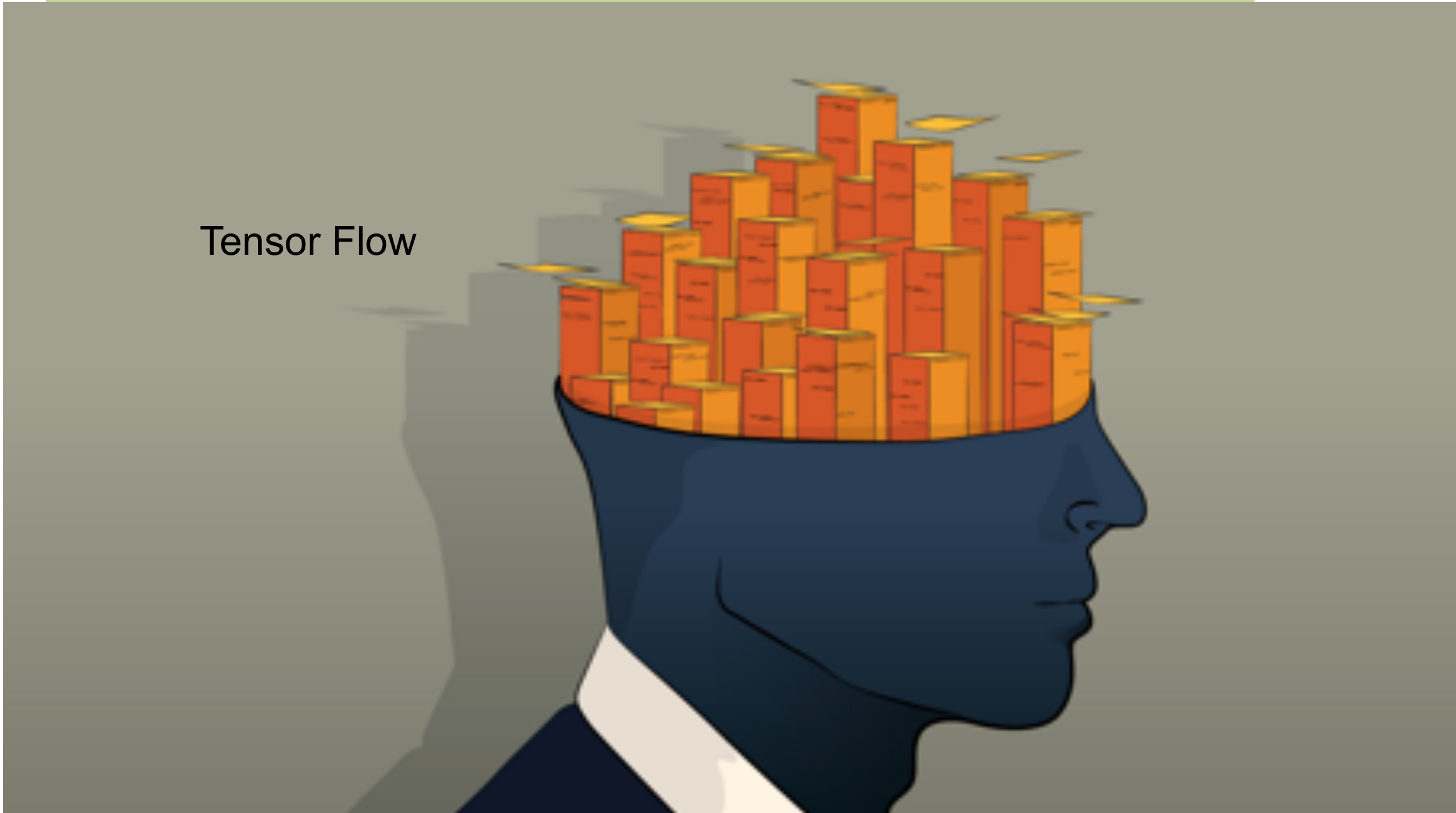
Our Path


Deep Learning
Linear Regression
Naïve Bayes
Logistic Regression
Parameter Estimation

Our Path


Deep Learning
Linear Regression
Naïve Bayes
Logistic Regression
Unbiased estimators
Maximizing likelihood
Bayesian estimation

Jump Straight to Deep Learning?

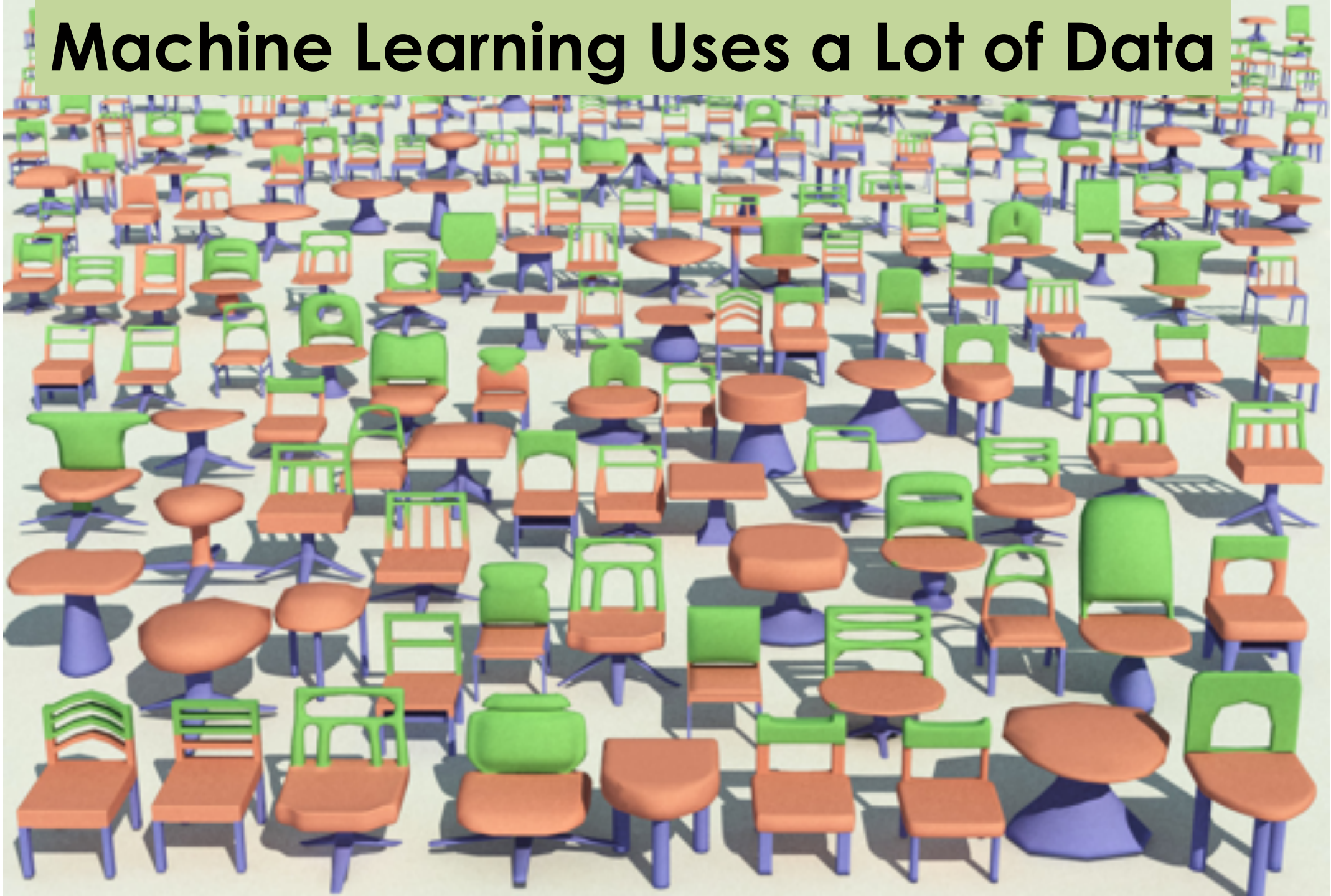


Jump Straight to Deep Learning?

Understand the theory to help you debug

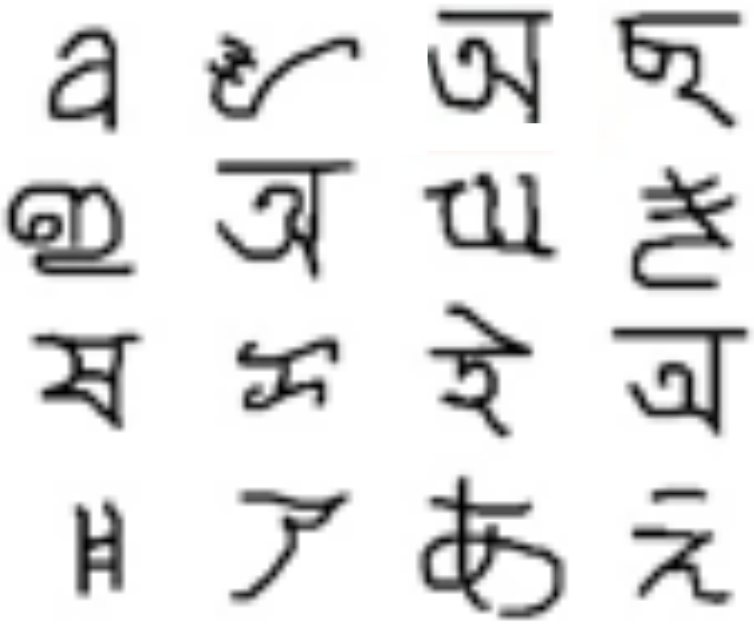
But another reason…

Machine Learning Uses a Lot of Data



One Shot Learning

Single training example: 

Test set: 

One Shot Learning

Single training example:

Computers struggle...

… especially for **human** problems.

Understand the theory
to push on the **grand challenges**

Once upon a time…

…there was parameter estimation

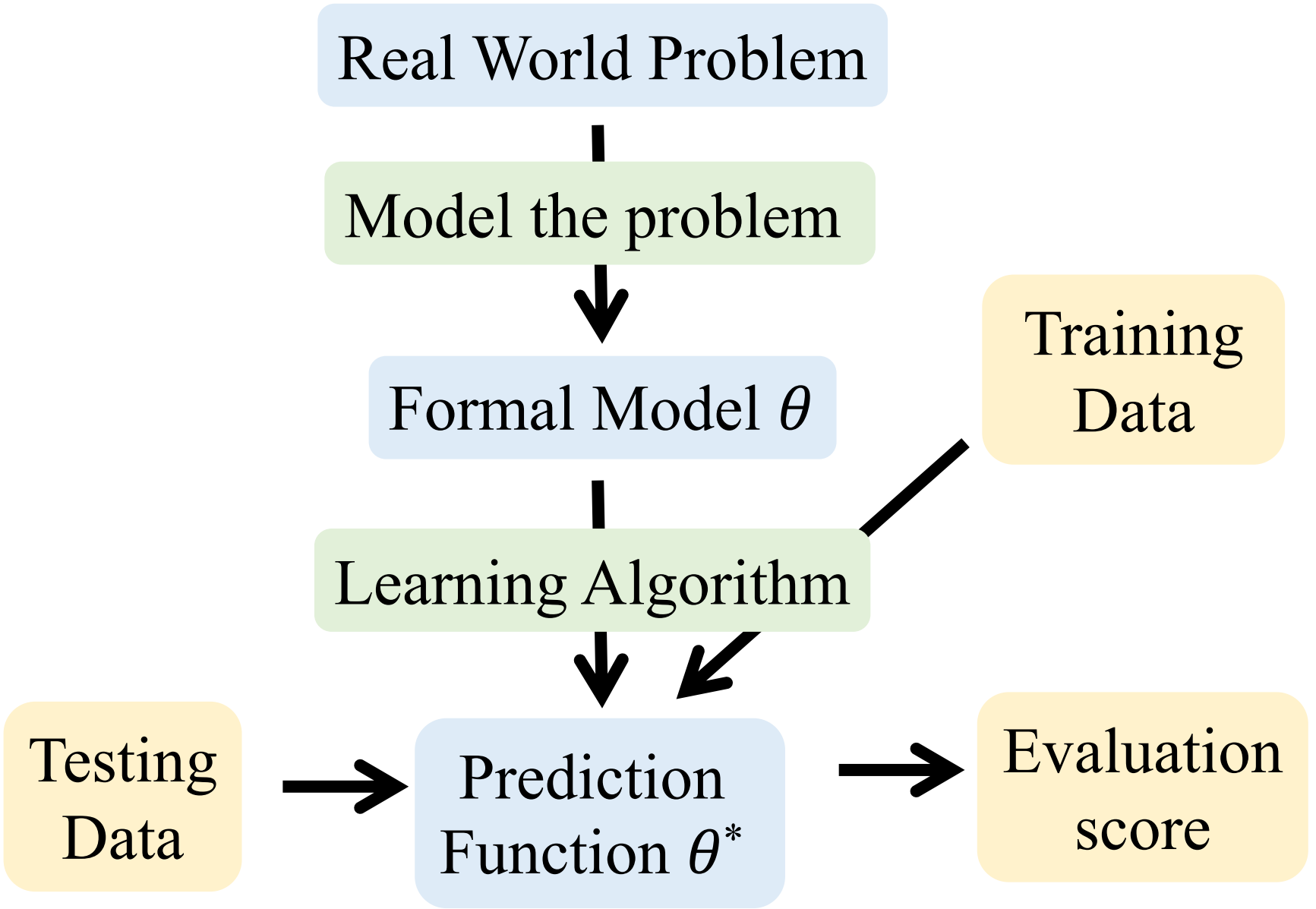
What are Parameters?

- Consider some probability distributions:
  - Ber(p) $\quad \theta = p$
  - Poi(λ) $\quad \theta = \lambda$
  - Uni(α, β) $\quad \theta = (\alpha, \beta)$
  - Normal(μ, σ^2) $\quad \theta = (\mu, \sigma^2)$
  - $Y = mX + b$ $\quad \theta = (m, b)$
  - etc…
- Call these “parametric models”
- Given model, **parameters** yield actual distribution
  - Usually refer to parameters of distribution as θ
  - Note that θ that can be a vector of parameters

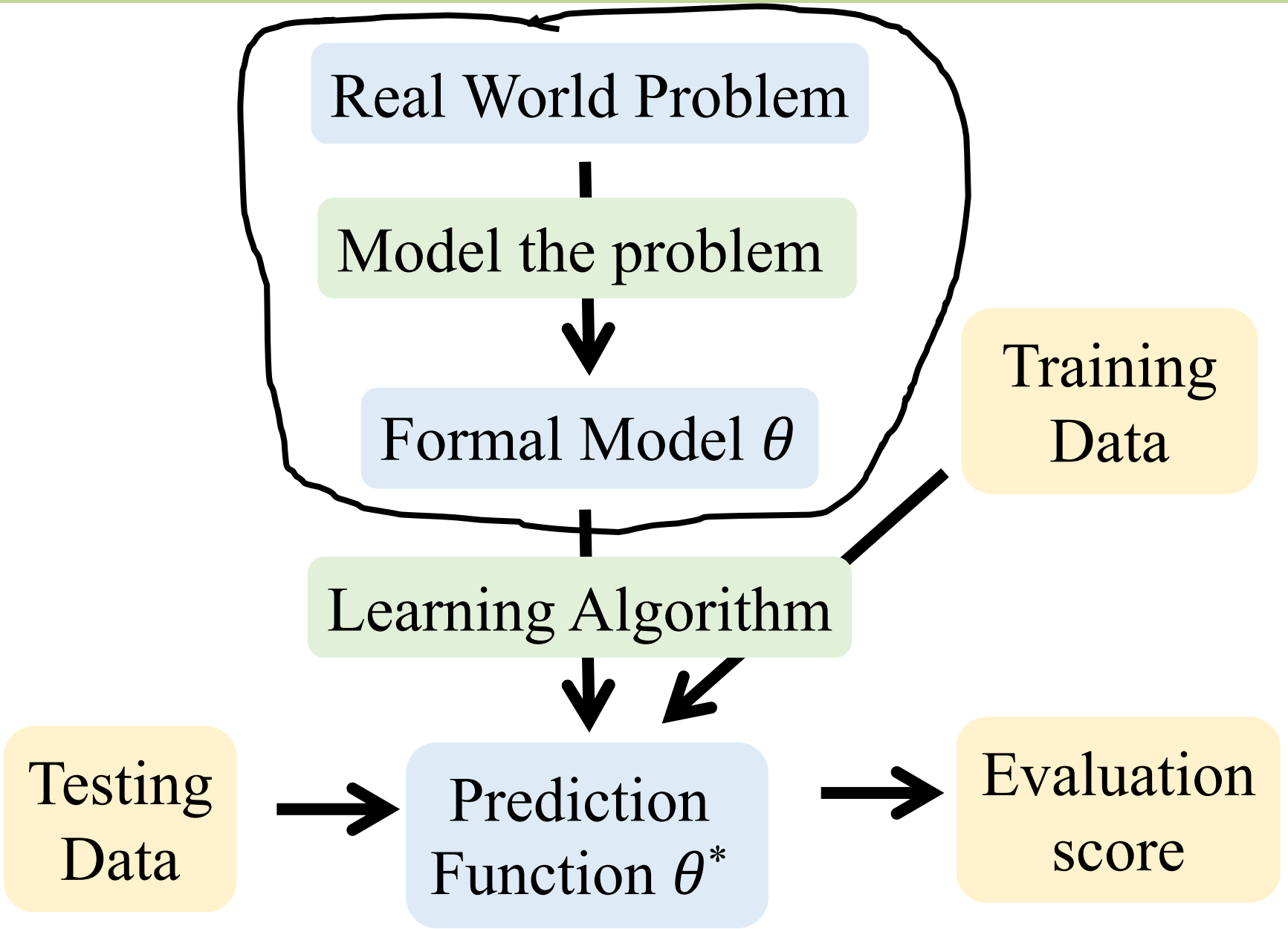
Why Do We Care?

- In real world, don't know "true" parameters
  - But, **we do get to observe data**
    - E.g., number of times coin comes up heads, lifetimes of disk drives produced, number of visitors to web site per day, etc.
  - Need to estimate model parameters from data
  - "Estimator" is random variable estimating parameter
- Estimate of parameters allows:
  - Better understanding of process producing data
  - Future **predictions** based on model
  - Simulation of processes

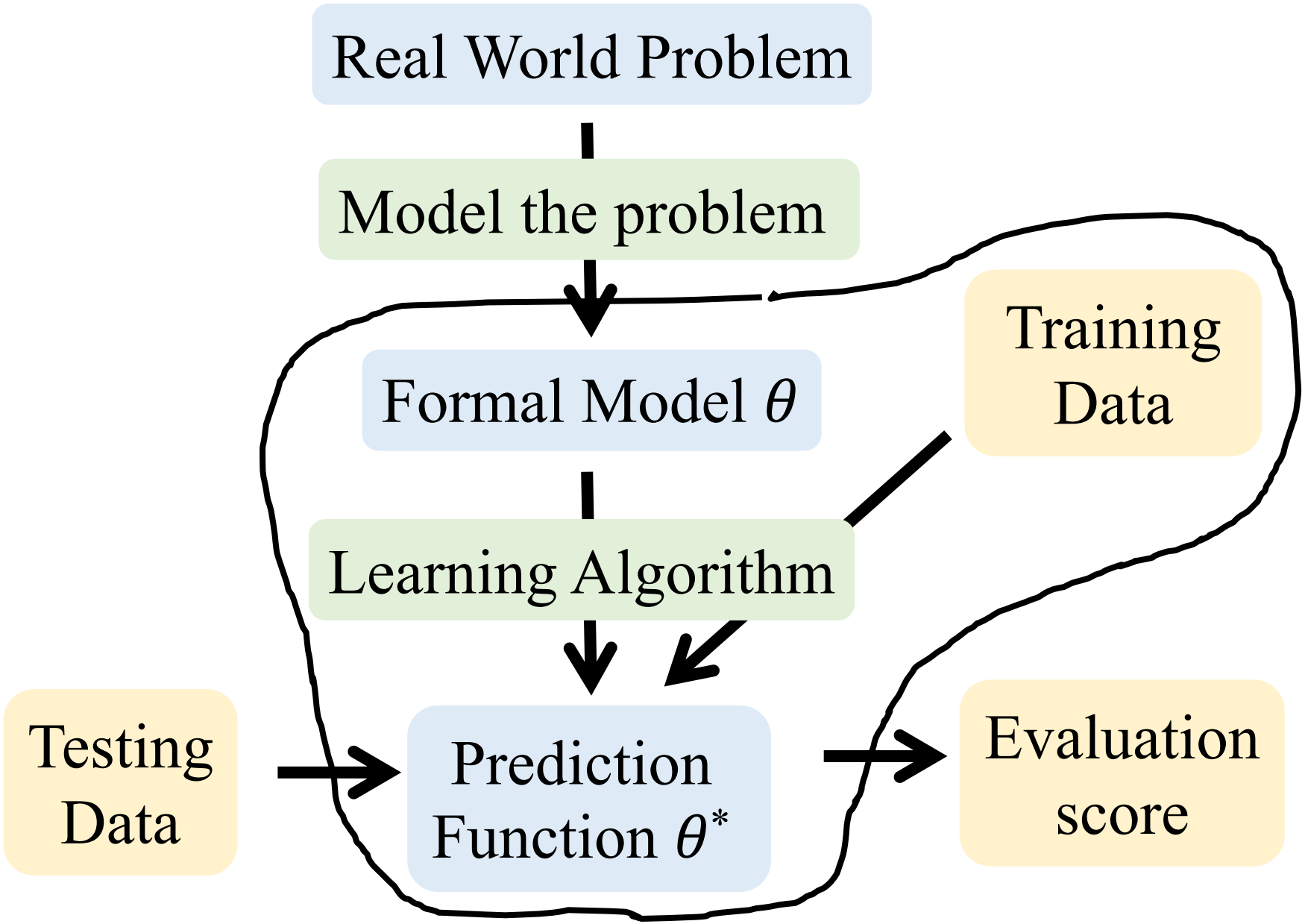
Supervised Learning



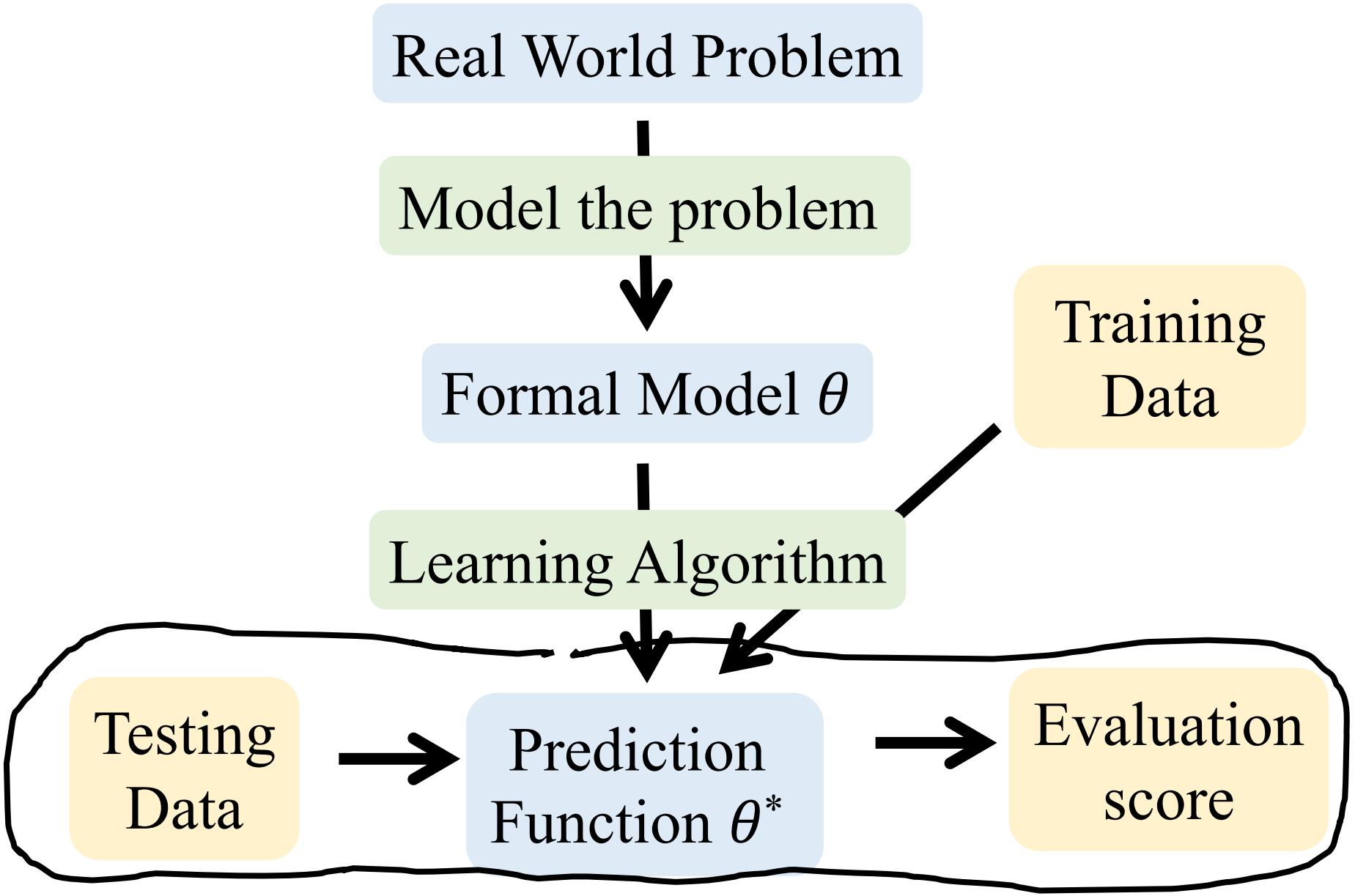
Modelling


Real World Problem
Model the problem
Formal Model θ
Training Data
Learning Algorithm
Testing Data
Prediction Function θ^*
Evaluation score

Training


Real World Problem
Model the problem
Formal Model θ
Learning Algorithm
Training Data
Testing Data
Prediction Function θ^*
Evaluation score

Testing


Real World Problem
Model the problem
Formal Model θ
Training Data
Learning Algorithm
Testing Data
Prediction Function θ^*
Evaluation score