20: Sampling + Inference

Lisa Yan

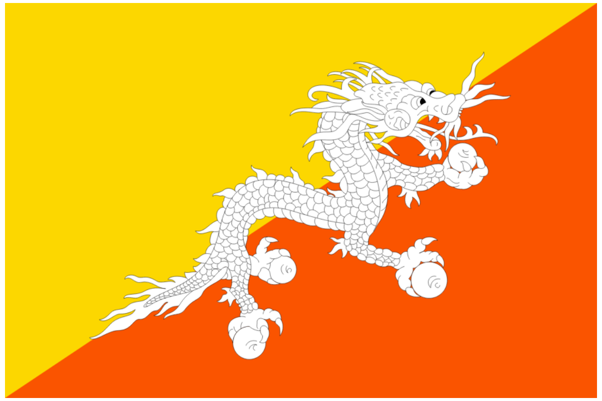
November 6, 2019

Motivating example

Review

You want to know the true mean and variance of happiness in Bhutan.

- But you can't ask everyone.
- You poll 200 random people, a **sample**.



A happy
Bhutanese person

The underlying distribution F has unknown statistics:

- μ, the **population mean**
- σ^2, the **population variance**

Sample statistics

Review

Sample mean

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

estimates μ

Sample variance

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

estimates σ^2

(both random variables that depend on your sample)

Standard error of the mean

$$SE = \sqrt{\frac{S^2}{n}}$$

Estimates variance of sample mean,

$$\text{Var}(\bar{X}) = \frac{\sigma^2}{n}$$

Standard error of the variance

can estimate via bootstrapping

Standard error

1. Mean happiness:

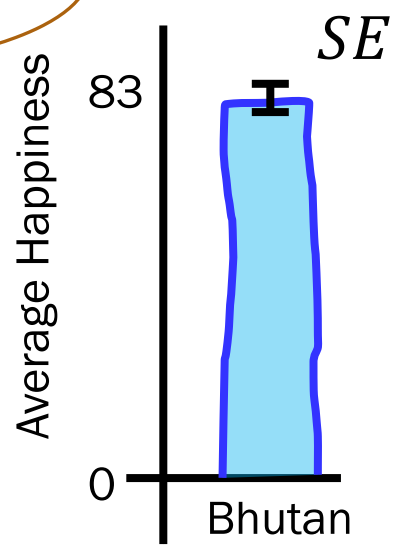
Claim: The average happiness of Bhutan is 83, with a standard error of 1.99.

this is our best estimate of μ

Closed form: $SE = \sqrt{\dfrac{S^2}{n}}$

this is how close we are

✅

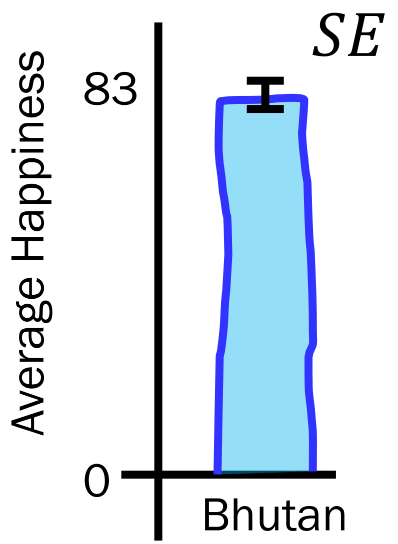


Standard error

1. Mean happiness:

   Claim: The average happiness of Bhutan is 83, with a standard error of 1.99.

   Closed form: $SE = \sqrt{\dfrac{S^2}{n}}$



2. Variance of happiness:

   Claim: The variance of happiness of Bhutan is 793.

   Closed form: Not covered in CS109

this is our best estimate of μ

But how close are we?

We can bootstrap for standard error, which is a statistic of a statistic.

Bootstrap

The Bootstrap:

Probability for Computer Scientists

Today's plan

Bootstrapping

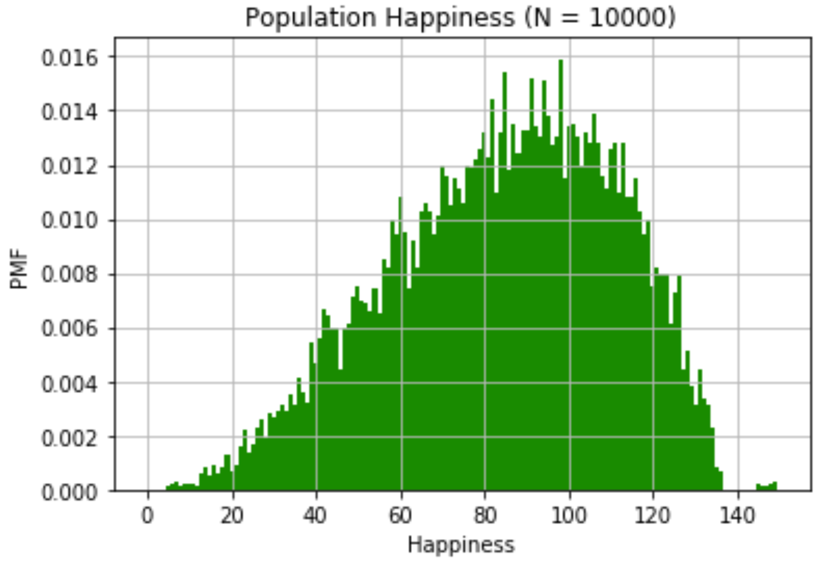
- For a statistic
- For a p-value

Definition: Bayesian Networks

Inference:

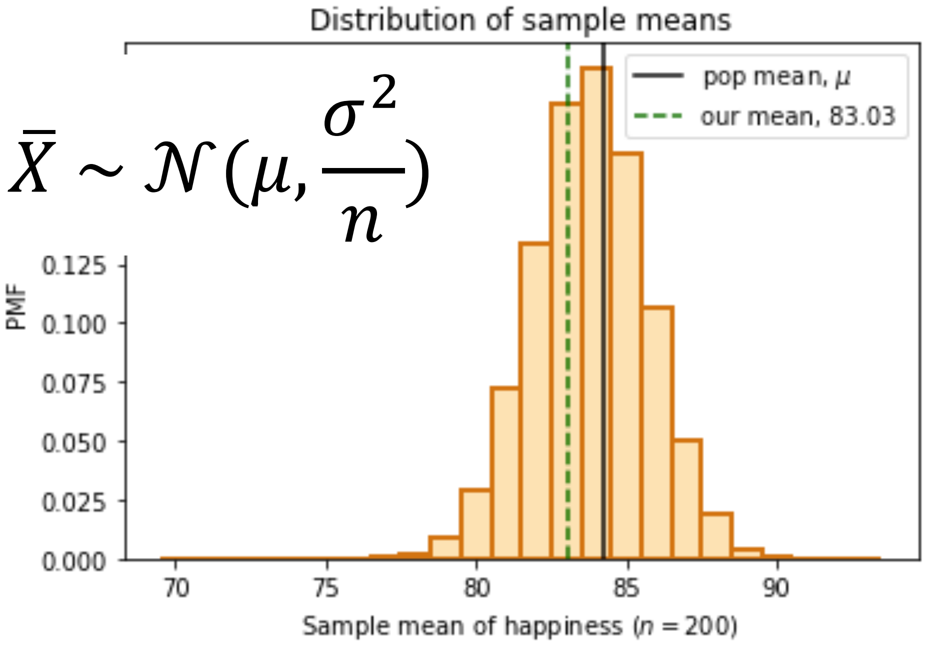
1. Math
2. Rejection sampling ("joint" sampling)
3. Optional: Gibbs sampling (MCMC algorithm)

Bootstrap insight



Distribution of means

$$\bar{X} \sim \mathcal{N}\left(\mu, \frac{\sigma^2}{n}\right)$$



If we had the underlying distribution…

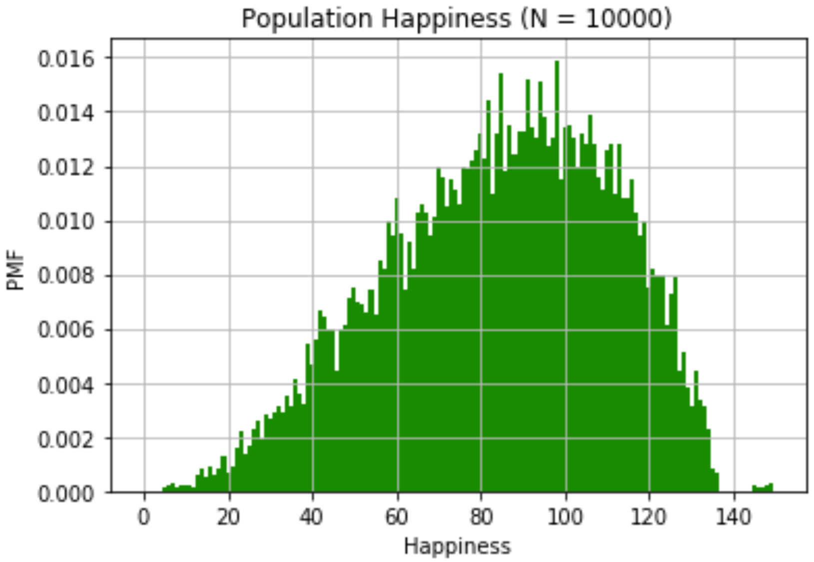
…we could generate a distribution over any statistic and report anything (for example, report $\text{Var}(\bar{X}) = \sigma^2/n$).

If we don't have the underlying distribution, what's our best estimate of the distribution?

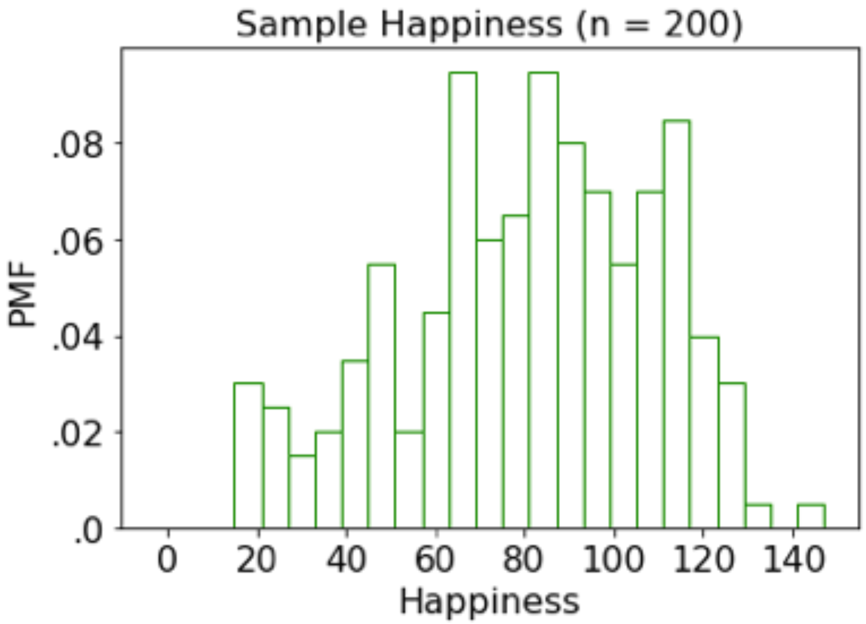
Bootstrap insight

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

*This is just a histogram of your data!

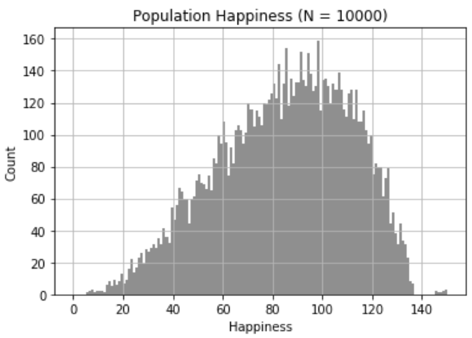


$\approx$

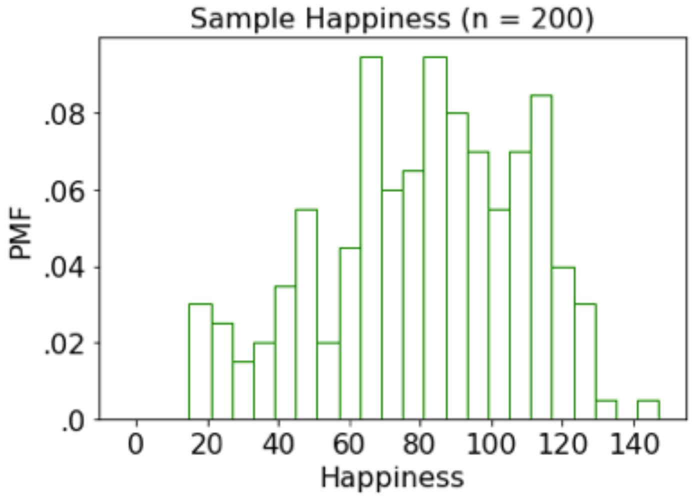


The underlying distribution $\rightarrow$ $F \approx \hat{F}$ $\leftarrow$ the sample distribution (aka the histogram of your data)

Bootstrap is an algorithm

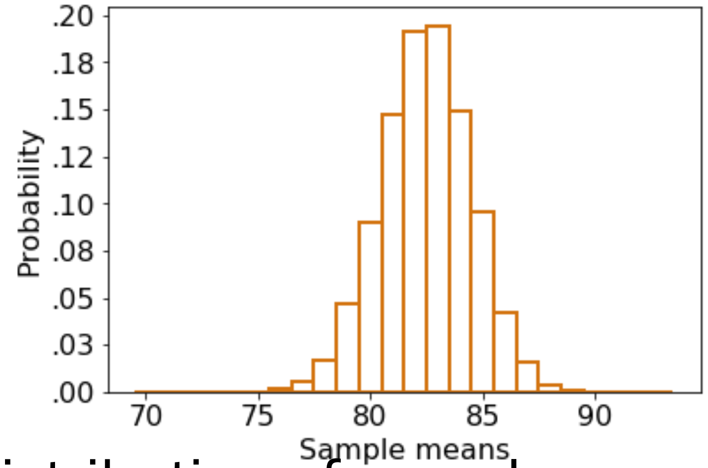


Population distribution
(we don't have this)



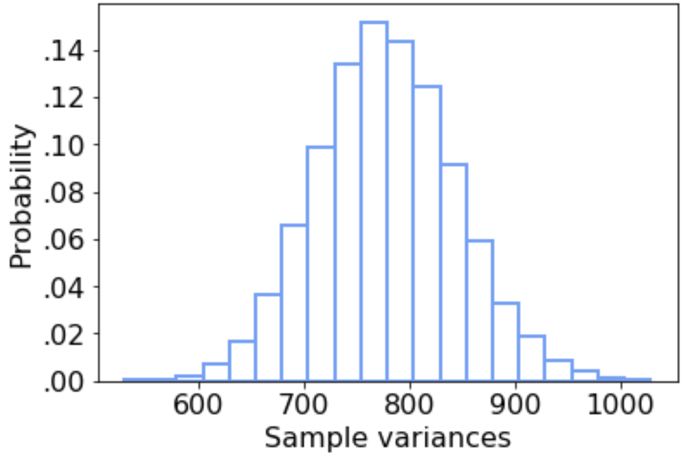
Sample distribution
(we do have this)

Bootstrap means



Distribution of sample means

Bootstrap variances



Distribution of sample variances

Bootstrap algorithm

```
Bootstrap Algorithm (sample):
1. Estimate the PMF using the sample
2. Repeat 10,000 times:
     a. Resample sample.size() from PMF
     b. Recalculate the statistic on the resample
3. You now have a distribution of your statistic
```

What is the distribution of your statistic?

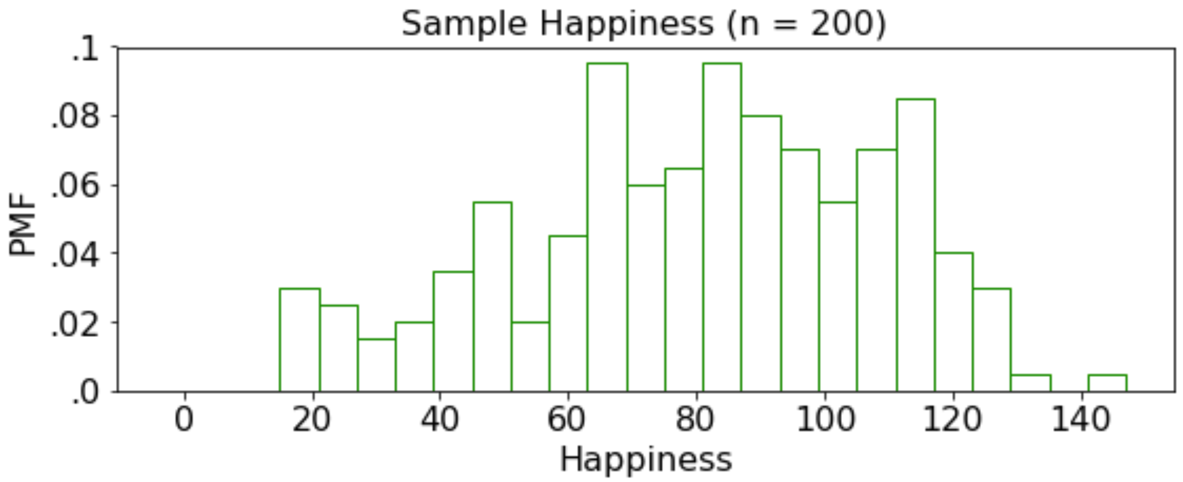
Bootstrapped means

Bootstrap Algorithm (sample):

1. Estimate the **PMF** using the sample
2. Repeat **10,000** times:
    a. Resample **sample.size()** from PMF
    b. Recalculate the **mean** on the resample
3. You now have a **distribution of your mean**

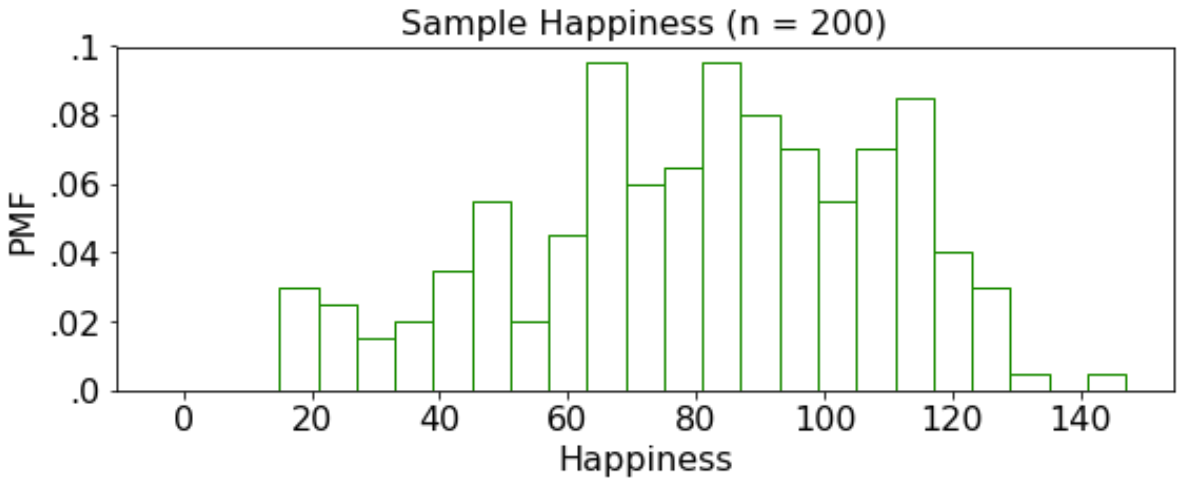
What is the distribution of your mean?

Bootstrapped means



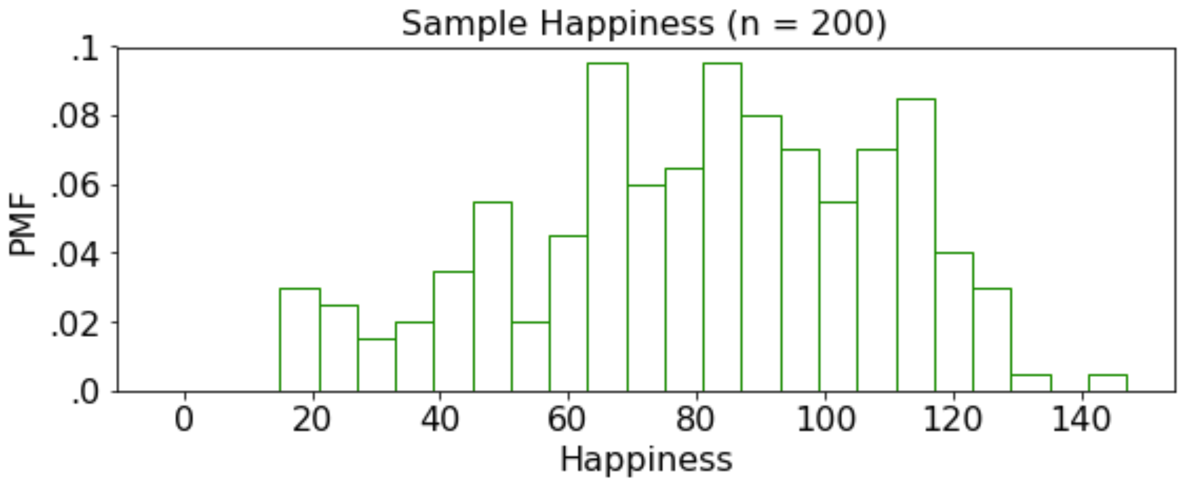
1. Estimate the **PMF** using the sample
2. Repeat **10,000** times:
    a. Resample **sample.size()** from PMF
    b. Recalculate the **mean** on the resample
3. You now have a **distribution of your mean**

Bootstrapped means

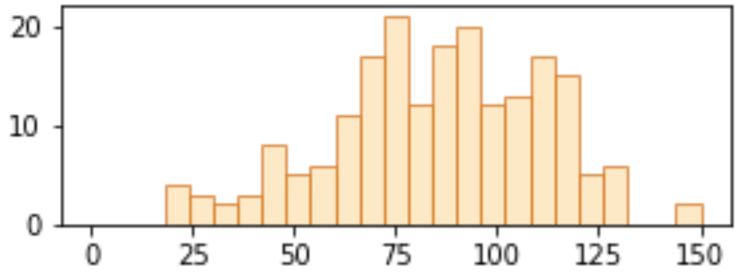


1. Estimate the **PMF** using the sample
2. Repeat **10,000** times:
    a. Resample **sample.size()** from PMF
    b. Recalculate the **mean** on the resample
3. You now have a distribution of your mean

Bootstrapped means

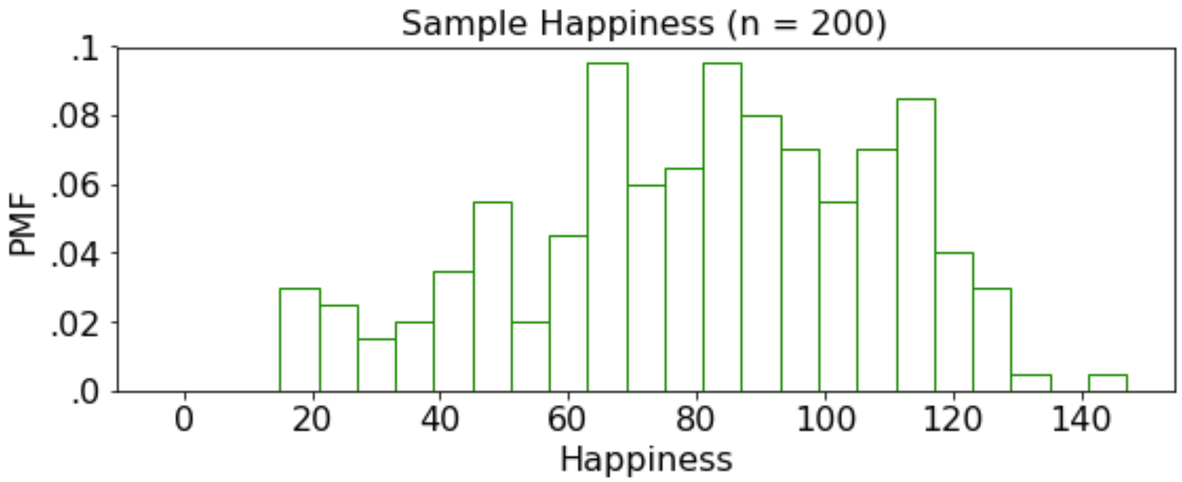


[57, 100, 72, 120, ..., 57]

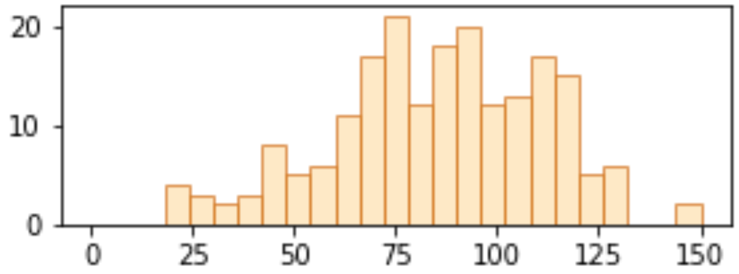


1. Estimate the **PMF** using the sample
2. Repeat **10,000** times:
   a. Resample **sample.size()** from PMF
   b. Recalculate the **mean** on the resample
3. You now have a distribution of your mean

Bootstrapped means



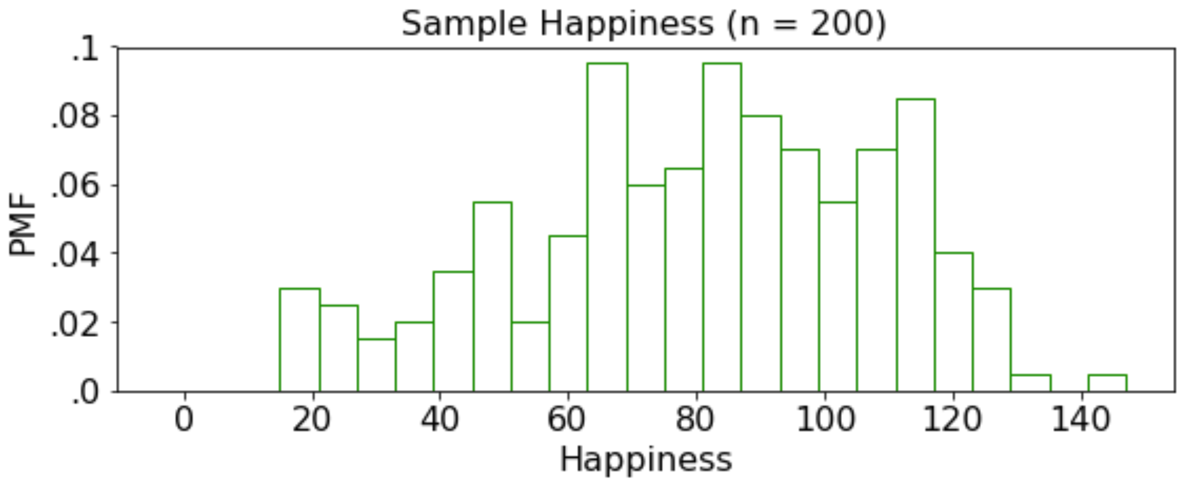
[57, 100, 72, 120, ..., 57]



1. Estimate the **PMF** using the sample
2. Repeat **10,000** times:
    a. Resample **sample.size()** from PMF
    b. Recalculate the **mean** on the resample
3. You now have a distribution of your mean

means = [84.7]

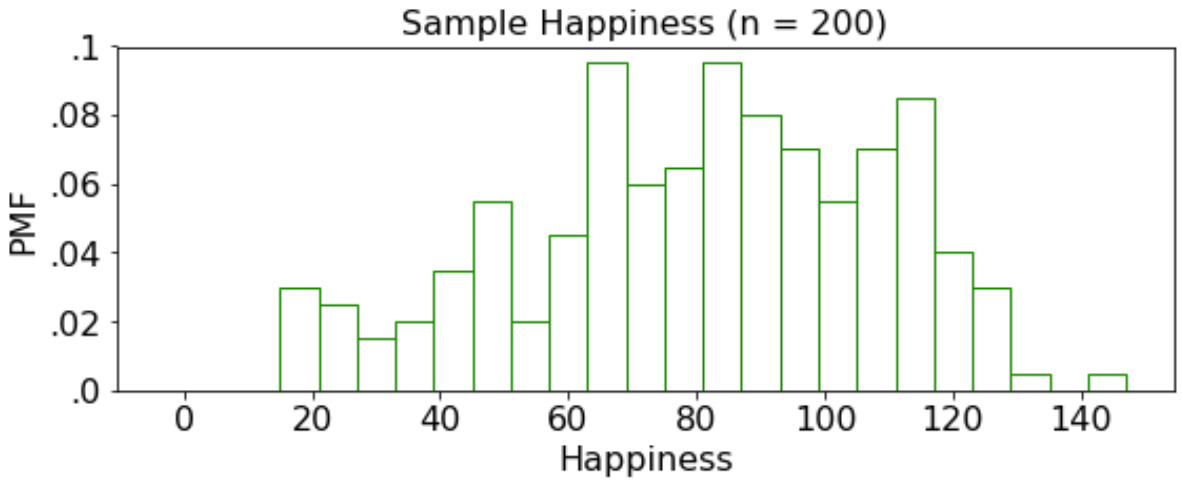
Bootstrapped means



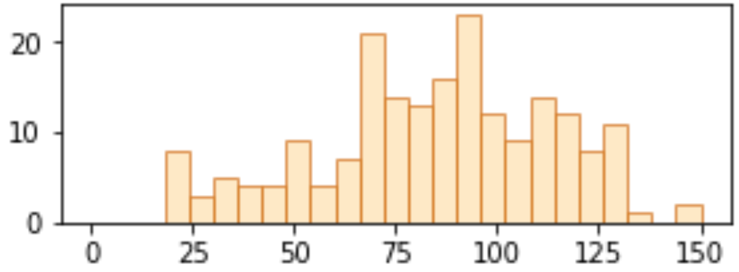
1. Estimate the **PMF** using the sample
2. Repeat **10,000** times:
    a. Resample **sample.size()** from PMF
    b. Recalculate the **mean** on the resample
3. You now have a distribution of your mean

means = [84.7]

Bootstrapped means



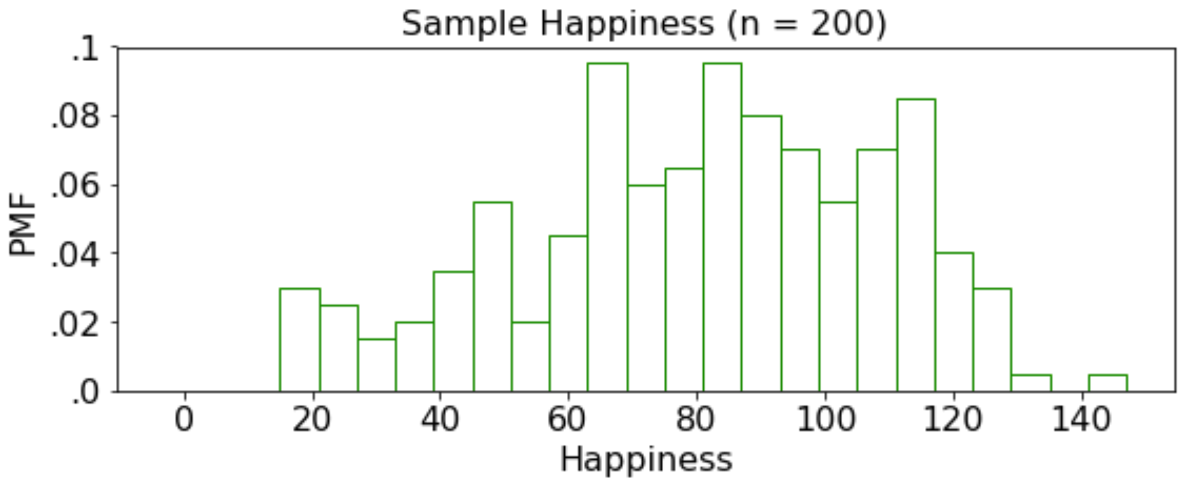
[84, 54, 88, 122, ..., 92]



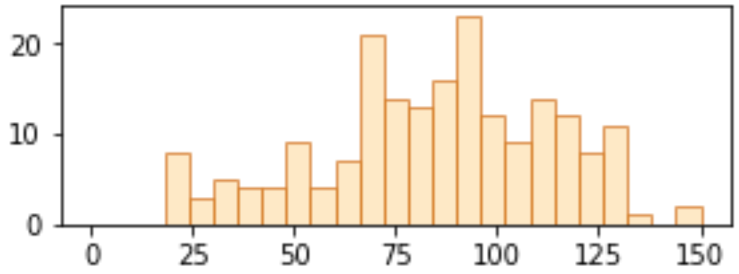
1. Estimate the PMF using the sample
2. Repeat **10,000** times:
   a. Resample **sample.size()** from PMF
   b. Recalculate the **mean** on the resample
3. You now have a distribution of your mean

```
means = [84.7]
```

Bootstrapped means



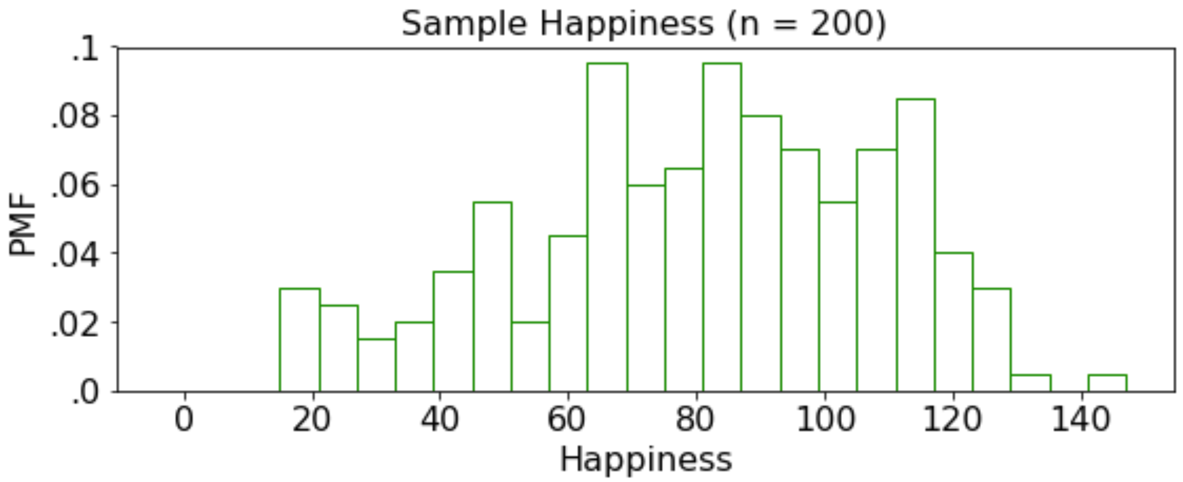
[84, 54, 88, 122, ..., 92]



1. Estimate the **PMF** using the sample
2. Repeat **10,000** times:
   a. Resample **sample.size()** from PMF
   b. Recalculate the **mean** on the resample
3. You now have a distribution of your mean

means = [84.7, 83.9]

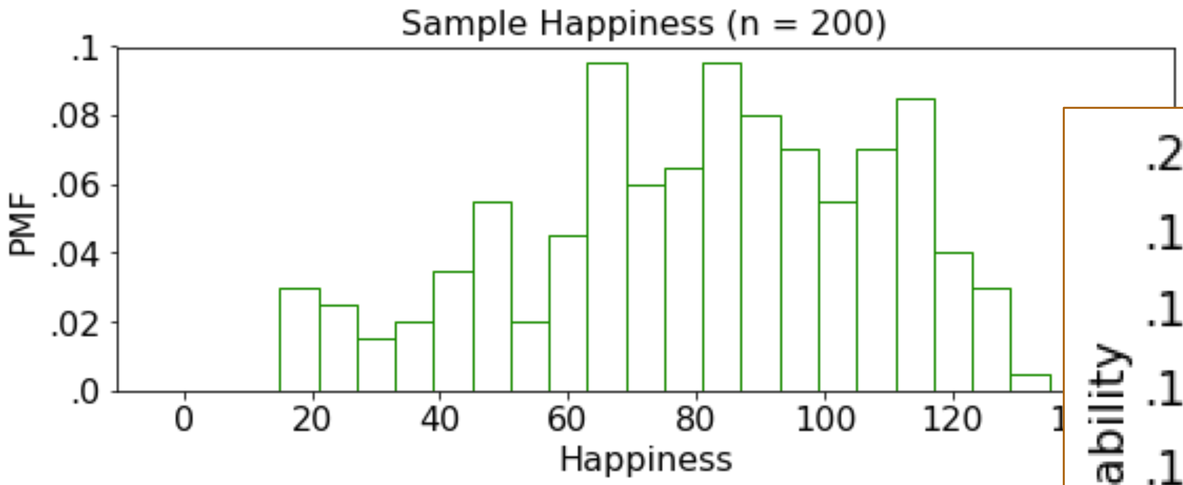
Bootstrapped means

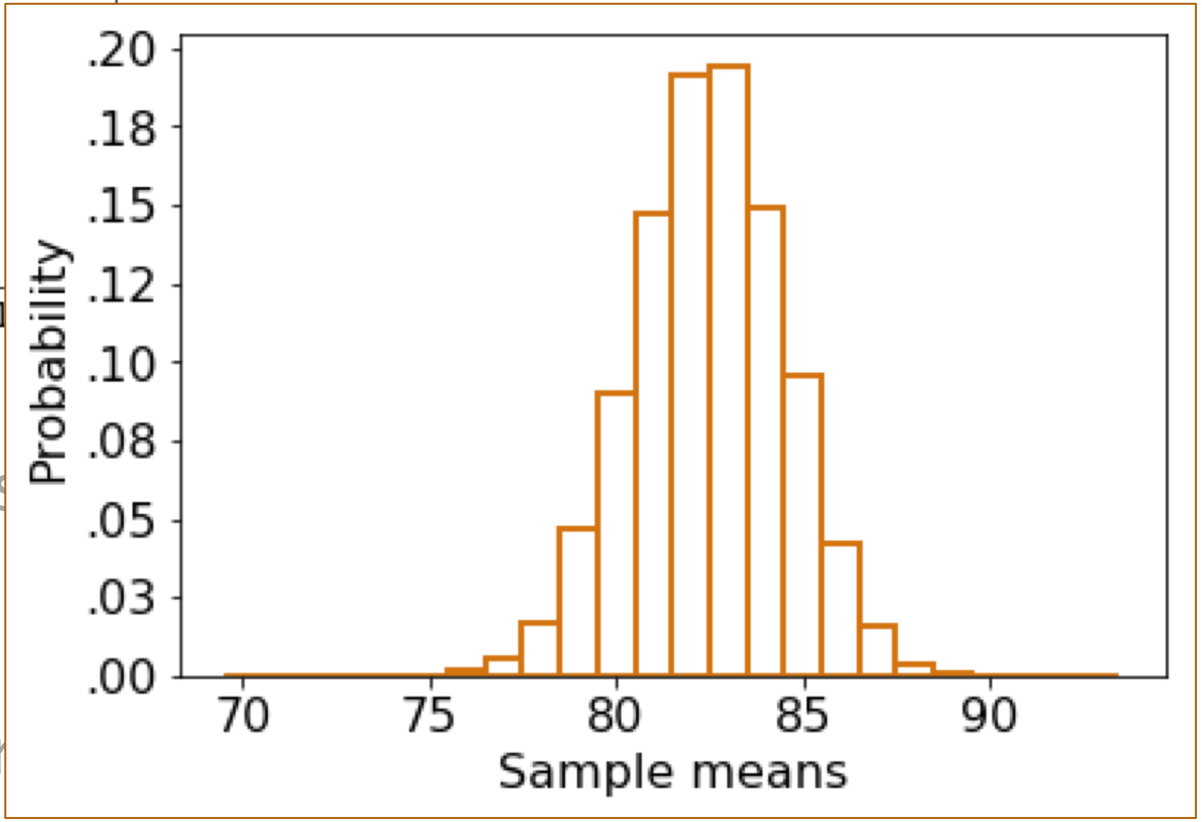


1. Estimate the **PMF** using the sample
2. Repeat **10,000** times:
    a. Resample **sample.size()** from PMF
    b. Recalculate the **mean** on the resample
3. You now have a distribution of your mean

```
means = [84.7, 83.9]
```

Bootstrapped means





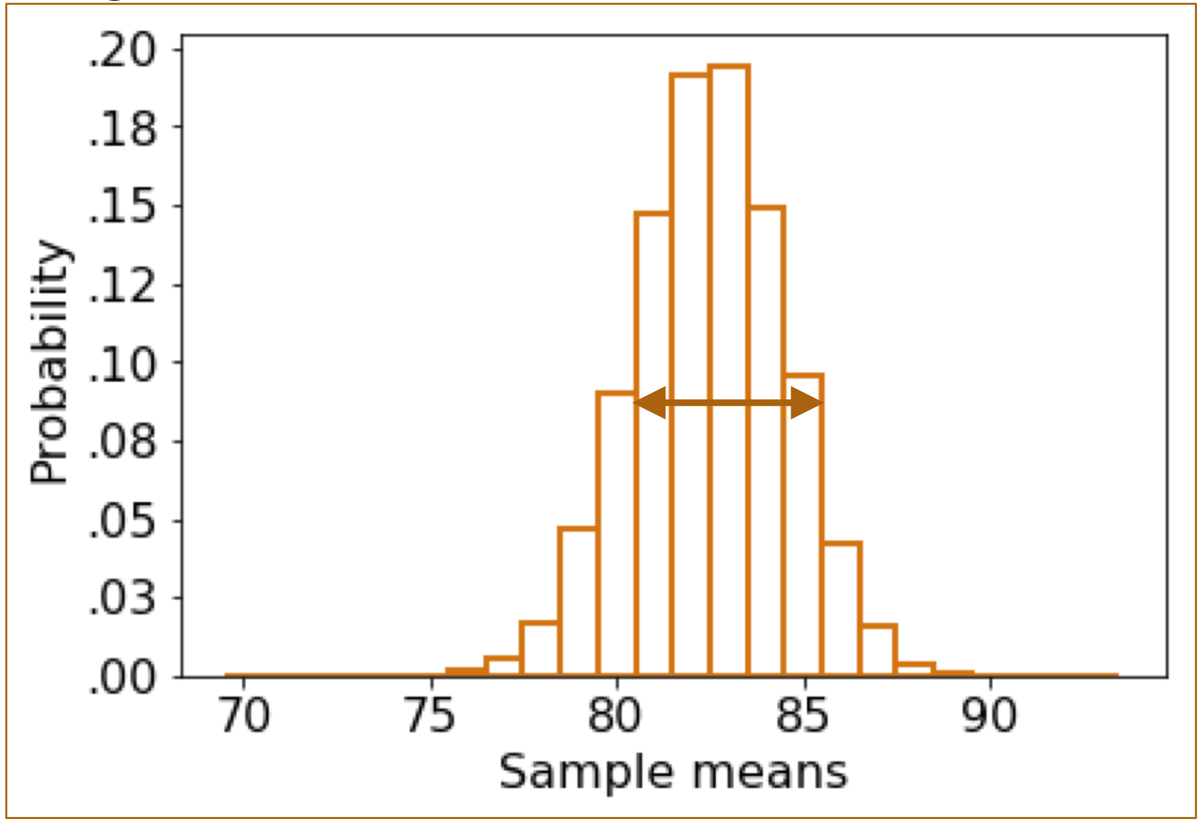
1. Estimate the **PMF** using the s
2. Repeat **10,000** times:
   a. Resample **sample.size()**
   b. Recalculate the **mean** o
3. You now have a **distribution of your mean**

means = [84.7, 83.9, 80.6, 79.8, 90.3, …, 85.2]

Bootstrapped means

3. You now have a **distribution of your mean**

```
means = [84.7, 83.9,
         80.6, 79.8, 90.3,
         …, 85.2]
```

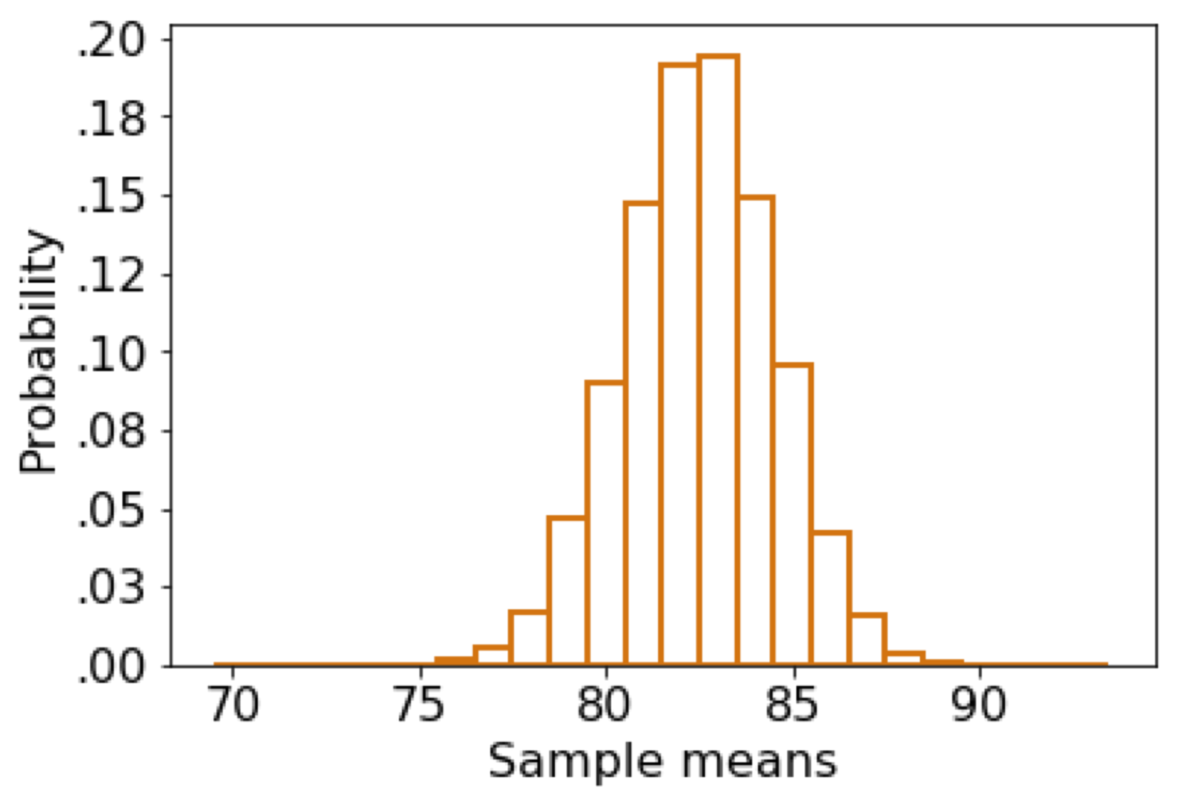


What is the probability that the mean of a subsample of 200 people is within the range 81 to 85?

Bootstrapped means

3. You now have a **distribution of your mean**

```
means = [84.7, 83.9,
         80.6, 79.8, 90.3,
         …, 85.2]
```



What is the bootstrapped standard error?

```
np.std(means)
```

Bootstrapped standard error: 1.99

Standard error via formula: $SE = \sqrt{S^2/n} = 1.99$

Standard error

1. Mean happiness:

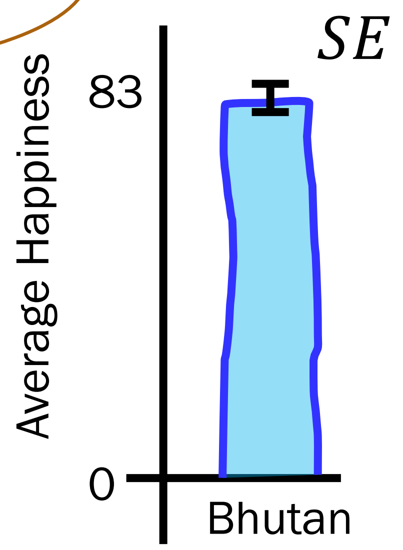
Claim: The average happiness of Bhutan is 83, with a standard error of 1.99.

this is our best estimate of μ

this is how close we are

Closed form: $SE = \sqrt{\dfrac{S^2}{n}}$

✅ Verified via bootstrap



Standard error

1. Mean happiness:

   Claim: The average happiness of Bhutan is 83, with a standard error of 1.99.

   Closed form: $SE = \sqrt{\frac{S^2}{n}}$

2. Variance of happiness:

   this is our best estimate of σ^2

   Claim: The variance of happiness of Bhutan is 793.

   Closed form: Not covered in CS109

   But how close are we?

   We can bootstrap for standard error, which is a statistic of a statistic.

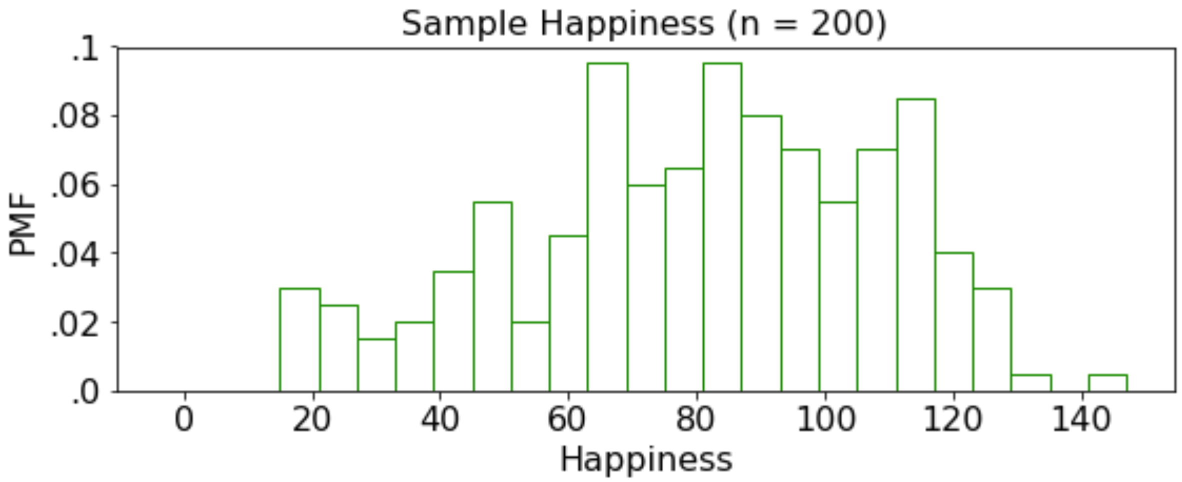
Bootstrapped variance

Bootstrap Algorithm (sample):

1. Estimate the **PMF** using the sample
2. Repeat **10,000** times:
   a. Resample **sample.size()** from PMF
   b. Recalculate the variance on the resample
3. You now have a **distribution of your** variance

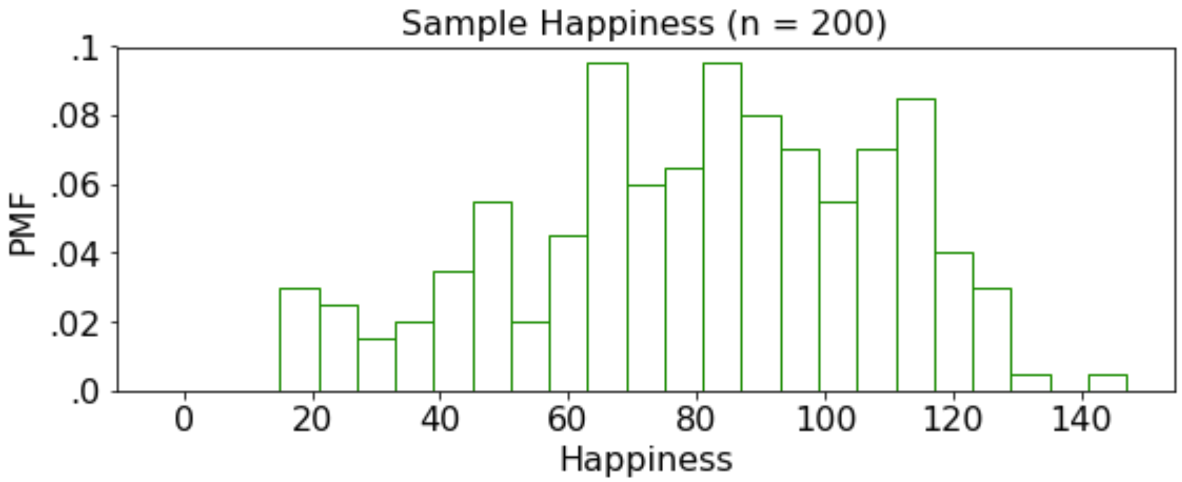
What is the distribution of your variance?

Bootstrapped means



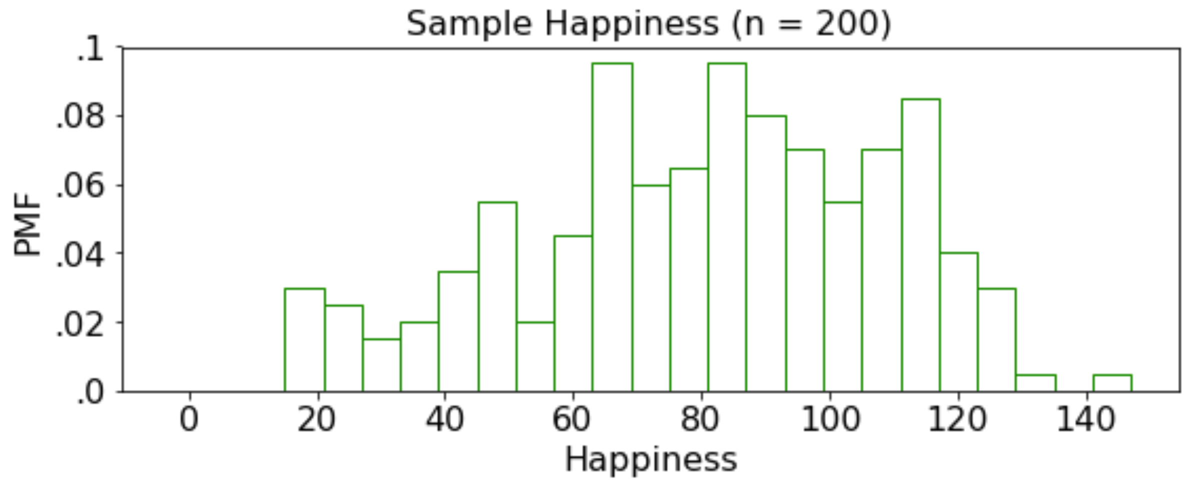
1. Estimate the **PMF** using the sample
2. Repeat **10,000** times:
    a. Resample **sample.size()** from PMF
    b. Recalculate the **variance** on the resample
3. You now have a **distribution of your variance**

Bootstrapped means

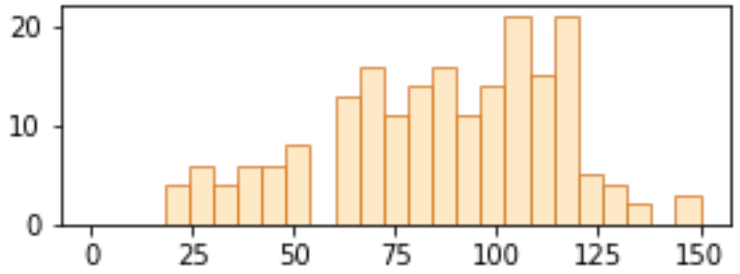


1. Estimate the PMF using the sample
2. Repeat **10,000** times:
    a. Resample **sample.size()** from PMF
    b. Recalculate the variance on the resample
3. You now have a distribution of your variance

Bootstrapped means

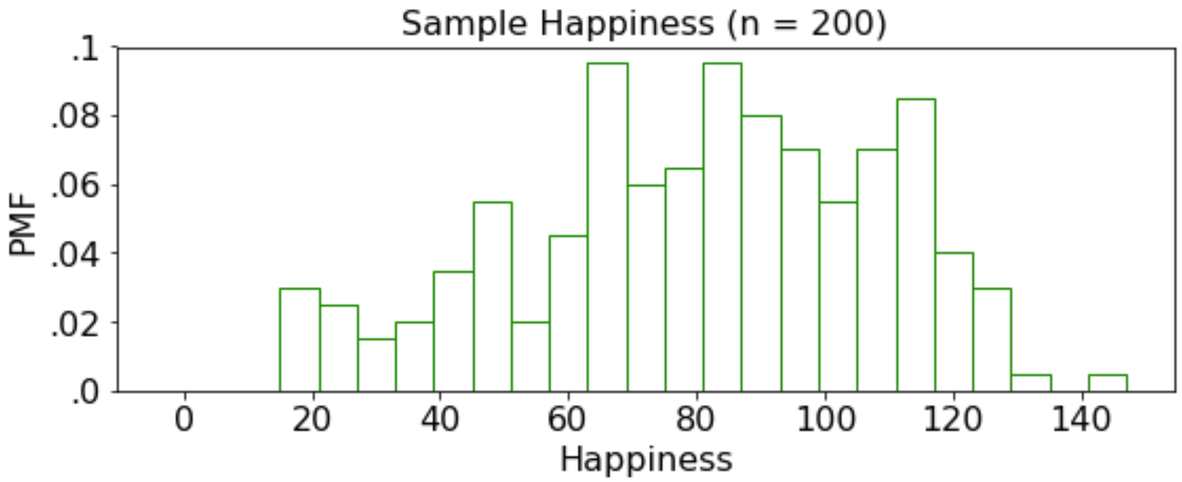


[52, 38, 98, 107, ..., 94]

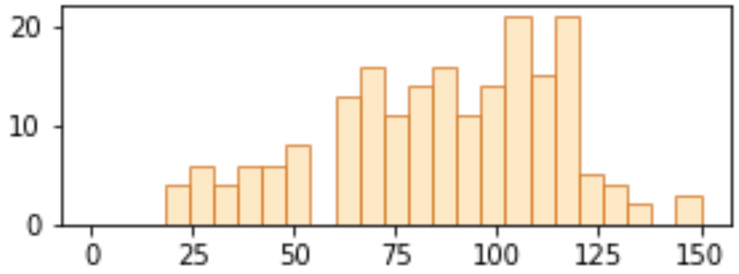


1. Estimate the **PMF** using the sample
2. Repeat **10,000** times:
   a. Resample **sample.size()** from PMF
   b. Recalculate the variance on the resample
3. You now have a distribution of your variance

Bootstrapped means



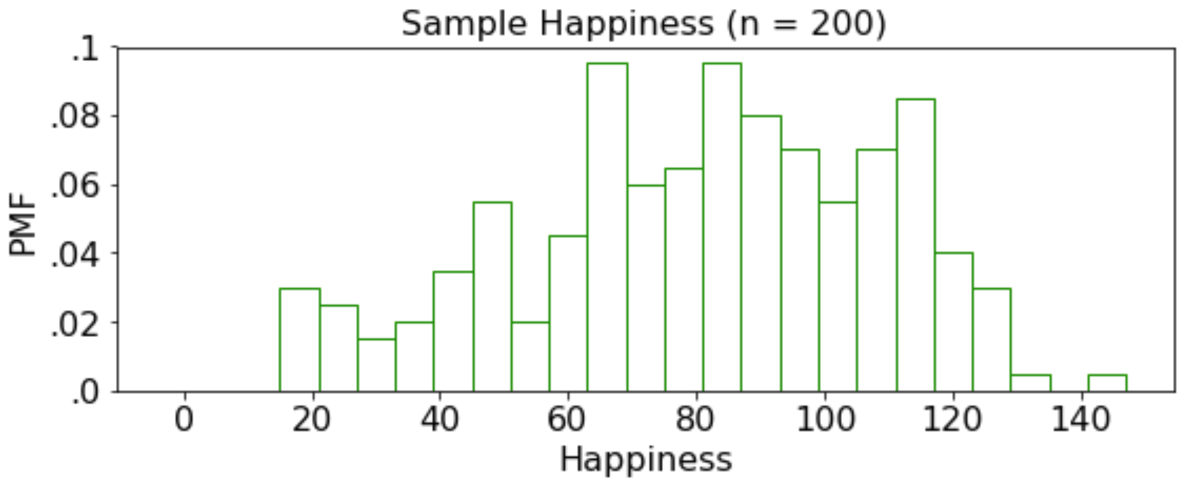
[52, 38, 98, 107, ..., 94]



1. Estimate the **PMF** using the sample
2. Repeat **10,000** times:
    a. Resample **sample.size()** from PMF
    b. Recalculate the variance on the resample
3. You now have a distribution of your variance

variances = [827.4]

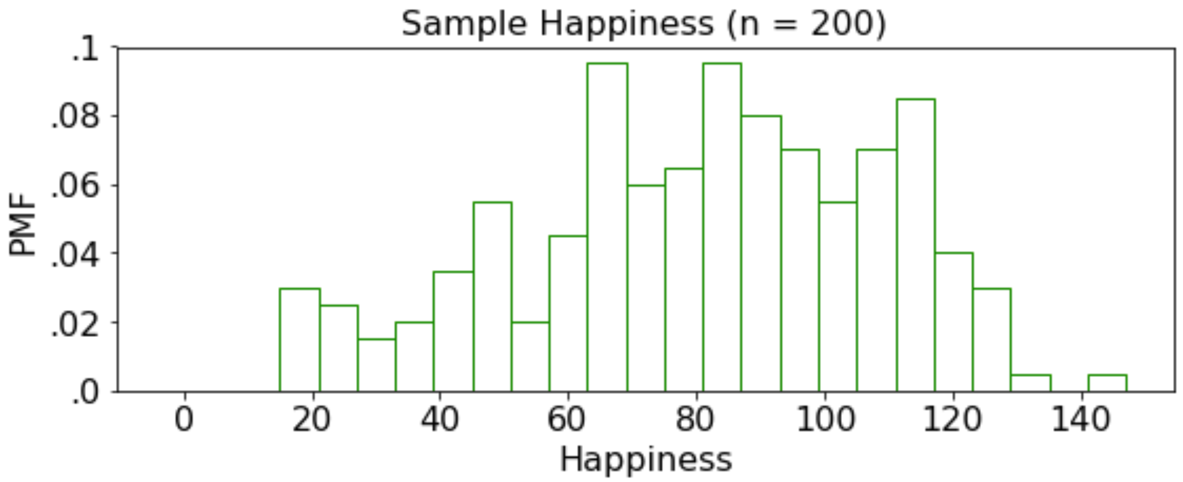
Bootstrapped means



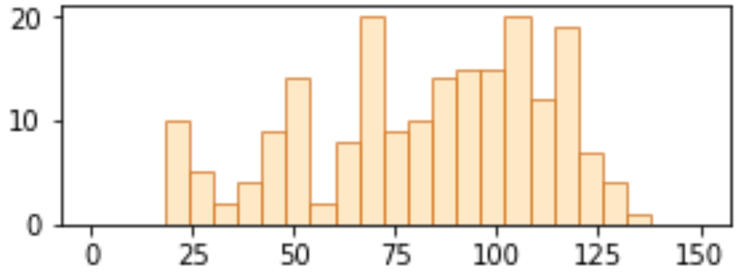
1. Estimate the PMF using the sample
2. Repeat **10,000** times:
   a. Resample **sample.size()** from PMF
   b. Recalculate the variance on the resample
3. You now have a distribution of your variance

variances = [827.4]

Bootstrapped means



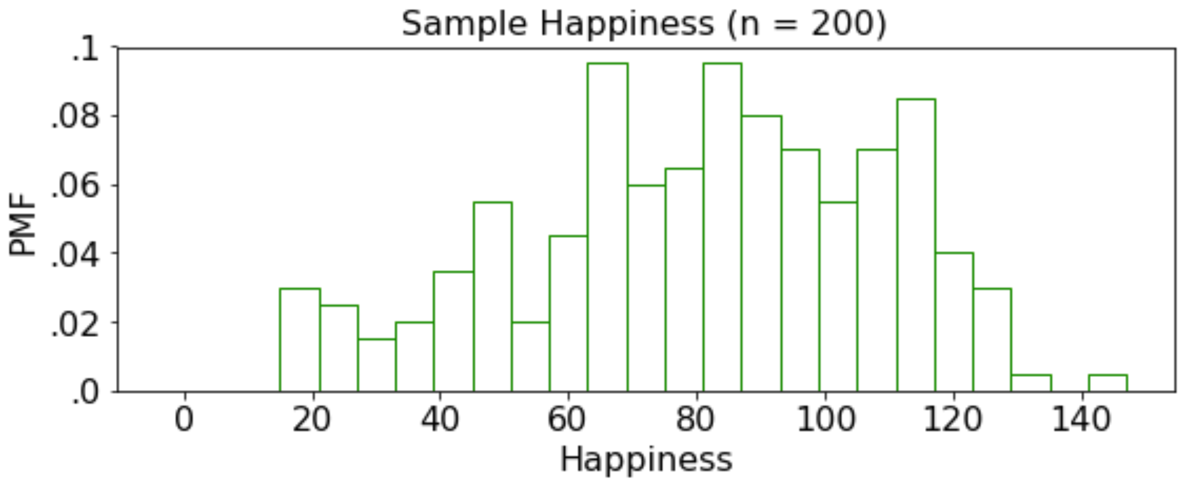
[116, 76, 132, 85, ..., 78]



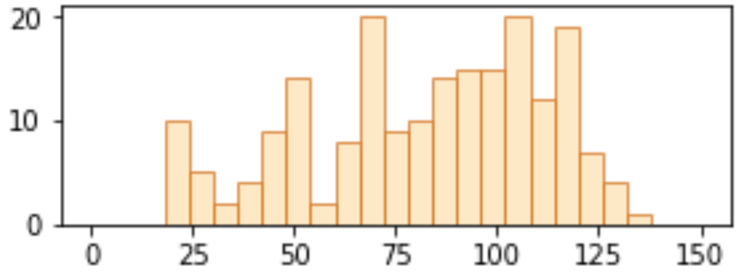
1. Estimate the **PMF** using the sample
2. Repeat **10,000** times:
   a. Resample **sample.size()** from PMF
   b. Recalculate the **variance** on the resample
3. You now have a distribution of your variance

```
variances = [827.4]
```

Bootstrapped means



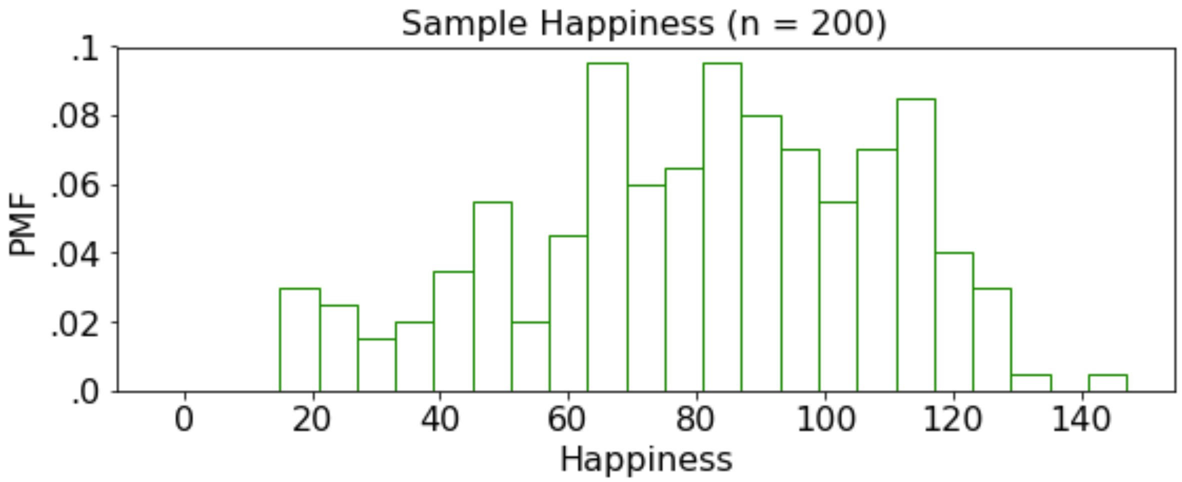
[116, 76, 132, 85, ..., 78]



1. Estimate the **PMF** using the sample
2. Repeat **10,000** times:
   a. Resample **sample.size()** from PMF
   b. Recalculate the variance on the resample
3. You now have a distribution of your variance

variances = [827.4, 846.1]

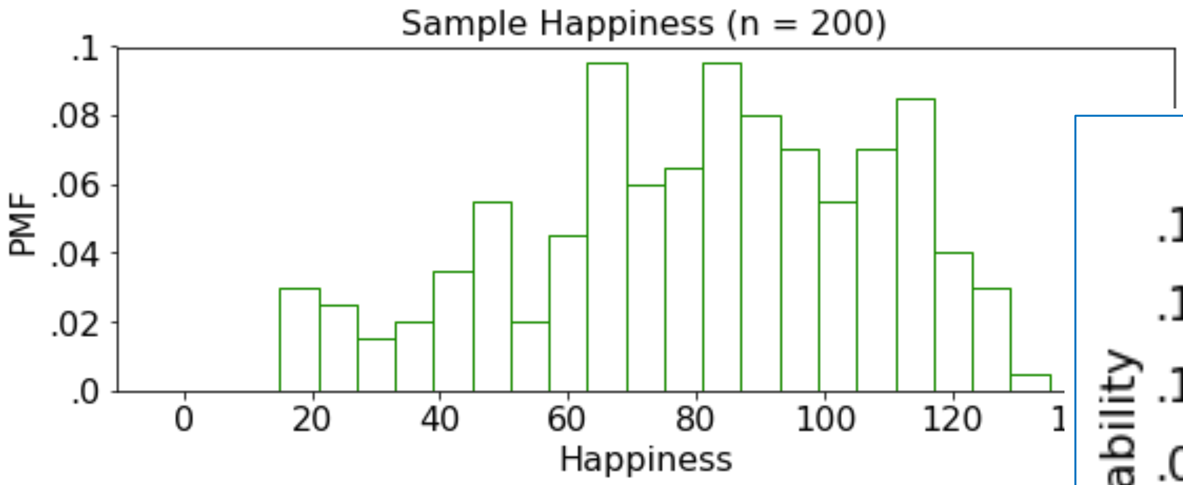
Bootstrapped means

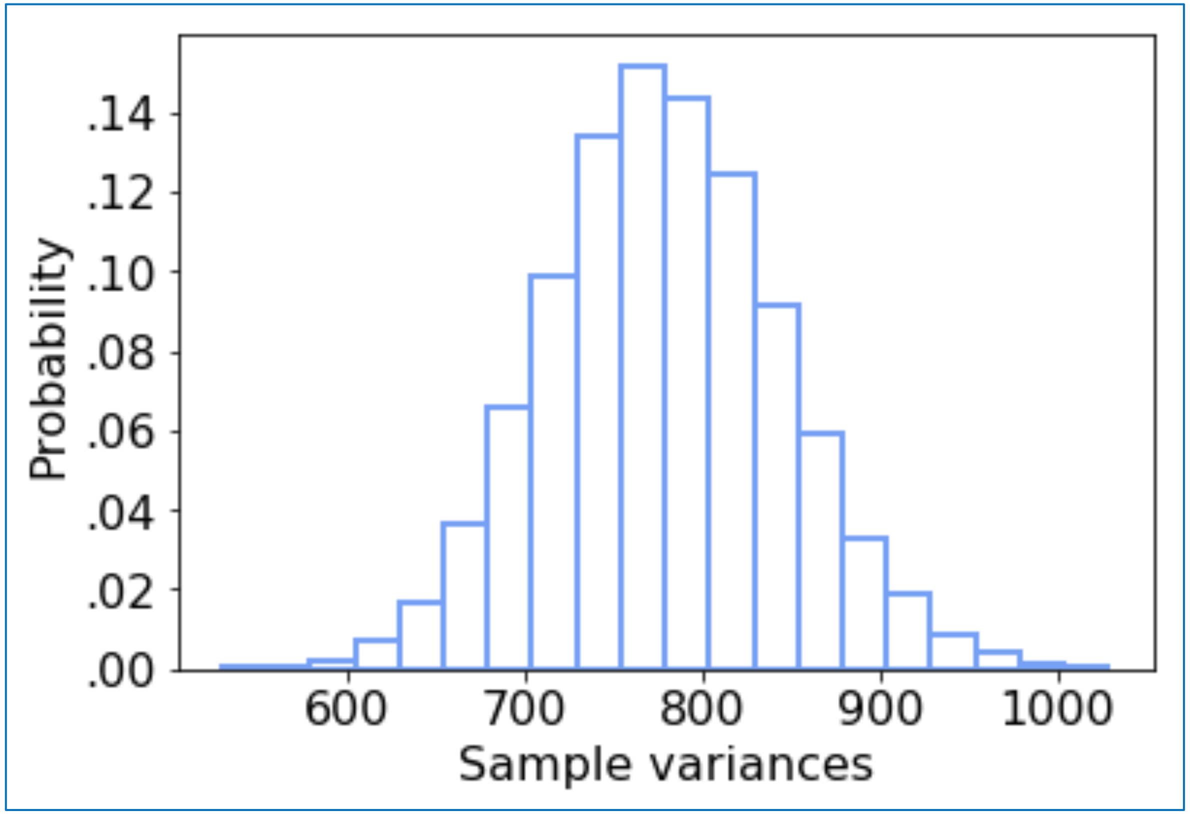


1. Estimate the **PMF** using the sample
2. Repeat **10,000** times:
    a. Resample **sample.size()** from PMF
    b. Recalculate the variance on the resample
3. You now have a distribution of your variance

variances = [827.4, 846.1]

Bootstrapped means





1. Estimate the **PMF** using the s
2. Repeat **10,000** times:
   a. Resample **sample.size()**
   b. Recalculate the **varian**
3. You now have a **distribution of your** variance

variances = [827.4, 846.1, 726.0, …, 860.7]

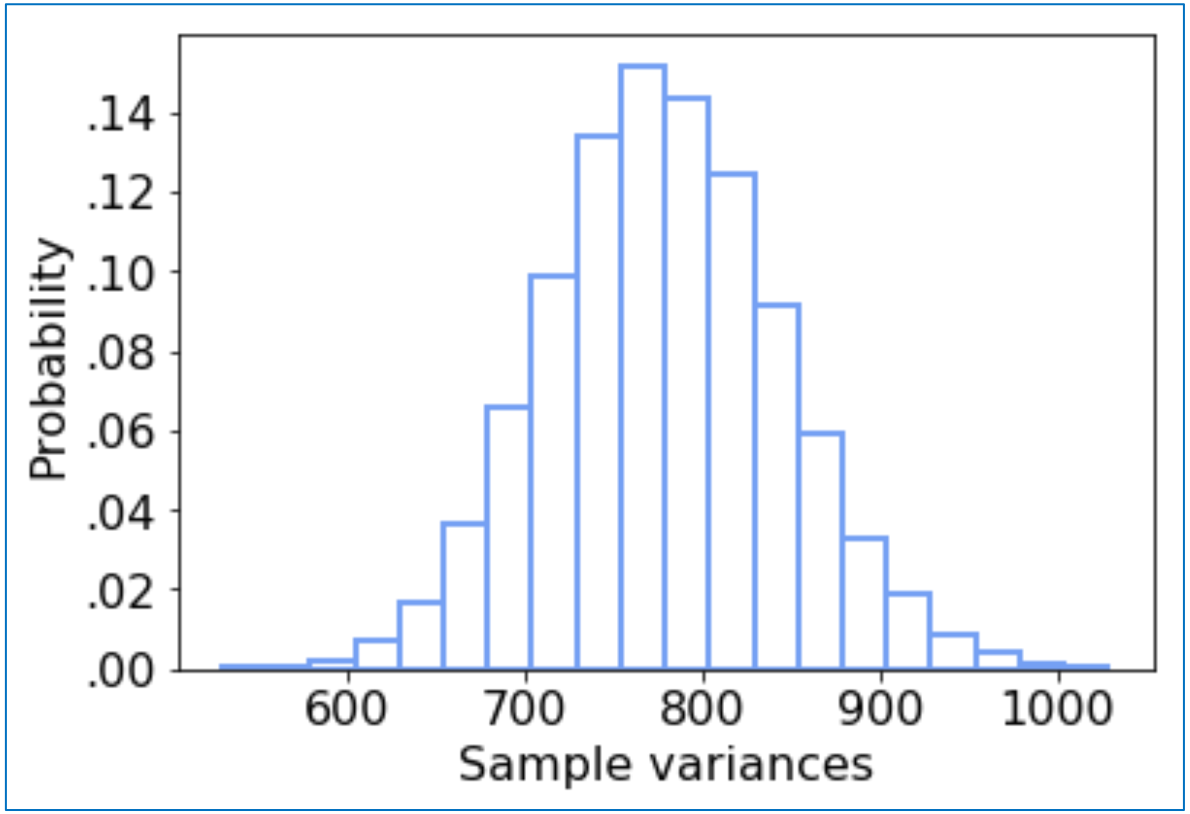
Bootstrapped means

3. You now have a **distribution of your variance**

```
variances = [827.4,
     846.1, 726.0, …,
     860.7]
```

What is the bootstrapped standard error?

```
np.std(variances)
```



Bootstrapped standard error: 66.16

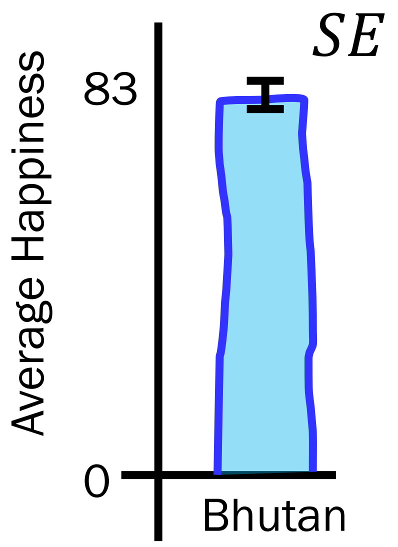
We can bootstrap for standard error of the sample variance.

Standard error

1. Mean happiness:

   Claim: The average happiness of Bhutan is 83, with a standard error of 1.99.

   Closed form: $SE = \sqrt{\frac{S^2}{n}}$



2. Variance of happiness:

   this is our best estimate of σ^2

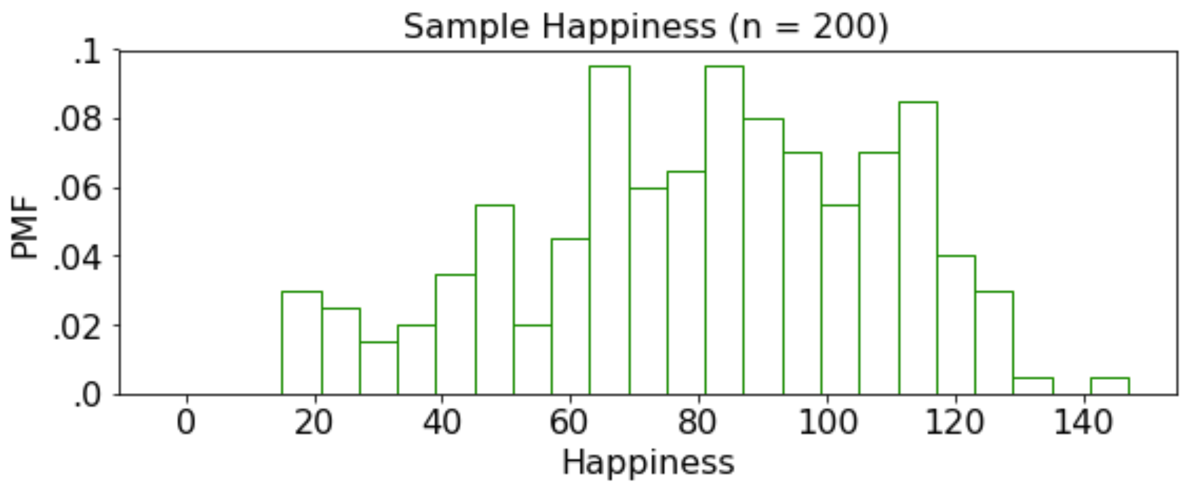
   Claim: The variance of happiness of Bhutan is 793, with a **bootstrapped standard error of 66.16**.

   this is how close we are, calculated by bootstrapping ✅

Algorithm in practice

?

1. Estimate the **PMF** using the sample
2. Repeat **10,000** times:
    a. Resample **sample.size()** from PMF
    b. Recalculate the **statistic** on the resample
3. You now have a **distribution of your statistic**

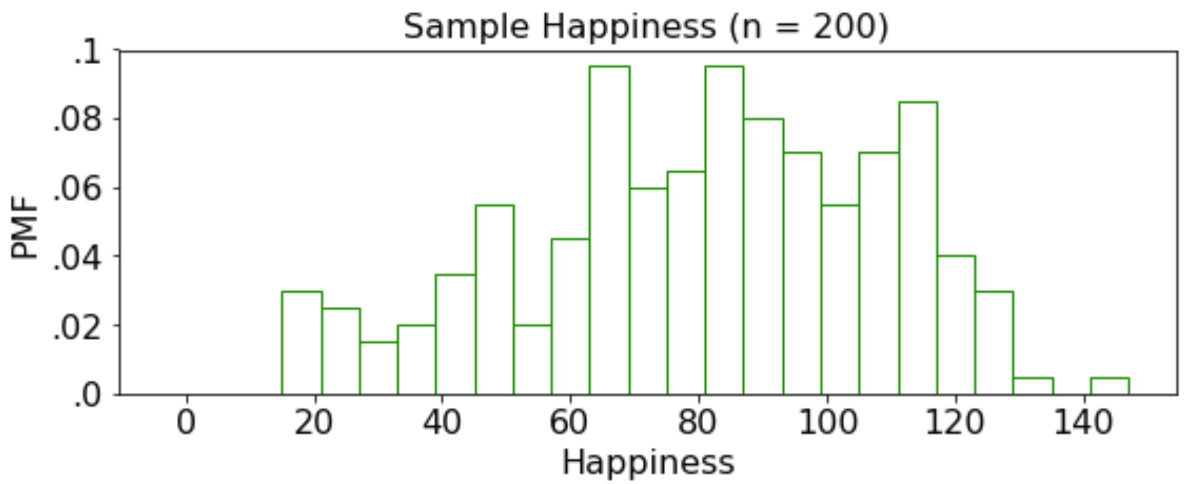


$$P(X = k) = \frac{\text{\# values in sample equal to } k}{n}$$

Algorithm in practice

1. Estimate the **PMF** using the sample
2. Repeat **10,000** times:
    a. Resample **sample.size()** from PMF
    b. Recalculate the **statistic** on the resample
3. You now have a **distribution of your statistic**

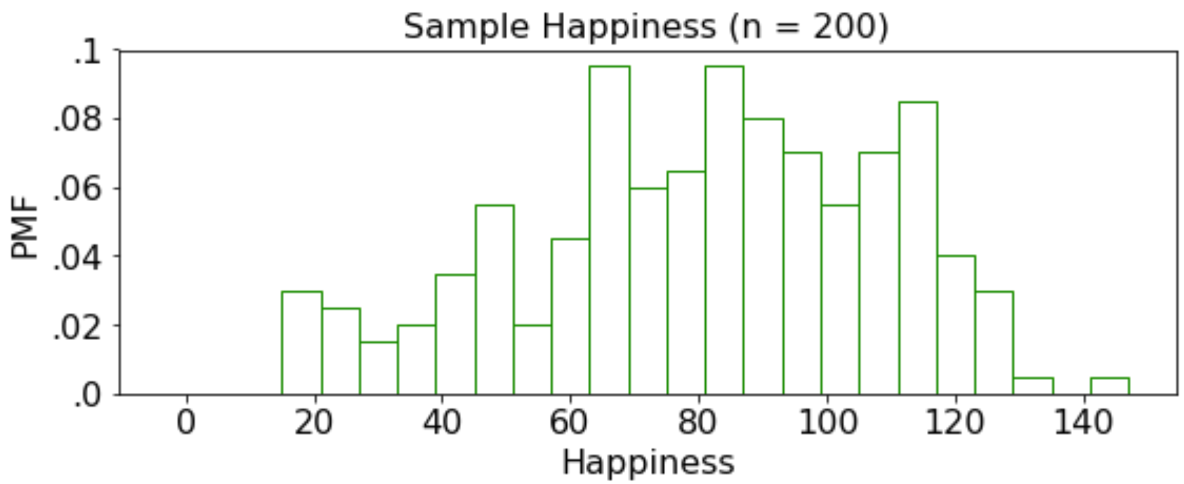
?



$$P(X = k) = \frac{\text{\# values in sample equal to } k}{n}$$

Algorithm in practice

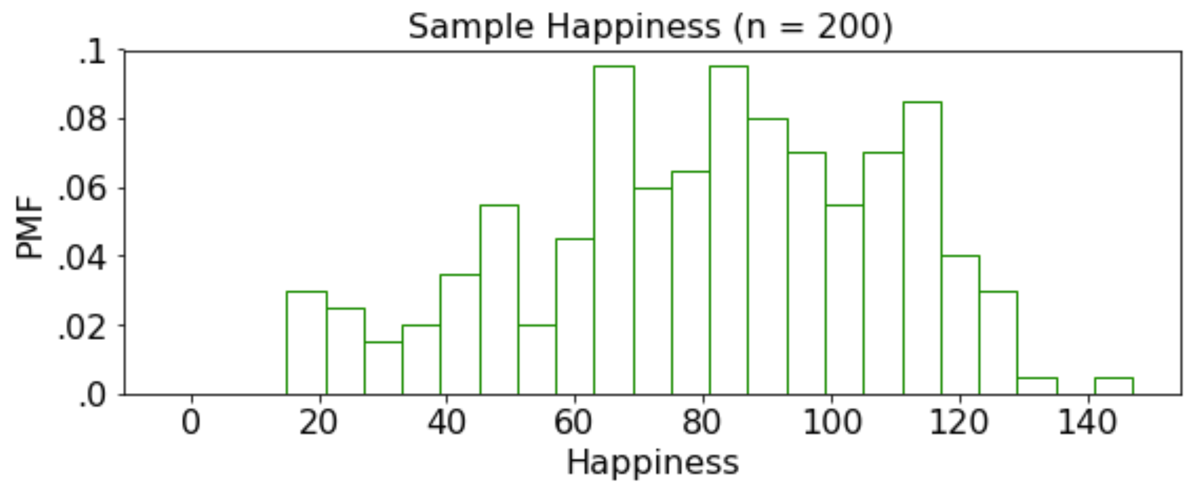
```
def resample(sample, n):
    # estimate the PMF using the sample
    # draw n new samples from the PMF
    return np.random.choice(sample,
                            n, replace=True)
```



$$P(X = k) = \frac{\text{\# values in sample equal to } k}{n}$$

Algorithm in practice

1. Estimate the **PMF** using the sample
2. Repeat **10,000** times:
   a. Resample **sample.size()** from PMF, **with replacement**
   b. Recalculate the **statistic** on the resample
3. You now have a **distribution of your statistic**



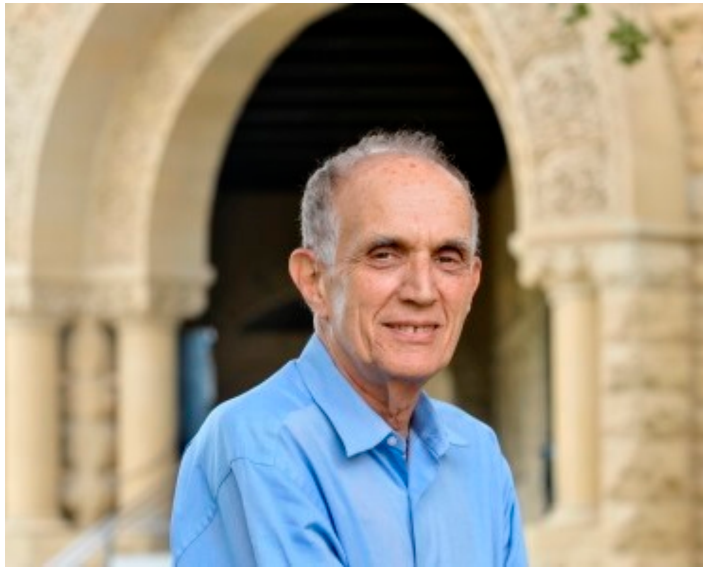
$$P(X = k) = \frac{\text{\# values in sample equal to } k}{n}$$

Questions?

To the code!

Bootstrap provides a way to calculate probabilities of statistics using code.

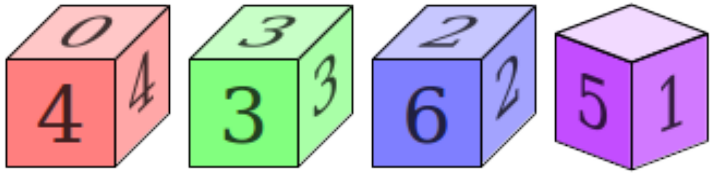
Bradley Efron



- Invented bootstrapping in 1979
- Still a professor at Stanford
- Won a National Science Medal

Efron's dice: 4 dice A, B, C, D such that

$$P(A > B) = P(B > C) = P(C > D) = P(D > A) = \frac{2}{3}$$



To the code!

Bootstrap provides a way to calculate probabilities of statistics using code.

Bootstrapping works for any statistic*

*as long as your sample is i.i.d. and the underlying distribution does not have a long tail

Today’s plan

Bootstrapping

- For a statistic
- For a p-value

Definition: Bayesian Networks

Inference:

1. Math
2. Rejection sampling (“joint” sampling)
3. Optional: Gibbs sampling (MCMC algorithm)

Null hypothesis test

Population 1
4.44
3.36
5.87
2.31
…
3.70

$\mu_1 = 3.1$

Population 2
4.44
3.36
5.87
2.31
…
3.70

$\mu_2 = 2.4$

Claim: Population 1 and Population 2 have a 0.7 difference of means.

Null hypothesis test

Nepal Happiness
4.44
3.36
5.87
2.31
…
3.70

$\mu_1 = 3.1$

Bhutan Happiness
4.44
3.36
5.87
2.31
…
3.70

$\mu_2 = 2.4$

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

Null hypothesis test

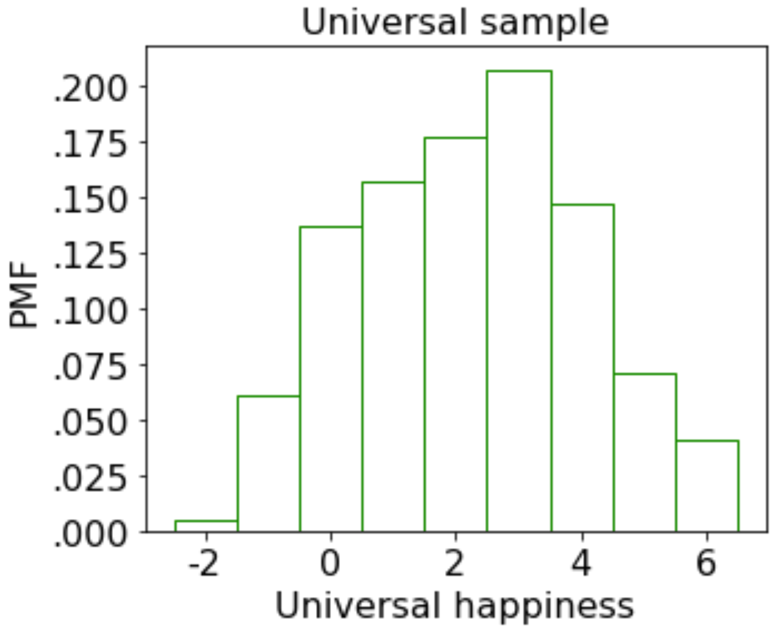
def null hypothesis – Even if there is no pattern (i.e., the two samples are from identical distributions), your claim might have arisen by chance.

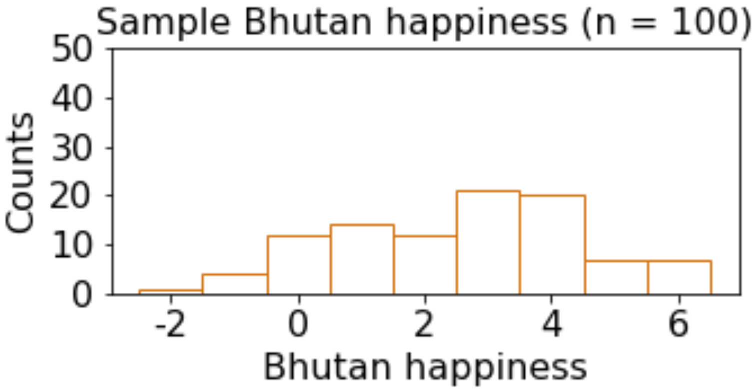
def p-value – What is the probability that, under the null hypothesis, the observed difference occurs?

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

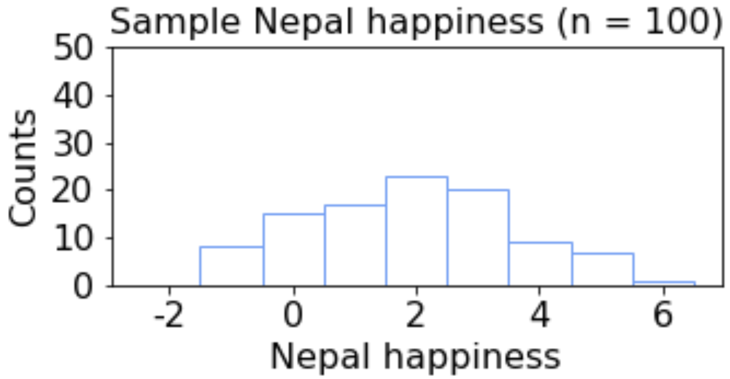
Universal sample

(this is what the null hypothesis assumes)





$\mu_1 = 3.1$



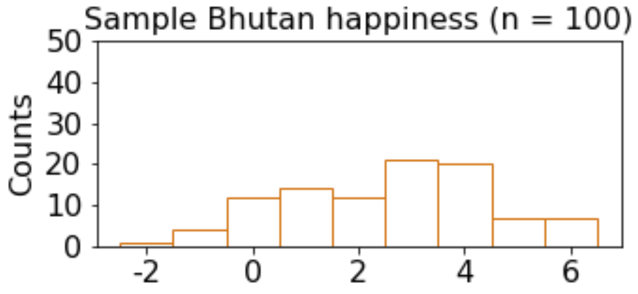
$\mu_2 = 2.4$

Want p-value: probability $|\mu_1 - \mu_2| = |3.1 - 2.4|$ happens under null hypothesis

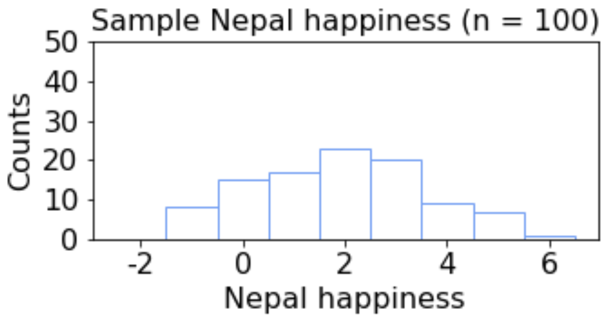
Bootstrap for p-values

1. Create a universal sample using your two samples

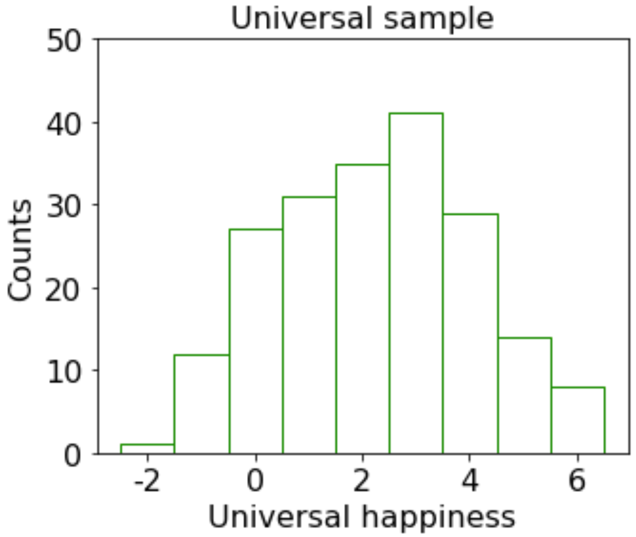
Recreate the null hypothesis



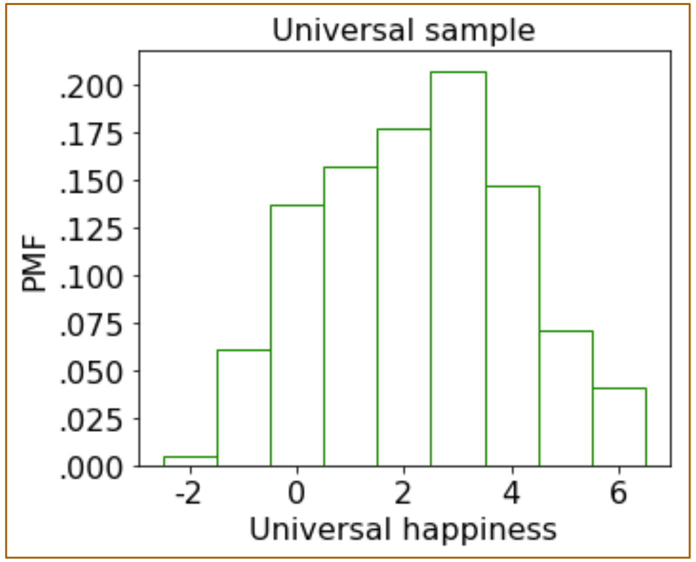
+



=



≈



Bootstrap for p-values

1. Create a universal sample using your two samples
2. Repeat **10,000** times:
   a. Resample **both samples**
   b. Recalculate the **mean difference** between the resamples
3. **p-value** = $\frac{\text{\# (mean diffs >= observed diff)}}{n}$

Probability that observed difference arose by chance

Bootstrap for p-values

```
def pvalue_boot(bhutan_sample, nepal_sample):
    N = size of the bhutan_sample
    M = size of the nepal_sample
    observed_diff = |mean of bhutan_sample – mean of nepal_sample|

    uni_sample = combine bhutan_sample and nepal_sample
    count = 0

    repeat 10,000 times:
        bhutan_resample = draw N resamples from the uni_sample
        nepal_resample = draw M resamples from the uni_sample
        muBhutan = sample mean of the bhutan_resample
        muNepal = sample mean of the nepal_resample
        diff = |muNepal - muBhutan|
        if diff >= observed_diff:
            count += 1

pValue = count / 10,000
```

Bootstrap for p-values

1. Create a universal sample using your two samples

```
def pvalue_boot(bhutan_sample, nepal_sample):
    N = size of the bhutan_sample
    M = size of the nepal_sample
    observed_diff = |mean of bhutan_sample - mean of nepal_sample|

    uni_sample = combine bhutan_sample and nepal_sample
    count = 0

    repeat 10,000 times:
        bhutan_resample = draw N resamples from the uni_sample
        nepal_resample = draw M resamples from the uni_sample
        muBhutan = sample mean of the bhutan_resample
        muNepal = sample mean of the nepal_resample
        diff = |muNepal - muBhutan|
        if diff >= observed_diff:
            count += 1

pValue = count / 10,000
```

Bootstrap for p-values

2. a. Resample **both** samples

```
def pvalue_boot(bhutan_sample, nepal_sample):
    N = size of the bhutan_sample
    M = size of the nepal_sample
    observed_diff = |mean of bhutan_sample - mean of nepal_sample|

    uni_sample = combine bhutan_sample and nepal_sample
    count = 0

    repeat 10,000 times:
        bhutan_resample = draw N resamples from the uni_sample
        nepal_resample = draw M resamples from the uni_sample
        muBhutan = sample mean of the bhutan_resample
        muNepal = sample mean of the nepal_resample
        diff = |muNepal - muBhutan|
        if diff >= observed_diff:
            count += 1

pValue = count / 10,000
```

Bootstrap for p-values

2. b. Recalculate the **mean difference** b/t resamples

```
def pvalue_boot(bhutan_sample, nepal_sample):
    N = size of the bhutan_sample
    M = size of the nepal_sample
    observed_diff = |mean of bhutan_sample – mean of nepal_sample|

    uni_sample = combine bhutan_sample and nepal_sample
    count = 0

    repeat 10,000 times:
        bhutan_resample = draw N resamples from the uni_sample
        nepal_resample = draw M resamples from the uni_sample
        muBhutan = sample mean of the bhutan_resample
        muNepal = sample mean of the nepal_resample
        diff = |muNepal - muBhutan|
        if diff >= observed_diff:
            count += 1

pValue = count / 10,000
```

Bootstrap for p-values

3. p-value =
    # (mean diffs > observed diff)
    ---
    n

```
def pvalue_boot(bhutan_sample, nepal_sample):
    N = size of the bhutan_sample
    M = size of the nepal_sample
    observed_diff = |mean of bhutan_sample – mean of nepal_sample|

    uni_sample = combine bhutan_sample and nepal_sample
    count = 0

    repeat 10,000 times:
        bhutan_resample = draw N resamples from the uni_sample
        nepal_resample = draw M resamples from the uni_sample
        muBhutan = sample mean of the bhutan_resample
        muNepal = sample mean of the nepal_resample
        diff = |muNepal - muBhutan|
        if diff >= observed_diff:
            count += 1

pValue = count / 10,000
```

Bootstrap for p-values

```
def pvalue_boot(bhutan_sample, nepal_sample):
    N = size of the bhutan_sample
    M = size of the nepal_sample
    observed_diff = |mean of bhutan_sample – mean of nepal_sample|

    uni_sample = combine bhutan_sample and nepal_sample
    count = 0
                                                  with replacement!
    repeat 10,000 times:
        bhutan_resample = draw N resamples from the uni_sample
        nepal_resample = draw M resamples from the uni_sample
        muBhutan = sample mean of the bhutan_resample
        muNepal = sample mean of the nepal_resample
        diff = |muNepal - muBhutan|
        if diff >= observed_diff:
            count += 1

pValue = count / 10,000
```

Bootstrap

Let’s try it!

Null hypothesis test

Nepal Happiness
4.44
3.36
5.87
2.31
…
3.70

Bhutan Happiness
4.44
3.36
5.87
2.31
…
3.70

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

Claim: The happiness of Nepal and Bhutan are from different distributions with a 0.7 difference of means ($p < 0.01$).

Questions?

Break for jokes/ announcements

Announcements

Weekly concept checks

Due: every Tuesday, 1pm
Selected answers: now on website!

Problem Set 5

Released: later today
Due: Friday 11/15
Covers: Up to Lecture Notes #20

Late day reminder: No late days permitted past last day of the quarter, 12/7

A small change in gears

Bootstrapping – Use code to compute statistics when you only have data, not the underlying distribution.

What if you have the underlying distribution of joint random variables (via an expert), but finding closed forms of joint probabilities is intractable?

Today's plan

Bootstrapping

- For a statistic
- For a p-value

Definition: Bayesian Networks

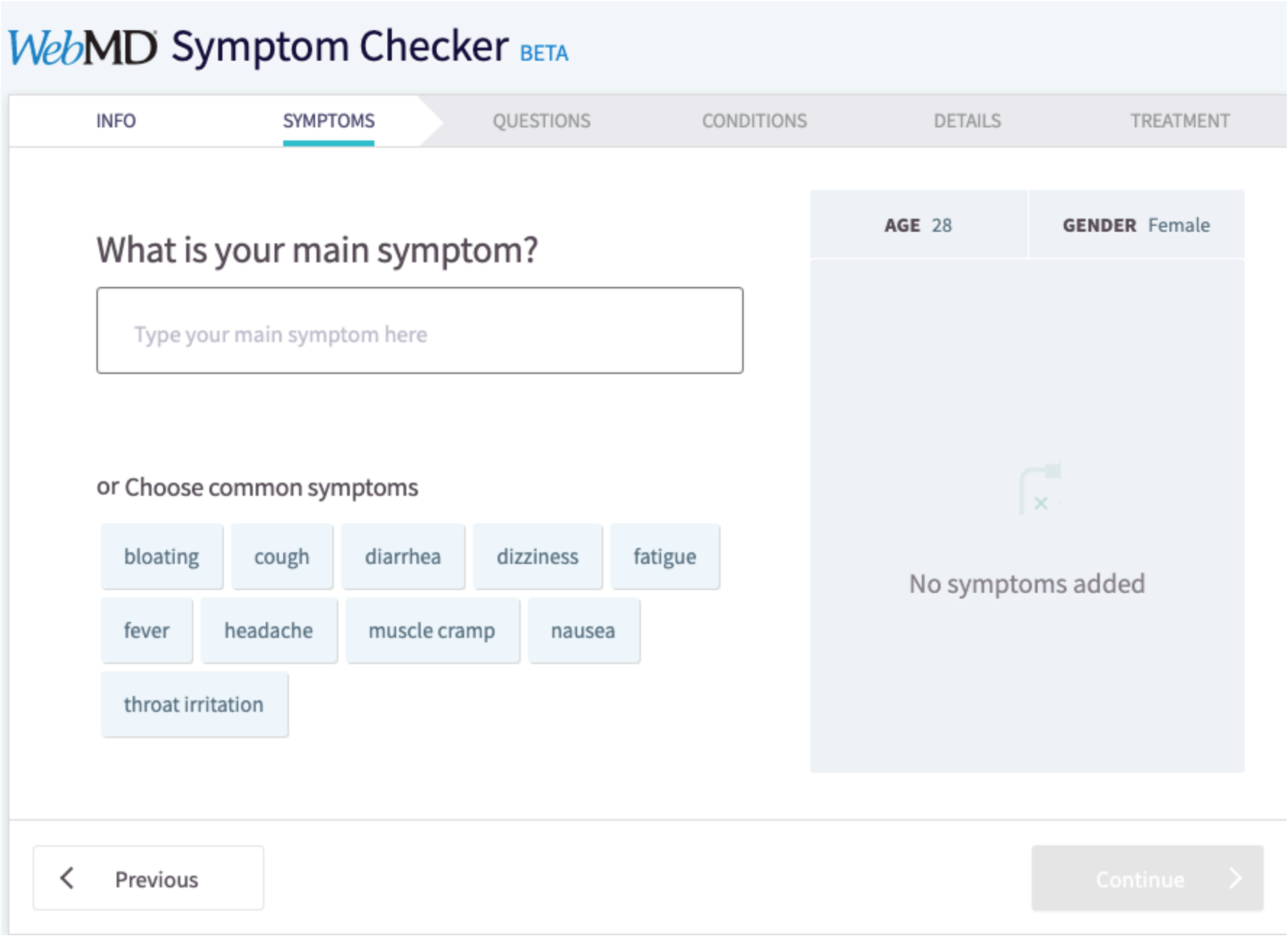
Inference:

1. Math
2. Rejection sampling ("joint" sampling)
3. Optional: Gibbs sampling (MCMC algorithm)

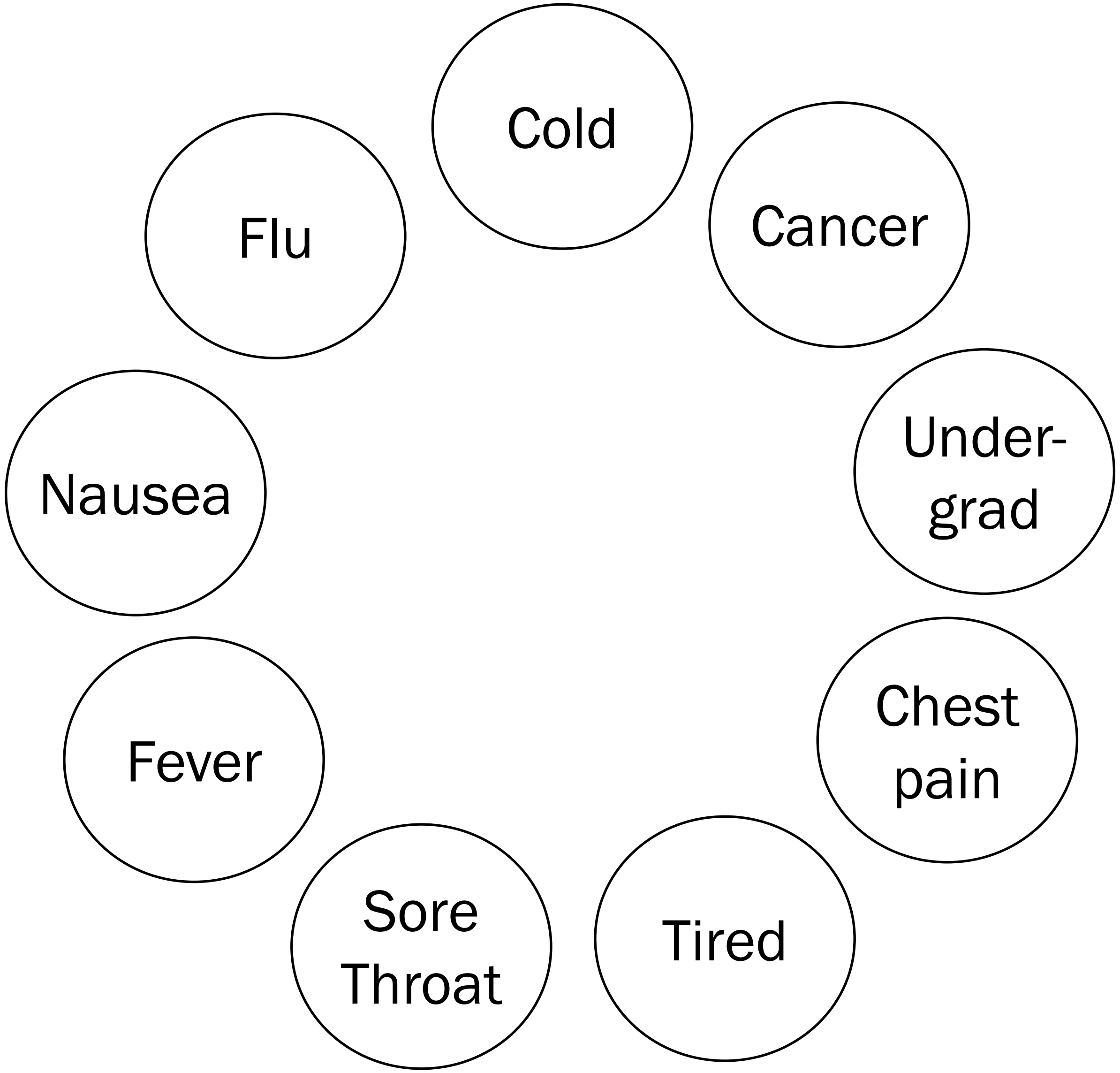
Inference

WebMD®

Inference


WebMD Symptom Checker BETA
INFO
SYMPTOMS
QUESTIONS
CONDITIONS
DETAILS
TREATMENT
What is your main symptom?
Type your main symptom here
or Choose common symptoms
bloating
cough
diarrhea
dizziness
fatigue
fever
headache
muscle cramp
nausea
throat irritation
AGE 28
GENDER Female
No symptoms added
Previous
Continue

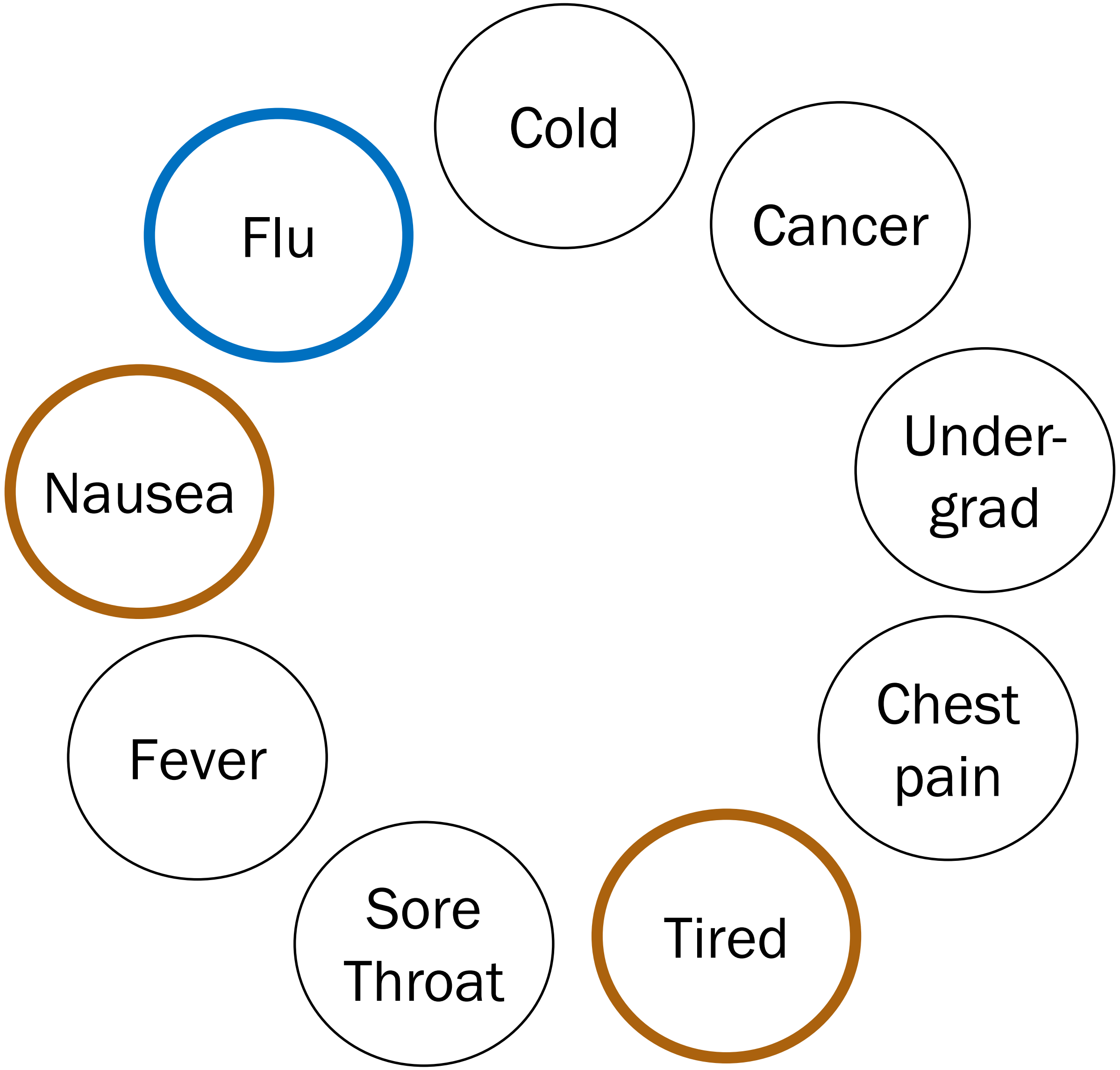
Inference



General inference question:

Given the values of some random variables, what is the conditional distribution of some other random variables?

Inference

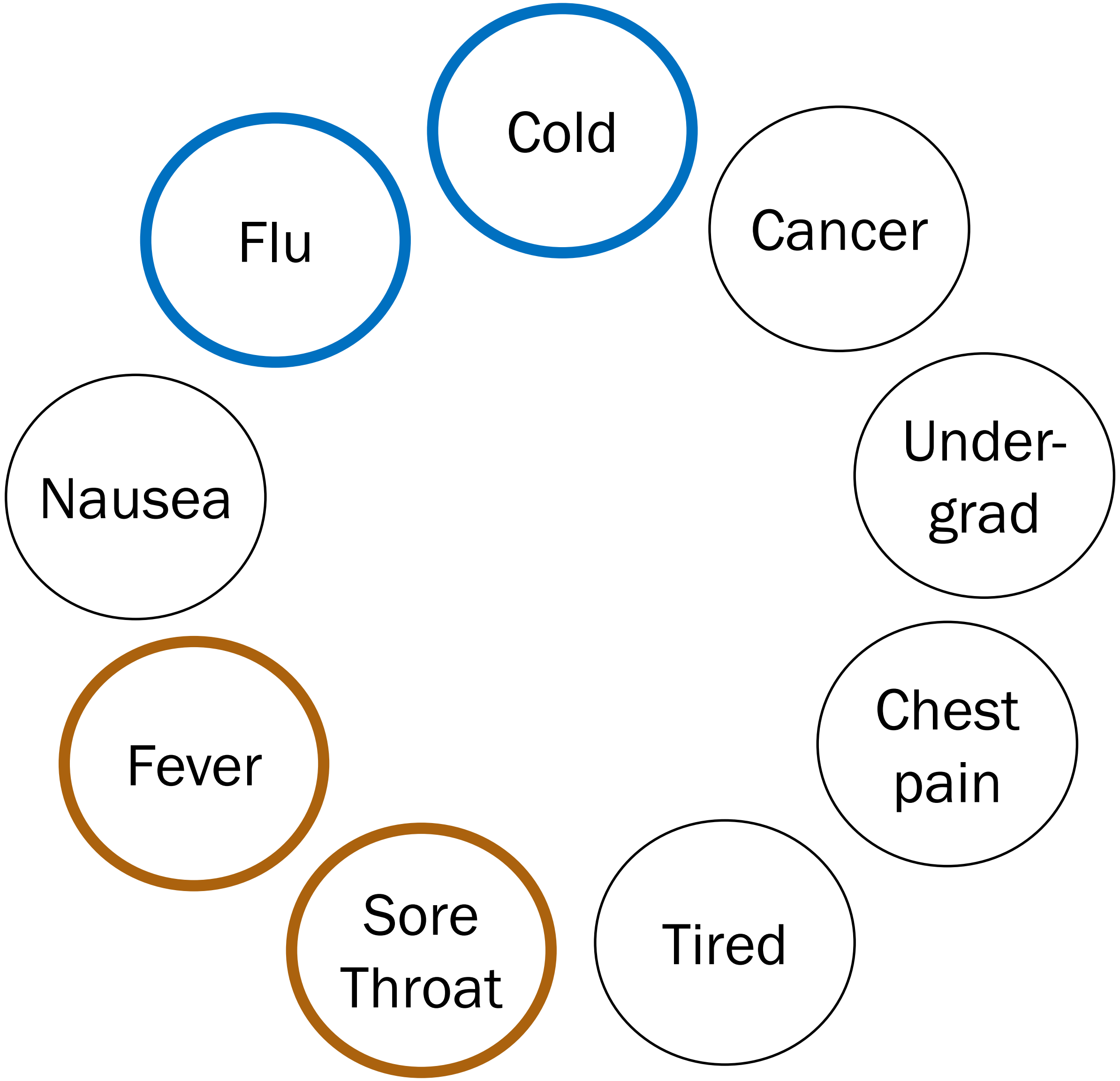


<u>One inference question</u>:

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

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

Inference

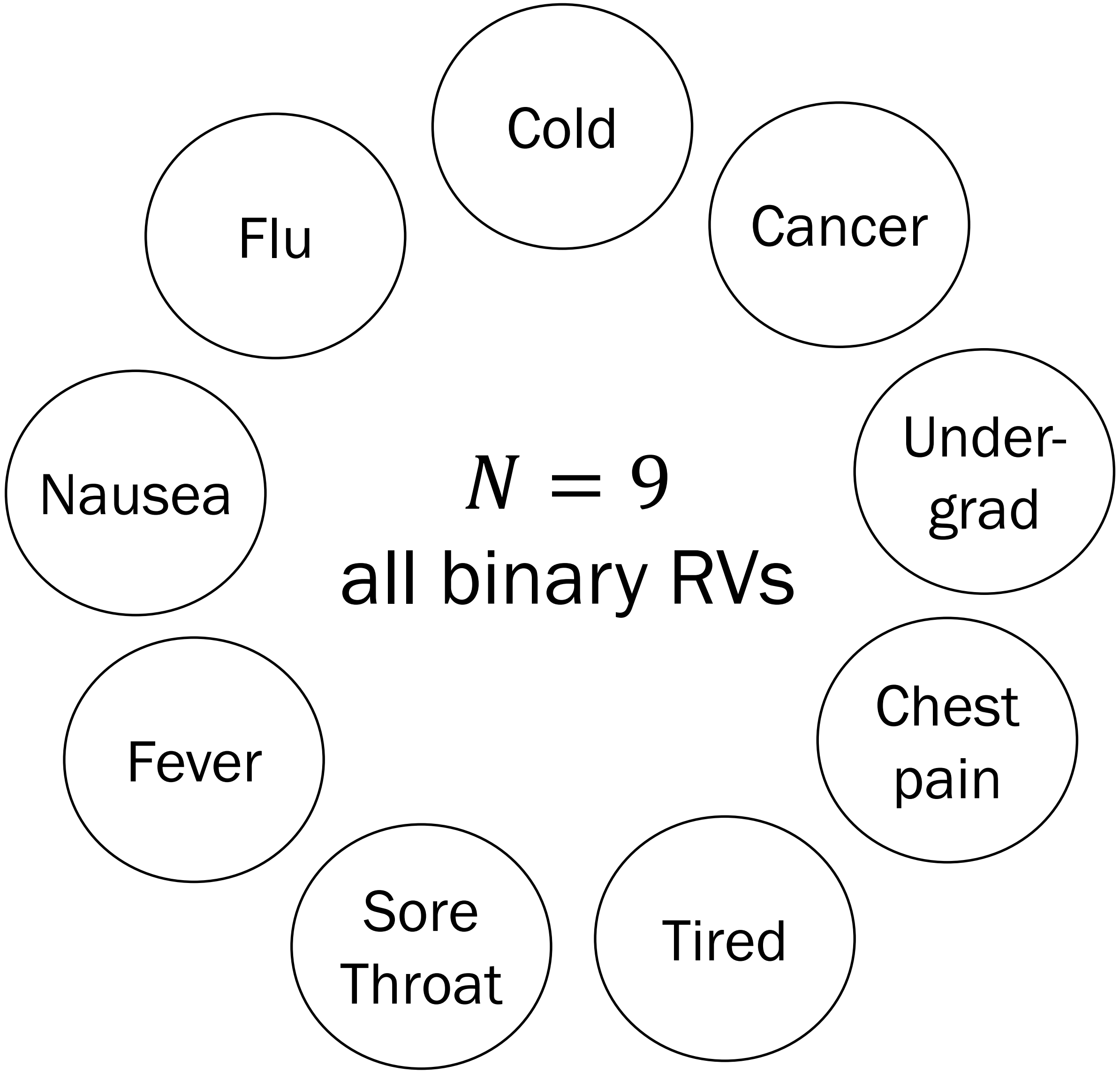


<u>Another inference question</u>:

$$P(C_o = 1, F_{lu} = 1 | S = 0, F_e = 0)$$

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

Inference



$N = 9$

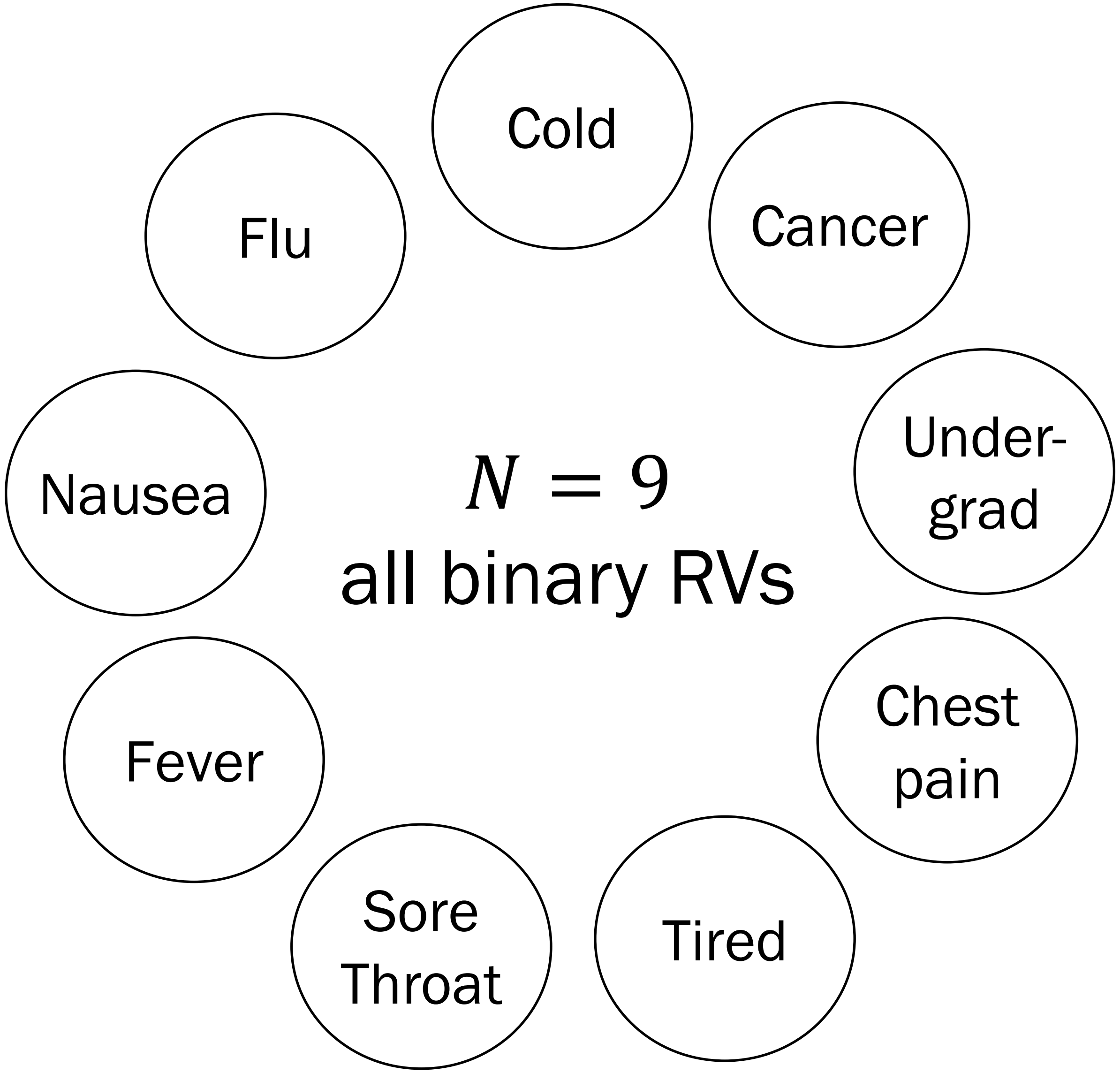
all binary RVs

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}
B. N^2
C. 2^N
D. None/other/don't know

Inference



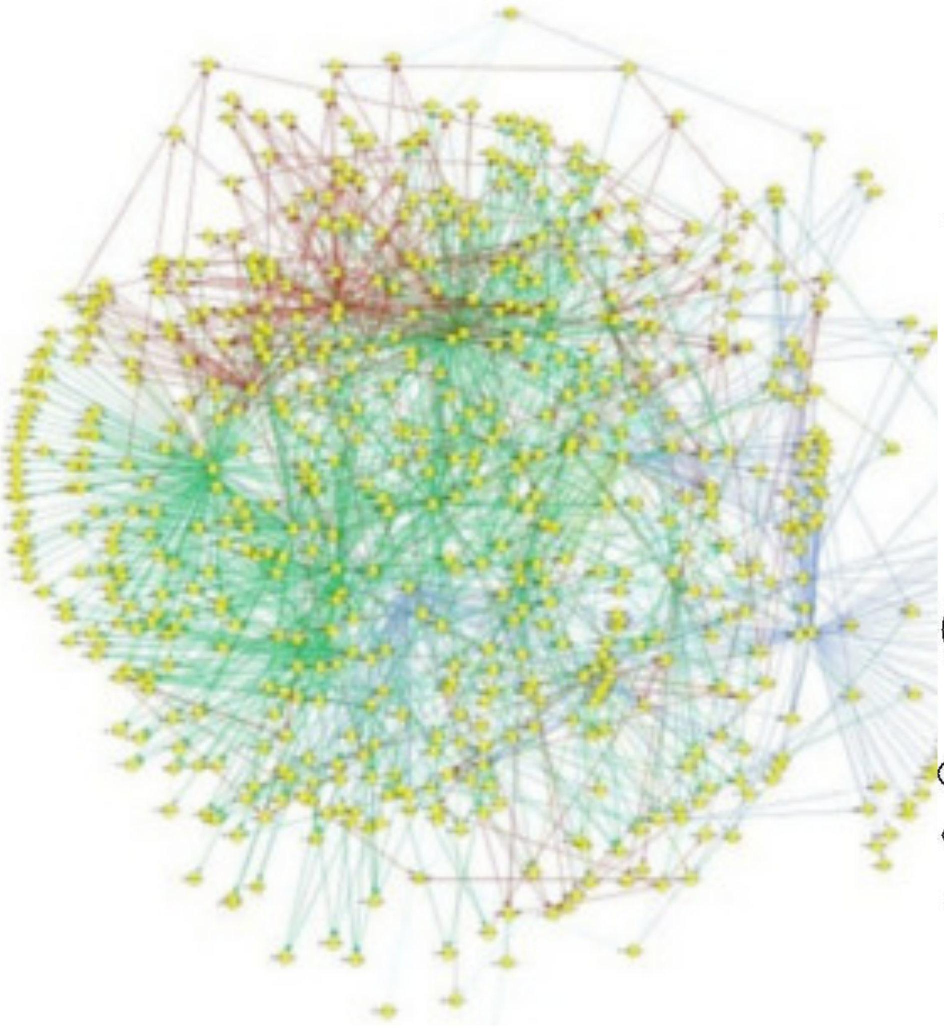
If we knew the **joint distribution**, we can answer all probabilistic inference questions.

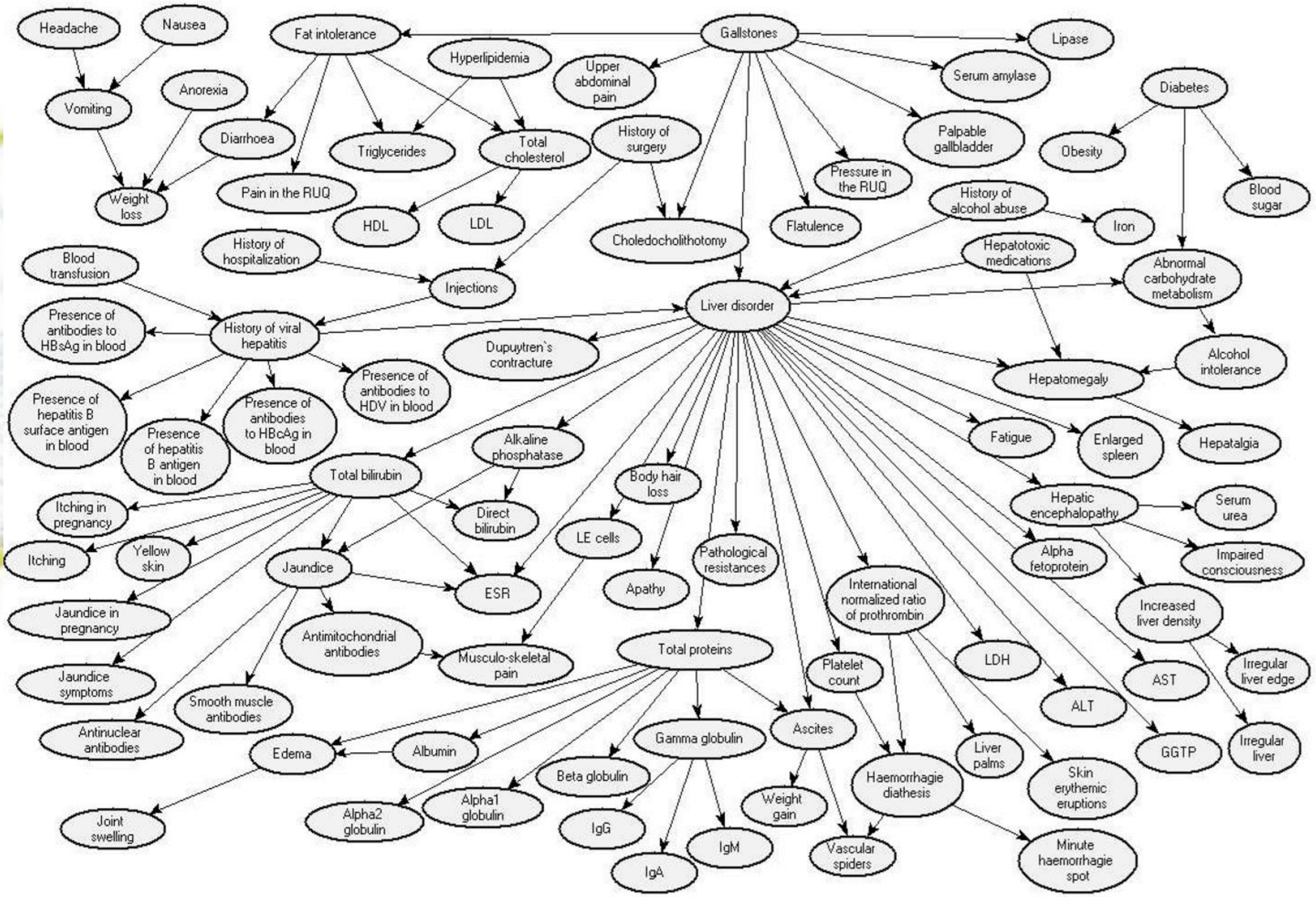
What is the size of the joint probability table?

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

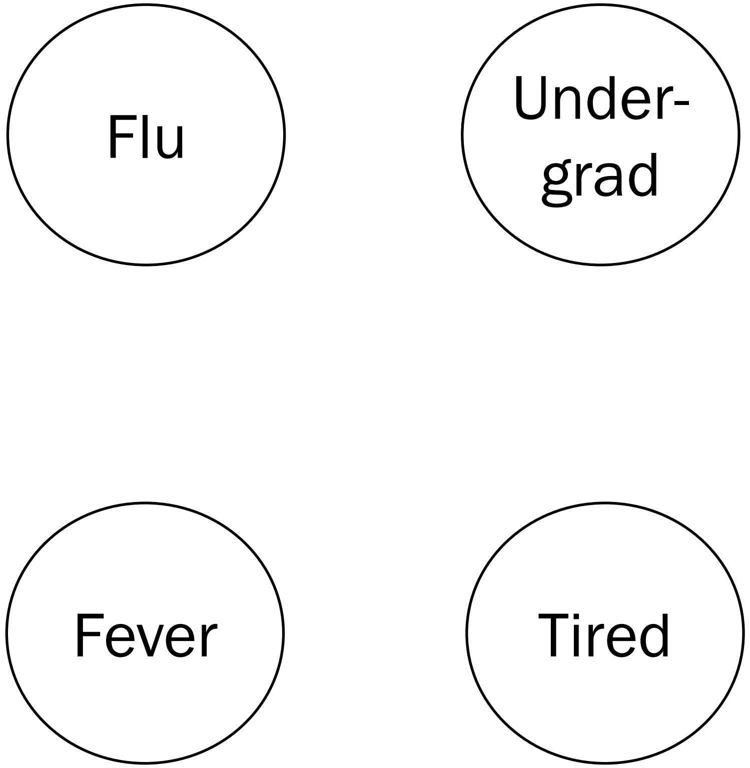
Naively specifying a joint is often intractable.

N can be large…





A simpler WebMD



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

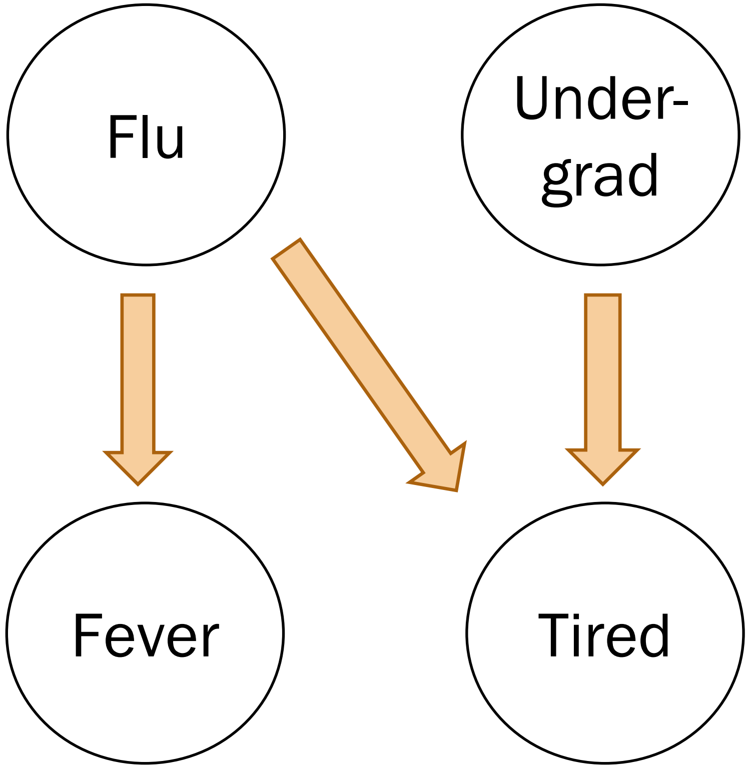
$$P(F_{lu} = a, F_{ev} = b, U = c, T = d)$$

What would a Stanford flu expert do?

Describe the joint distribution using causality!

A Bayesian Network

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



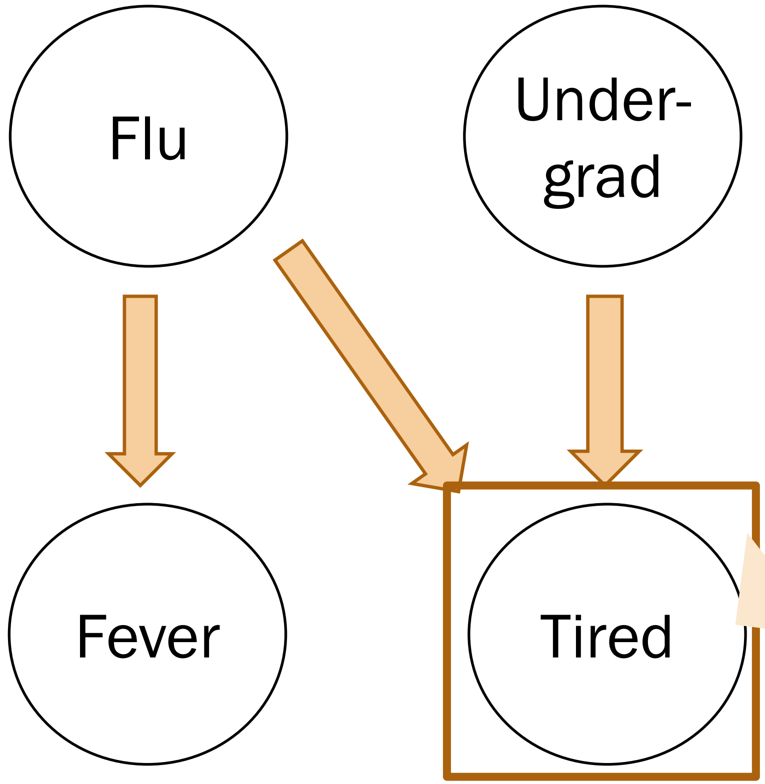
What would a Stanford flu expert do?

1. Describe the joint distribution using causality

A Bayesian Network

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

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



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

What would a Stanford flu expert do?

1. Describe the joint distribution using causality
2. Provide P(values|parents) for each random variable

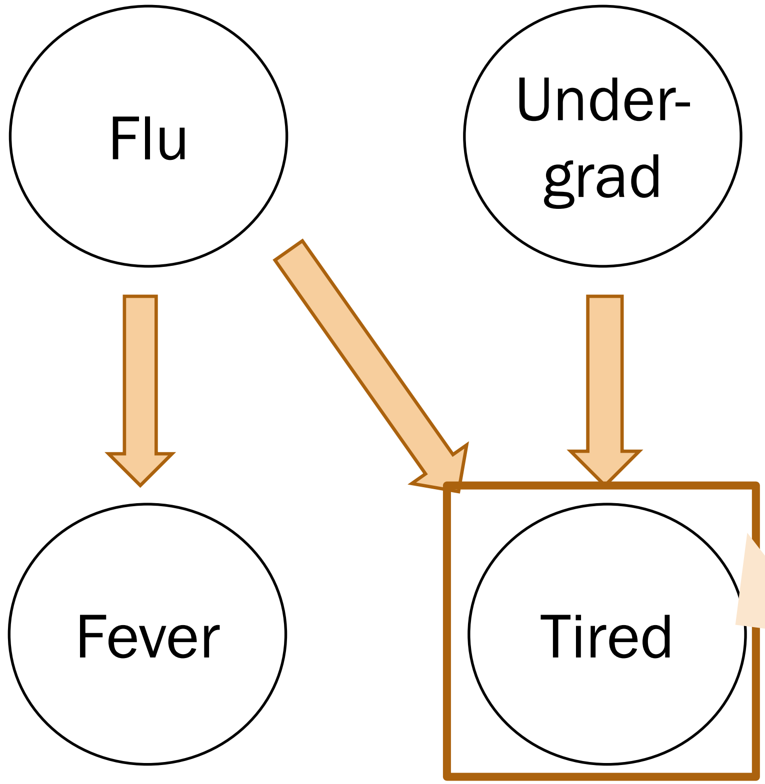
What conditional probabilities should our expert specify? (select all that apply)

A. $P(T = 1 | F_{lu} = 0, U = 0)$
B. $P(T = 1 | F_{lu} = 0, U = 1)$
C. $P(T = 1 | F_{lu} = 1, U = 0)$
D. $P(T = 1 | F_{lu} = 1, U = 1)$
E. $P(T = 1 | F_{lu} = 0)$
F. $P(T = 1 | F_{lu} = 1)$
G. $P(T = 0 | F_{lu} = 0, U = 0)$
H. $P(T = 0 | F_{lu} = 0, U = 1)$
I. $P(T = 0 | F_{lu} = 1, U = 0)$
J. $P(T = 0 | F_{lu} = 1, U = 1)$
K. $P(T = 1 | U = 0)$
L. $P(T = 1 | U = 1)$

A Bayesian Network

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

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



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

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

What would a Stanford flu expert do?

1. Describe the joint distribution using causality
2. Provide P(values|parents) for each random variable

What conditional probabilities should our expert specify? (select all that apply)

complements

A. $P(T = 1 | F_{lu} = 0, U = 0)$ $\quad$ G. $P(T = 0 | F_{lu} = 0, U = 0)$

B. $P(T = 1 | F_{lu} = 0, U = 1)$ $\quad$ H. $P(T = 0 | F_{lu} = 0, U = 1)$

C. $P(T = 1 | F_{lu} = 1, U = 0)$ $\quad$ I. $P(T = 0 | F_{lu} = 1, U = 0)$

D. $P(T = 1 | F_{lu} = 1, U = 1)$ $\quad$ J. $P(T = 0 | F_{lu} = 1, U = 1)$

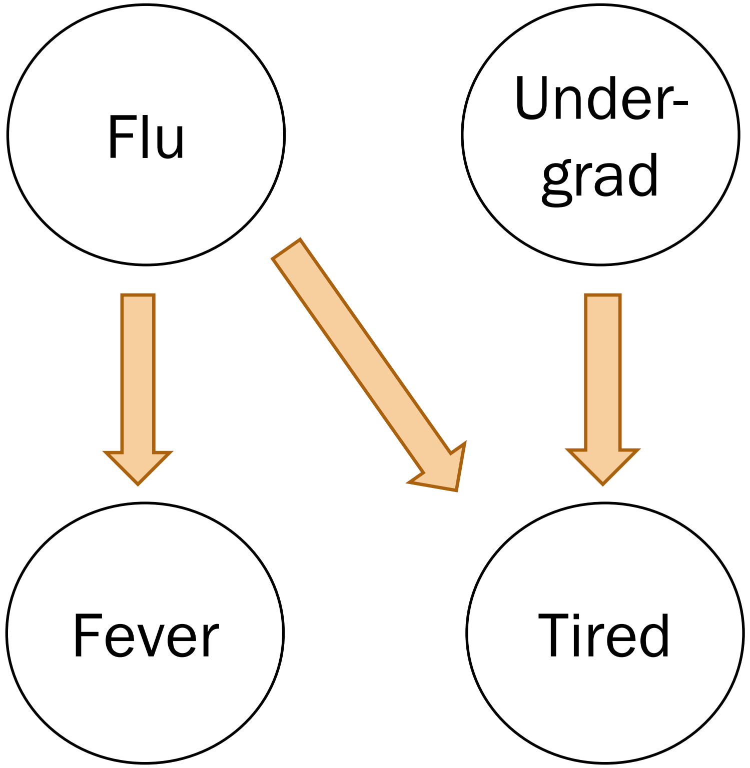
E. P

F. P

In a Bayesian Network, specify cond probs with respect to all parents.

A Bayesian Network

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



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

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

What would a CS109 student do?

1. Populate a Bayesian network by asking a Stanford flu expert
   or
   by using reasonable assumptions

2. **Answer inference questions**

Our focus today

Today's plan

Bootstrapping

- For a statistic
- For a p-value

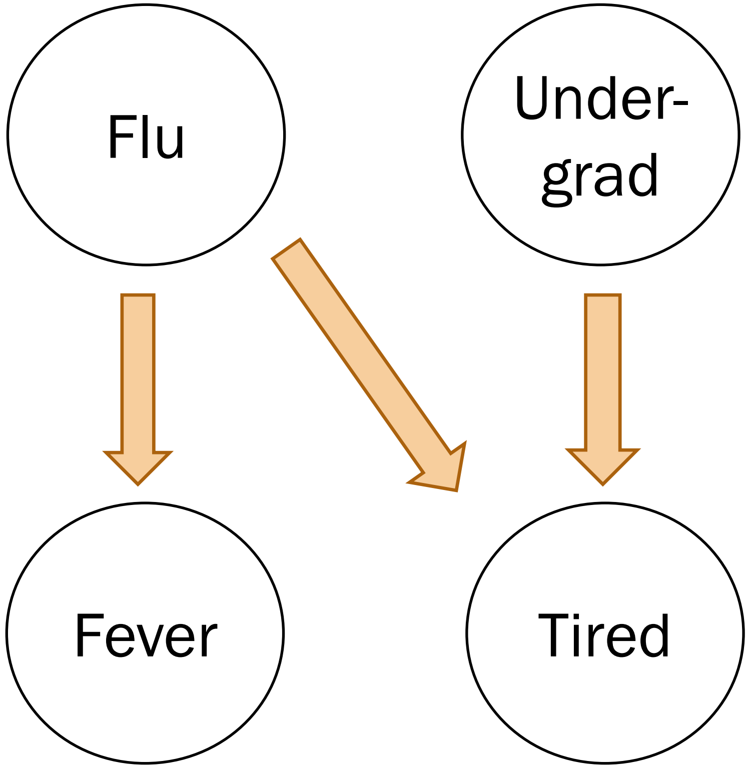
Definition: Bayesian Networks

Inference:

1. Math
2. Rejection sampling ("joint" sampling)
3. Optional: Gibbs sampling (MCMC algorithm)

Inference via math

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



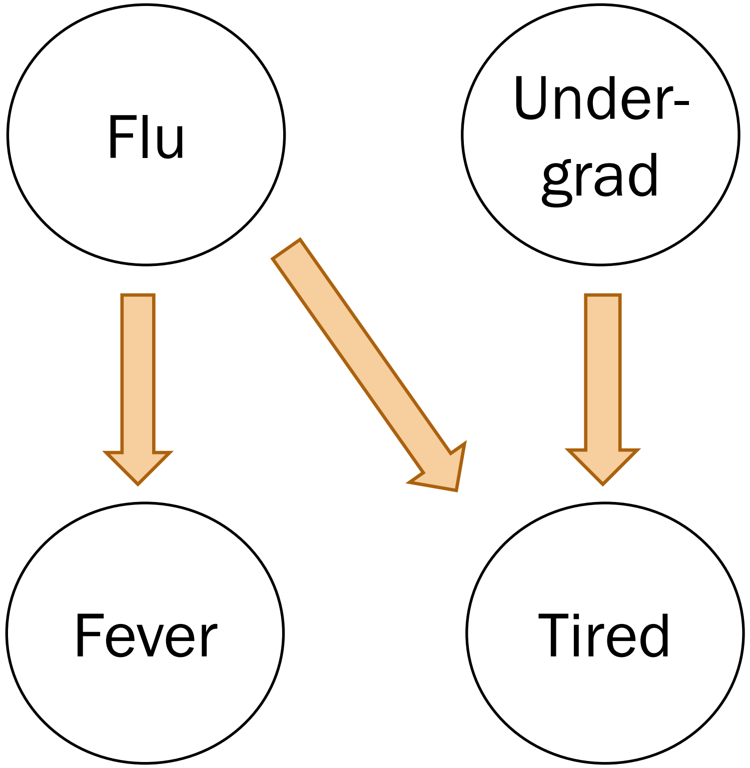
$P(F_{ev} = 1|F_{lu} = 1) = 0.9$
$P(F_{ev} = 1|F_{lu} = 0) = 0.05$

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

In a Bayesian Network, each random variable is conditionally independent of its non-descendants, given its parents.

Inference via math

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



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

Compute joint probabilities using chain rule.

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

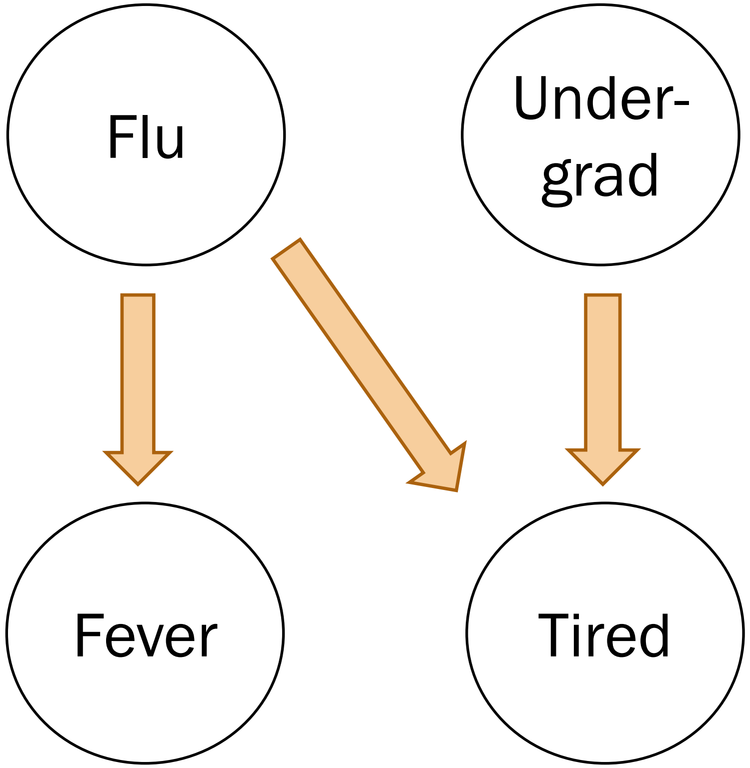
$$= 0.5472$$

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

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

Inference via math

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



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

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

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

   1. Compute joint probabilities

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

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

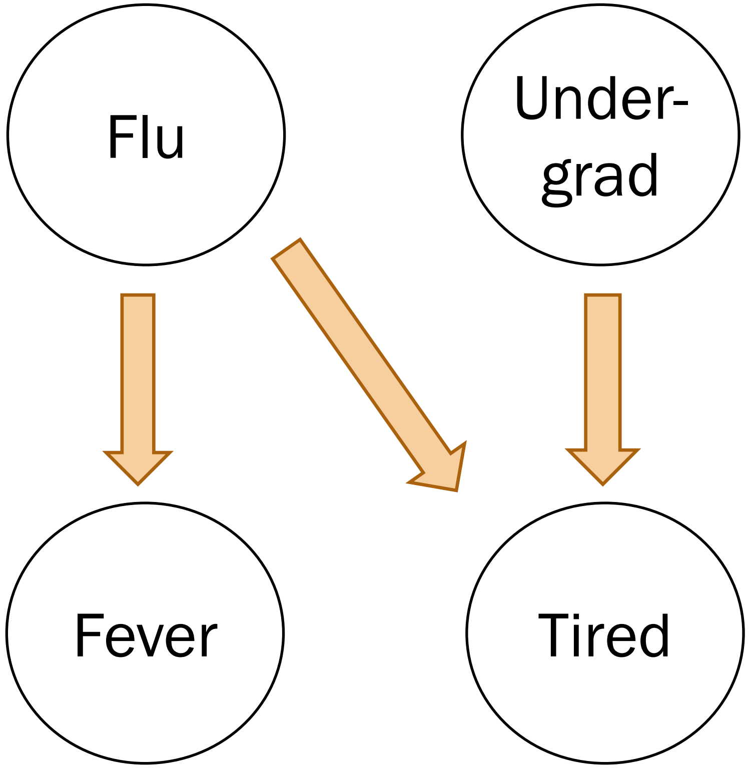
   2. Definition of conditional probability

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

$$= 0.095$$

Inference via math

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



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

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

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

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

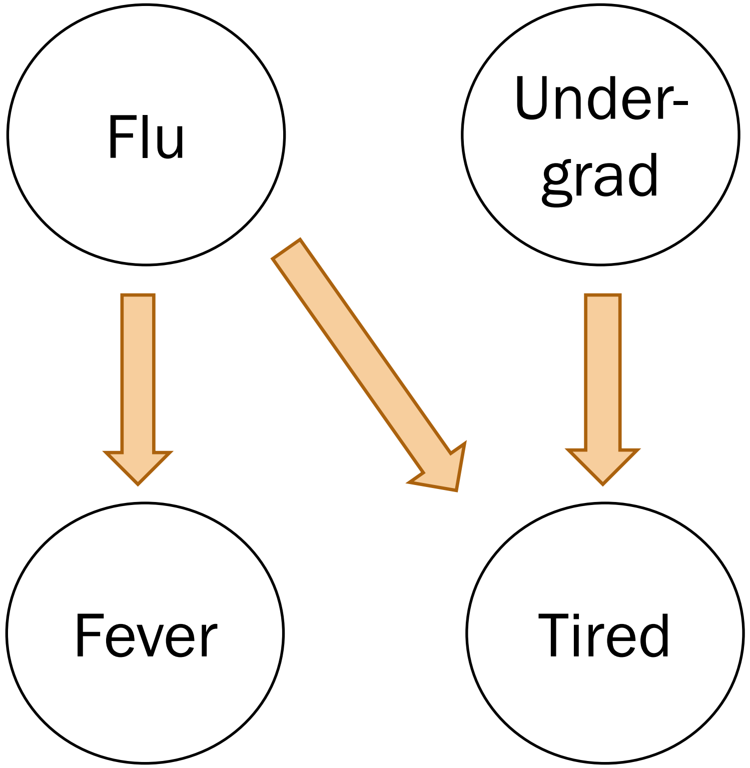
   2. Definition of conditional probability

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

$= 0.122$

Inference via math

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



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

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

Solving inference questions precisely is possible, but sometimes tedious.

Can we use sampling to do approximate inference?

Today's plan

Bootstrapping

- For a statistic
- For a p-value

Definition: Bayesian Networks

Inference:

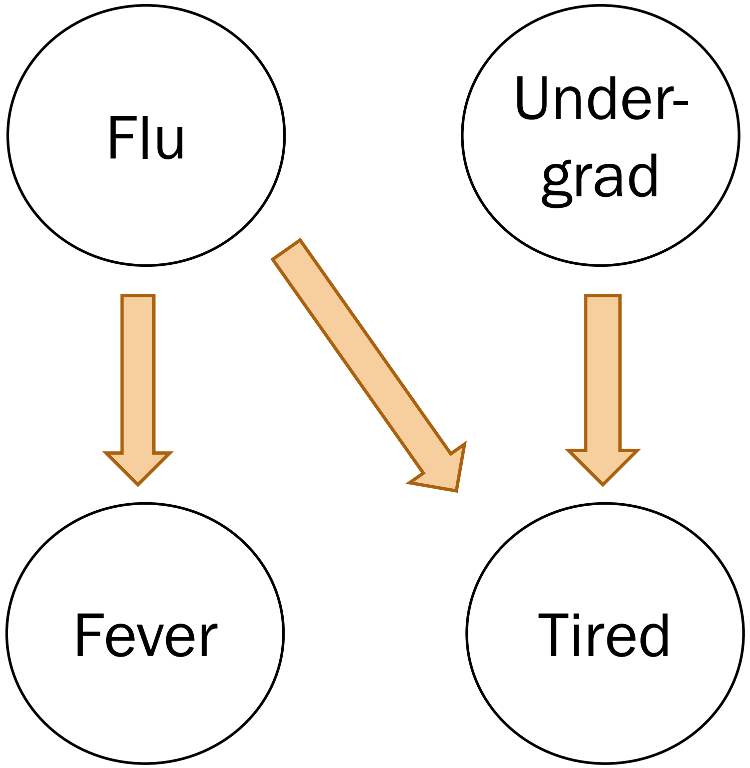
1. Math
2. Rejection sampling ("joint" sampling)
3. Optional: Gibbs sampling (MCMC algorithm)

Rejection sampling algorithm

Step 0:

Have a fully specified Bayesian Network

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



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

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

Rejection sampling algorithm

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

```
def rejection_sampling(event, observation):
    samples = sample_a_ton()
    samples_observation =
            reject_inconsistent(samples, observation)
    samples_event =
            reject_inconsistent(samples_observation, event)
    return len(samples_event)/len(samples_observation)
```

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

Rejection sampling algorithm

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

```
def rejection_sampling(event, observation):
    samples = sample_a_ton()
    samples_observation =
            reject_inconsistent(samples, observation)
    samples_event =
            reject_inconsistent(samples_observation, event)
    return len(samples_event)/len(samples_observation)
```

Rejection sampling algorithm

```python
N_SAMPLES = 100000
# Method: Sample a ton
# -------------------
# create N_SAMPLES with likelihood proportional
# to the joint distribution
def sample_a_ton():
    samples = []
    for i in range(N_SAMPLES):
        sample = make_sample() # a particle
        samples.append(sample)
    return samples
```

How do we make a sample $(F_{lu} = a, U = b, F_{ev} = c, T = d)$ according to the joint probability?

Rejection sampling algorithm

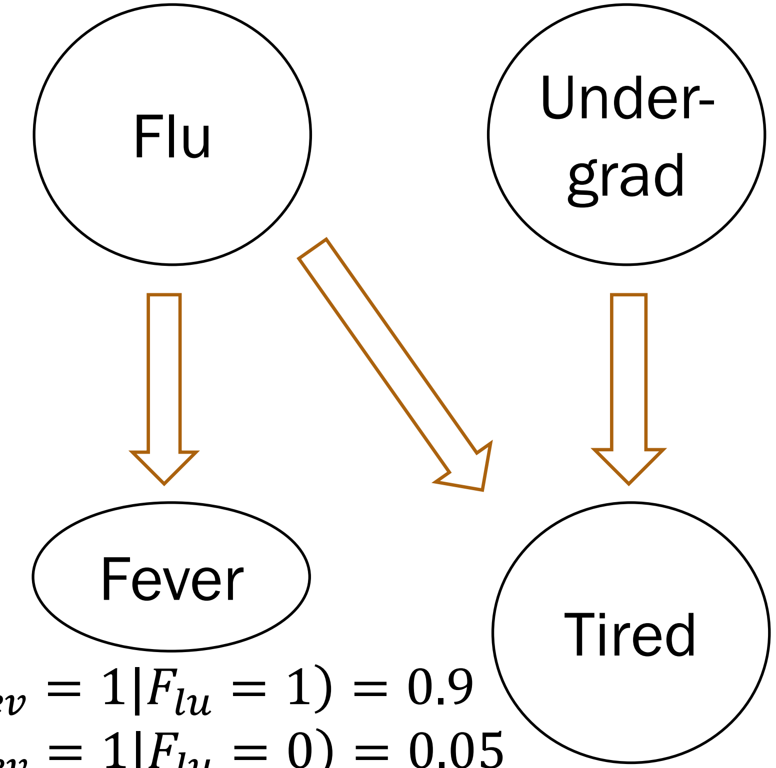
```python
# Method: Make Sample
# ---------------------
# create a single sample from the joint distribution
# based on the medical "WebMD" Bayesian Network
def make_sample():
    # prior on causal factors
    flu = bernoulli(0.1)
    und = bernoulli(0.8) # undergraduate

    # choose fever based on flu
    if flu == 1: fev = bernoulli(0.9)
    else:        fev = bernoulli(0.05)

    # choose tired based on (undergrad and flu)
    #
    # TODO: fill in
    #
    #

    # a sample from the joint has an
    # assignment to *all* random variables
    return [flu, und, fev, tir]
```

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



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

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

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

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

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

$P(T = 1 | F_{lu} = 1, U = 1) = 1.0$

Rejection sampling algorithm

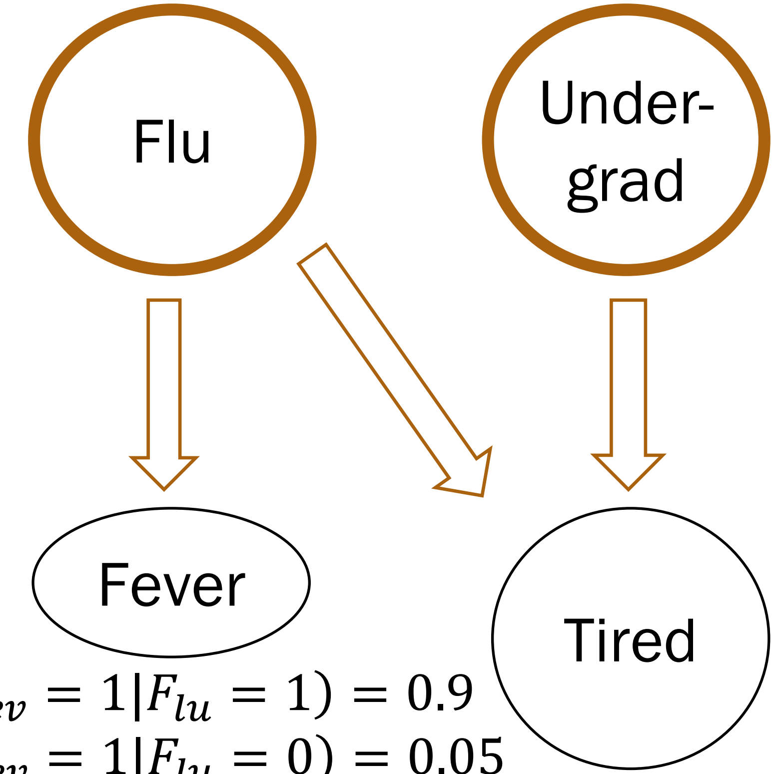
```python
# Method: Make Sample
# --------------------
# create a single sample from the joint distribution
# based on the medical "WebMD" Bayesian Network
def make_sample():
    # prior on causal factors
    flu = bernoulli(0.1)
    und = bernoulli(0.8) # undergraduate

    # choose fever based on flu
    if flu == 1: fev = bernoulli(0.9)
    else:        fev = bernoulli(0.05)

    # choose tired based on (undergrad and flu)
    #
    # TODO: fill in
    #
    #

    # a sample from the joint has an
    # assignment to *all* random variables
    return [flu, und, fev, tir]
```

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



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

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

Rejection sampling algorithm

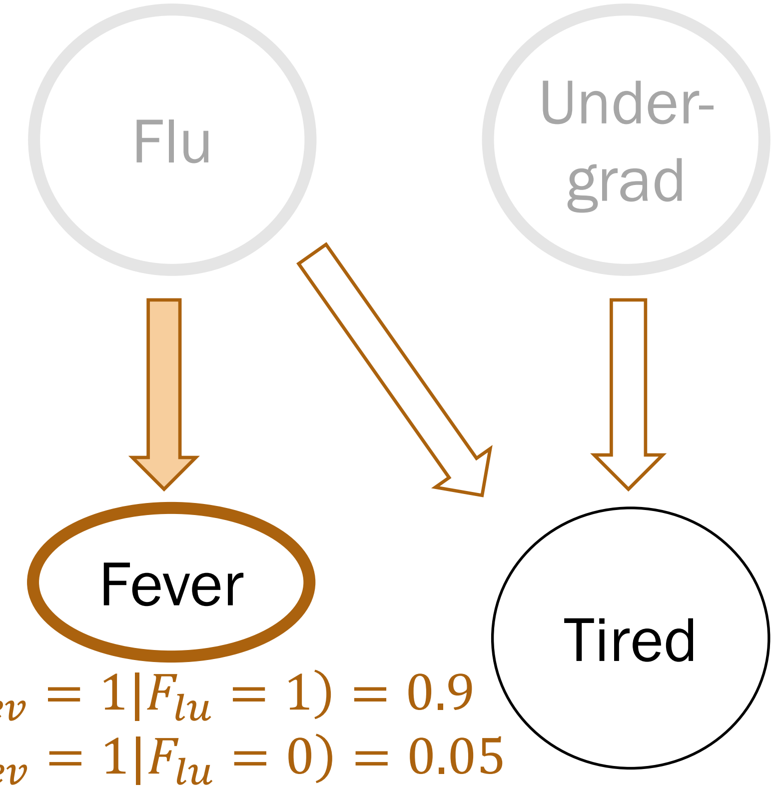
```python
# Method: Make Sample
# --------------------
# create a single sample from the joint distribution
# based on the medical "WebMD" Bayesian Network
def make_sample():
    # prior on causal factors
    flu = bernoulli(0.1)
    und = bernoulli(0.8) # undergraduate

    # choose fever based on flu
    if flu == 1: fev = bernoulli(0.9)
    else:        fev = bernoulli(0.05)

    # choose tired based on (undergrad and flu)
    #
    # TODO: fill in
    #
    #

    # a sample from the joint has an
    # assignment to *all* random variables
    return [flu, und, fev, tir]
```

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



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

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

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

Rejection sampling algorithm

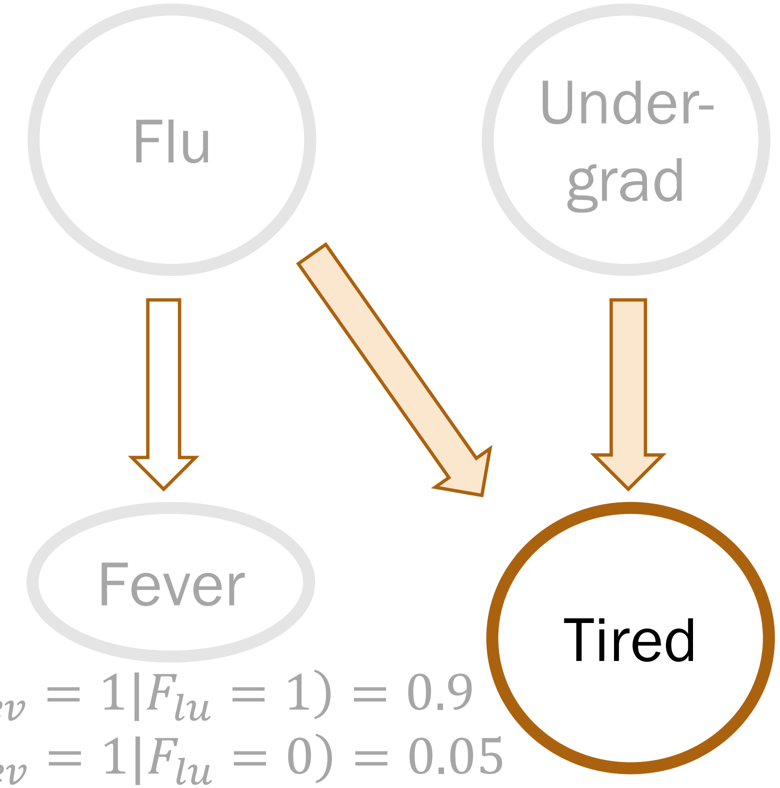
```python
# Method: Make Sample
# ---------------------
# create a single sample from the joint distribution
# based on the medical "WebMD" Bayesian Network
def make_sample():
    # prior on causal factors
    flu = bernoulli(0.1)
    und = bernoulli(0.8) # undergraduate

    # choose fever based on flu
    if flu == 1: fev = bernoulli(0.9)
    else:        fev = bernoulli(0.05)

    # choose tired based on (undergrad and flu)
    #
    # TODO: fill in
    #
    #

    # a sample from the joint has an
    # assignment to *all* random variables
    return [flu, und, fev, tir]
```

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



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

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

Rejection sampling algorithm

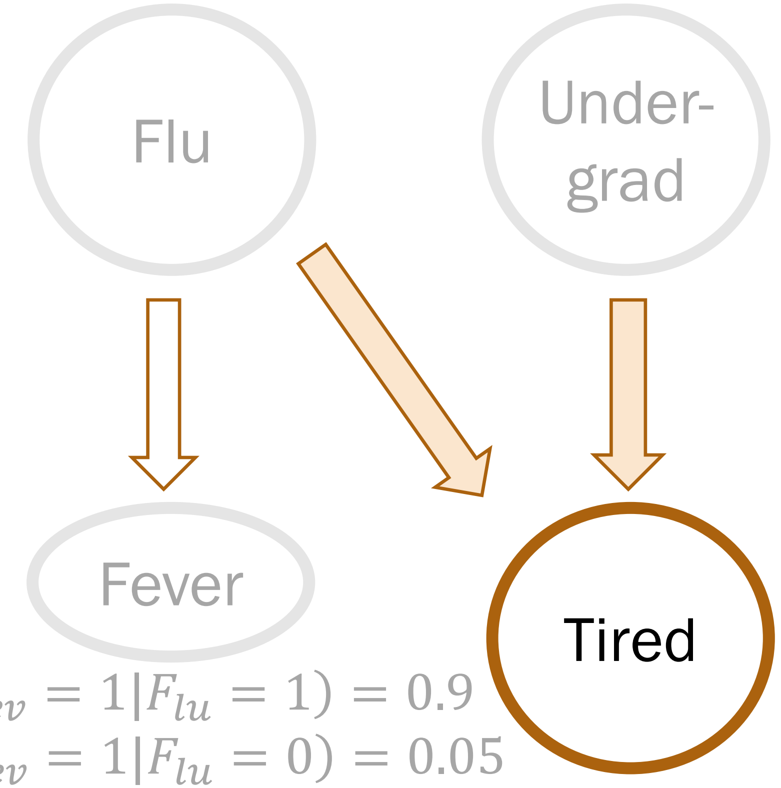
```python
# Method: Make Sample
# ---------------------
# create a single sample from the joint distribution
# based on the medical "WebMD" Bayesian Network
def make_sample():
    # prior on causal factors
    flu = bernoulli(0.1)
    und = bernoulli(0.8) # undergraduate

    # choose fever based on flu
    if flu == 1: fev = bernoulli(0.9)
    else:        fev = bernoulli(0.05)

    # choose tired based on (undergrad and flu)
    if   flu == 0 and und == 0: tir = bernoulli(1.0)
    elif flu == 0 and und == 1: tir = bernoulli(0.8)
    elif flu == 1 and und == 0: tir = bernoulli(0.9)
    else:                       tir = bernoulli(1.0)

    # a sample from the joint has an
    # assignment to *all* random variables
    return [flu, und, fev, tir]
```

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



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

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

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

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

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

$P(T=1|F_{lu}=1,U=1)=1.0$

Rejection sampling algorithm

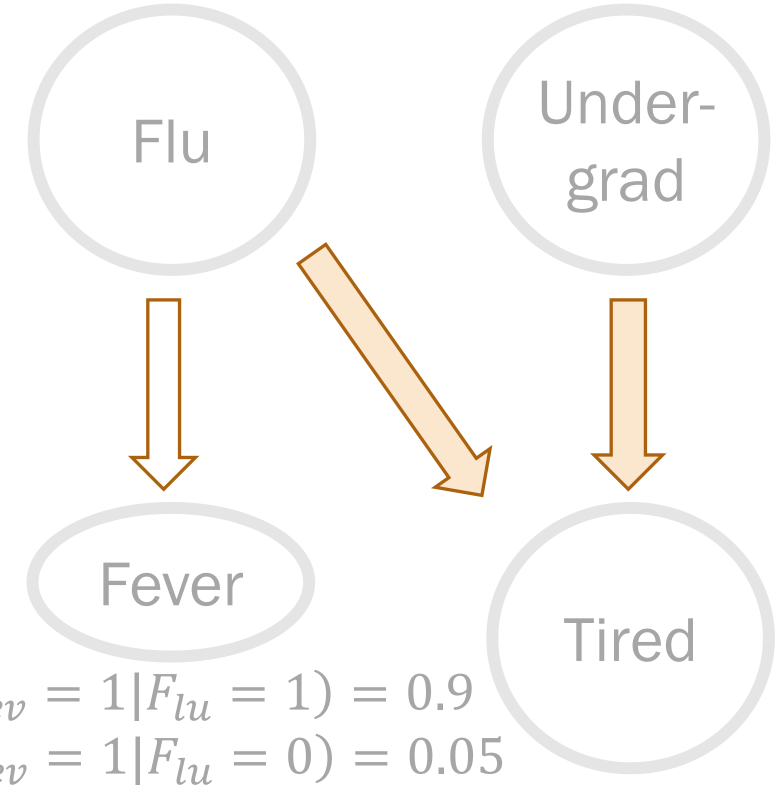
```python
# Method: Make Sample
# --------------------
# create a single sample from the joint distribution
# based on the medical "WebMD" Bayesian Network
def make_sample():
    # prior on causal factors
    flu = bernoulli(0.1)
    und = bernoulli(0.8) # undergraduate

    # choose fever based on flu
    if flu == 1: fev = bernoulli(0.9)
    else:        fev = bernoulli(0.05)

    # choose tired based on (undergrad and flu)
    if   flu == 0 and und == 0: tir = bernoulli(1.0)
    elif flu == 0 and und == 1: tir = bernoulli(0.8)
    elif flu == 1 and und == 0: tir = bernoulli(0.9)
    else:                       tir = bernoulli(1.0)

    # a sample from the joint has an
    # assignment to *all* random variables
    return [flu, und, fev, tir]
```

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



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

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

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

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

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

$P(T = 1 | F_{lu} = 1, U = 1) = 1.0$

Rejection sampling algorithm

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

```
def rejection_sampling(event, observation):
    samples = sample_a_ton()
    samples_observation =
              reject_inconsistent(samples, observ
    samples_event =
              reject_inconsistent(samples_observa
    return len(samples_event)/len(samples_obser
```

[flu, und, fev, tir]

```
Sampling...
[0, 1, 0, 1]
[0, 1, 0, 1]
[0, 1, 0, 1]
[0, 0, 0, 0]
[0, 1, 0, 1]
[0, 1, 1, 1]
[0, 1, 0, 0]
[1, 1, 1, 1]
[0, 0, 1, 1]
...
[0, 1, 0, 1]
Finished sampling
```

Rejection sampling algorithm

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

```
def rejection_sampling(event, observation):
    samples = sample_a_ton()
    samples_observation =
            reject_inconsistent(samples, observation)
    samples_event =
            reject_inconsistent(samples_observation, event)
    return len(samples_event)/len(samples_observation)
```

Keep only samples that are consistent with the observation $(U = 1, T = 1)$.

Rejection sampling algorithm

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

```
def rejection_sampling(event, observation):
    samples = sample_a_ton()
    samples_observation =
            reject_inconsistent(samples, observation)
    samples_
    
    return 
```

```
# Method: Reject Inconsistent
# ---------------------
# Rejects all samples that do not align with the outcome.
# Returns a list of consistent samples.
def reject_inconsistent(samples, outcome):
    consistent_samples = []
    for sample in samples:
        if check_consistent(sample, outcome):
            consistent_samples.append(sample)
    return consistent_samples
```

$(T = 1, U = 0)$

Rejection sampling algorithm

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

```
def rejection_sampling(event, observation):
    samples = sample_a_ton()
    samples_observation =
            reject_inconsistent(samples, observation)
    samples_event =
            reject_inconsistent(samples_observation, event)
    return len(samples_event)/len(samples_observation)
```

Conditional event = samples with $(F_{lu} = 1, U = 1, T = 1)$.

Rejection sampling algorithm

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

```
def rejection_sampling(event, observation):
    samples = sample_a_ton()
    samples_observation =
            reject_inconsistent(samples, observation)
    samples_event =
            reject_inconsistent(samples_observation, event)
    return len(samples_event)/len(samples_observation)
```

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

To the code!

Rejection sampling

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

With enough samples, you can correctly compute:

- Probability estimates
- Conditional probability estimates
- Expectation estimates

Because your samples are a representation of the joint distribution!

```
[flu, und, fev, tir]
Sampling...
[0, 1, 0, 1]
[0, 1, 0, 1]
[0, 1, 0, 1]
[0, 0, 0, 0]
[0, 1, 0, 1]
[0, 1, 1, 1]
[0, 1, 0, 0]
[1, 1, 1, 1]
[0, 0, 1, 1]
...
[0, 1, 0, 1]
Finished sampling
```

```
P(has flu | undergrad and is tired) = 0.122
```

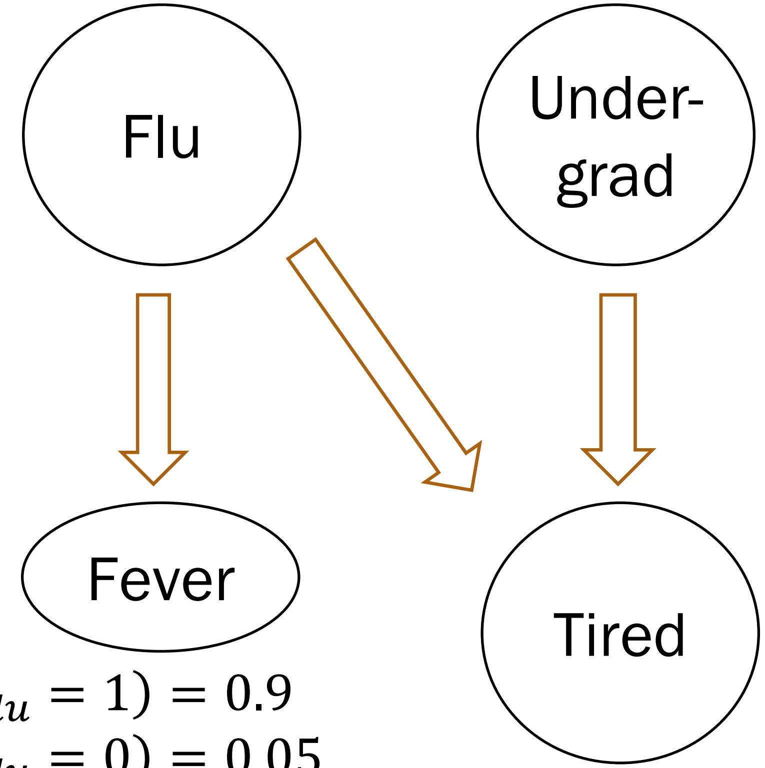
Disadvantages of rejection sampling

$$P(F_{lu} = 1 | F_{ev} = 1)?$$

What if we never encounter some samples?

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

$P(U = 1) = 0.8$



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

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

Disadvantages of rejection sampling

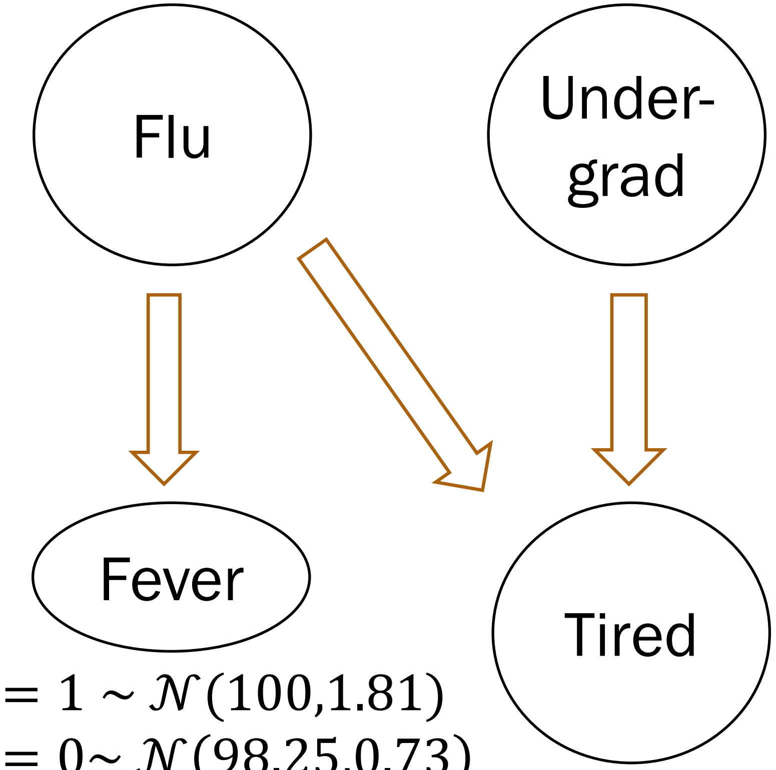
$$P(F_{lu} = 1 | F_{ev} = 99.4)?$$

What if we never encounter some samples?

What if random variables are continuous?

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

$P(U = 1) = 0.8$



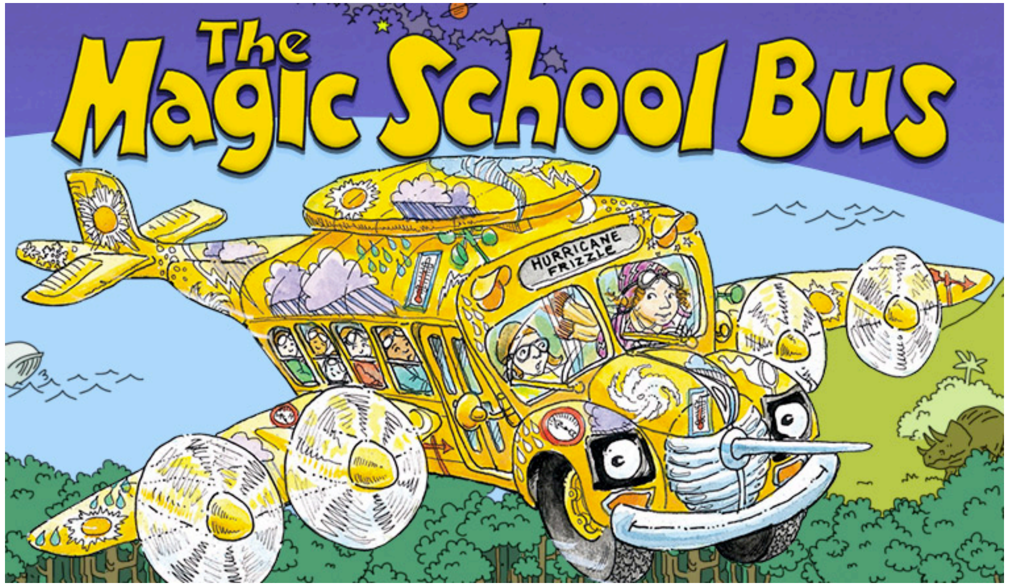
$F_{ev} = 1 | F_{lu} = 1 \sim \mathcal{N}(100, 1.81)$
$F_{ev} = 1 | F_{lu} = 0 \sim \mathcal{N}(98.25, 0.73)$

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

Gibbs Sampling (optional)

Basic idea:

- Fix all observed events
- Incrementally sample a new value for each random variable
- Difficulty: More coding for computing different posterior probabilities



Learn in extra slides/extra notebook!

(or by taking CS228/CS238)

Today's plan

Bootstrapping for hypothesis testing

Definition: Bayesian Networks

Inference:

1. Math
2. Rejection sampling ("joint" sampling)
3. Optional: Gibbs sampling (MCMC algorithm)

Gibbs Sampling

MCMC algorithm – Markov Chain Monte Carlo

- Monte Carlo: random algorithms
- Markov Chain: random event sequence state machine

Gibbs Sampling – a particular MCMC technique

Note: This material is *optional* and covered more in CS228,
but I want to show you that understanding Gibbs Sampling is not beyond your capabilities.

To the Jupyter Notebook!