Gaining knowledge from tests

UCSF NS Orientation 2024 — Stats II

Maxine Collard

A clairvoyant mouse?

A clairvoyant mouse?

You look at your data, and you see that you **reject the null hypothesis** that *manipulated mice do not predict the future* with a p-value of 0.014.

What do you do with this information?

Let's think about it another way

> **N.B.:** This example was originally made in September 2021, and has become less relatable over time.

Let’s think about it another way

For the last four weeks, you’ve been doing nothing but sitting in your room alone studying for your qualifying exam, with **absolutely zero human contact**.

You take it and pass—**hooray!**

Let’s think about it another way

But while you’ve been away studying, UCSF has announced a new policy requiring all graduate students to be tested for Covid-19 daily. You must provide a negative test result before entering any research building on-campus!

> *No, not really.*

You grab a test from one of the Color vending machines and wait at home for the results to come back. The next day, you get an email:

SARS-CoV-2 mRNA:
DETECTED

Do you think you have Covid?

Why do you think that?

Let’s try a simulation!

The scenario

> **N.B.**: SF no longer tracks Covid case rates, so I'll be using last year's numbers.

- Current rolling average for reported Covid cases in San Francisco: ~65 per day.
- SF's population is ~815,000. So, **reported** incidence of Covid in SF: ~7.97 per 100,000 per day.
- Let's aim high: assume Covid incidence is **4x reported**: ~31.88 per 100,000 per day.

The scenario

- To get to prevalence, let's assume everyone infected has Covid for 14 days.
- So, ballpark prevalence of Covid in SF: **~111.6 per 100,000**.

> If you were to pick a person at random from San Francisco with no other knowledge, this would be about the chance of picking someone who currently has Covid.

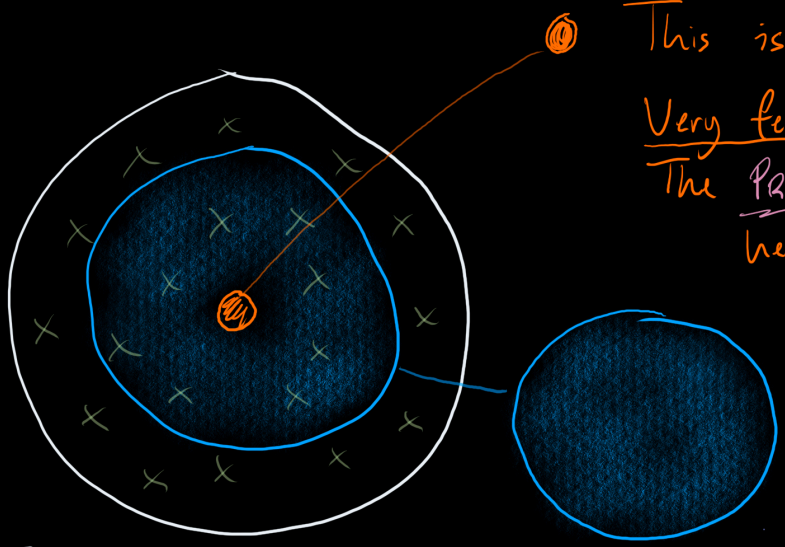
```
# September 2021 (using reported case count)
# covid_prevalence = 350 / 100000

# August 2022 (using reported case count)
# covid_prevalence = 288.4 / 100000

# September 2023 (using 4x reported case count)
# covid_prevalence = 446.32 / 100000

# September 2024 (no more case counts, using 2023 data)
covid_prevalence = 446.32 / 100000
```

Recall

The p-value is defined by thinking about what our observations would be by chance if we **presuppose that the null hypothesis is in fact true**:

$$p = \Pr(\text{we observe a difference} \mid \text{there is no difference})$$

> This is convenient to use, because we can always **impose** the null hypothesis by shuffling our data. This is called **permutation testing**.



This is SMALL.

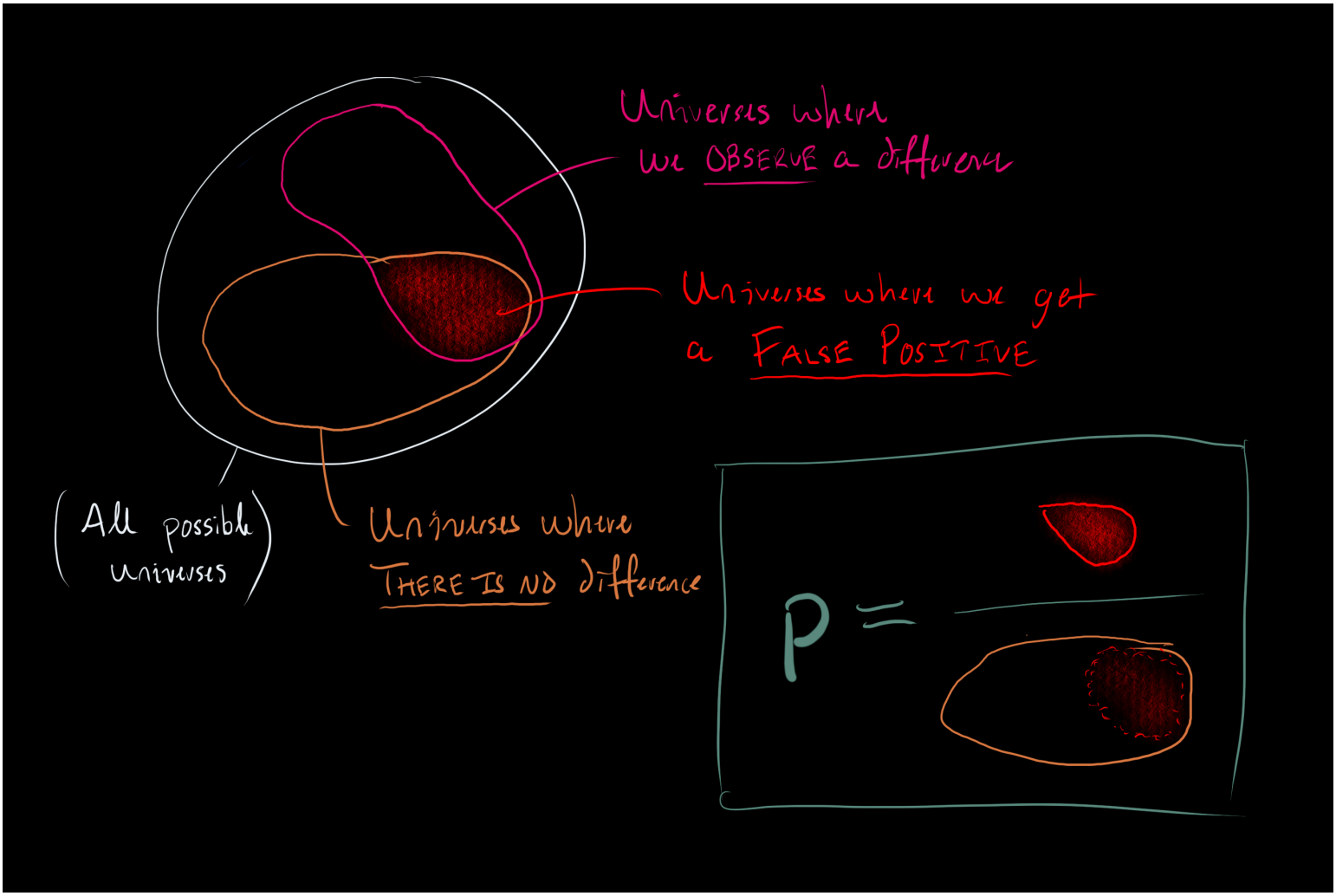
Very few darts will land here.
The PROBABILITY of a dart landing here is very low.

This is LARGE.

Lots of darts will land here.
The PROBABILITY of a dart landing here is very high.

The space of all possible universes is a giant dart board

PROBABILITY is a measure of SIZE.


Universes where
we OBSERVE a difference
Universes where we get
a FALSE POSITIVE
(All possible
Universes)
Universes where
THERE IS NO difference
p =

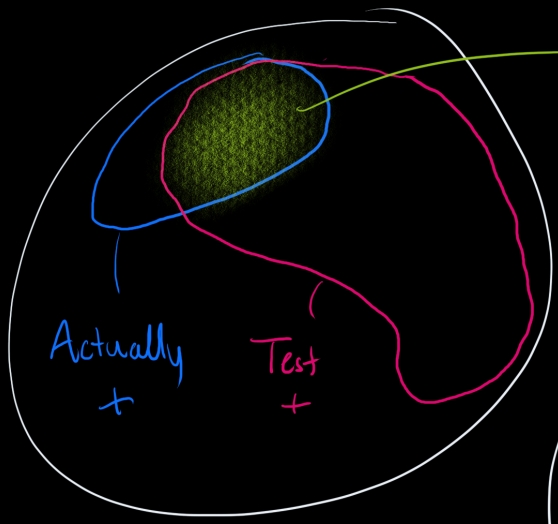
Usually, diagnostic tests are evaluated kind of like p-values: we use measures that ask how the *test* behaves when we **pre-suppose the truth**.

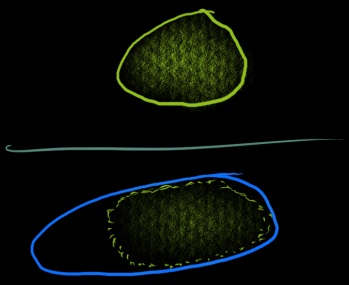
Sensitivity measures the probability that a person tests positive **given that they actually have the disease**:

$$\text{sensitivity} := \Pr(\text{test} + \mid \text{actually} +)$$

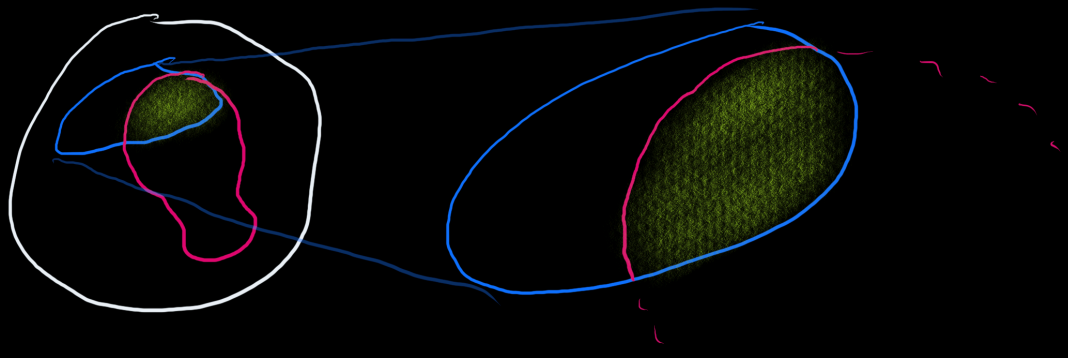
Specificity measures the probability that a person tests negative **given that they actually *do not* have the disease**:

$$\text{specificity} := \Pr(\text{test} - \mid \text{actually} -)$$



Sensitivity = 

(We "ZOOM IN" to only think about possible universes where we actually have Covid)



N.B.: The p-value can be thought of as

$$p = \Pr(\text{test} + \mid \text{actually} -)$$

Because there are only two test outcomes, $+$ and $-$, this means that

$$\begin{aligned} p &= 1 - \Pr(\text{test} - \mid \text{actually} -) \\ &= 1 - \text{specificity} \end{aligned}$$

So, the p-value is in the same "family" of measures about a test as sensitivity and specificity.

Back to the code

Let's suppose that the currently used RT-PCR test for Covid has approximately the following characteristics:

```
# September 2021
# rtpcr_sensitivity = 0.777
# rtpcr_specificity = 0.988

# August 2022
# https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7350782/
rtpcr_sensitivity = 0.733
# https://www.uptodate.com/contents/covid-19-diagnosis
rtpcr_specificity = 0.97

# September 2023: These characteristics haven't appreciably changed

# September 2024: "
```

> **N.B.**: Finding accurate values is challenging in practice.

> **N.B.**: We can compute the p-value of this test using the formula from earlier, $1 - \text{specificity}$:

```
print( f'> $p$ = **{1 - rtpcr_specificity:0.4f}**' )
```

> p = **0.0300**

You just got a positive test

What do the sensitivity and specificity **tell** you?

You already **know** that you've tested positive. What you would like to **infer** is whether or not you actually have Covid.

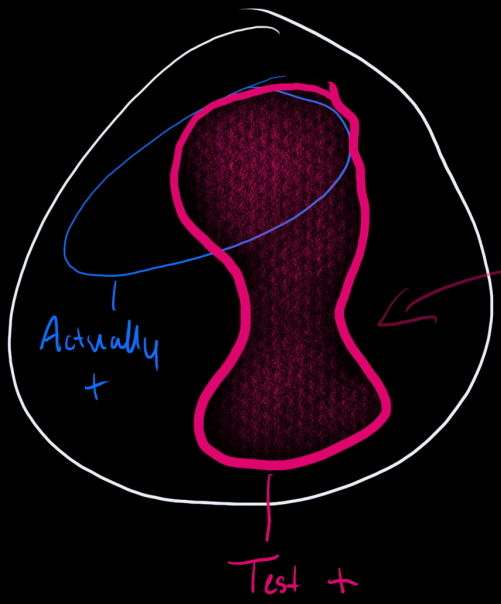
Sensitivity and specificity do not tell you this.

And so, neither do p-values!

What you want to know are the quantities with the **opposite conditioning**:

$$p = \Pr(\text{test} + \mid \text{actually} -)$$

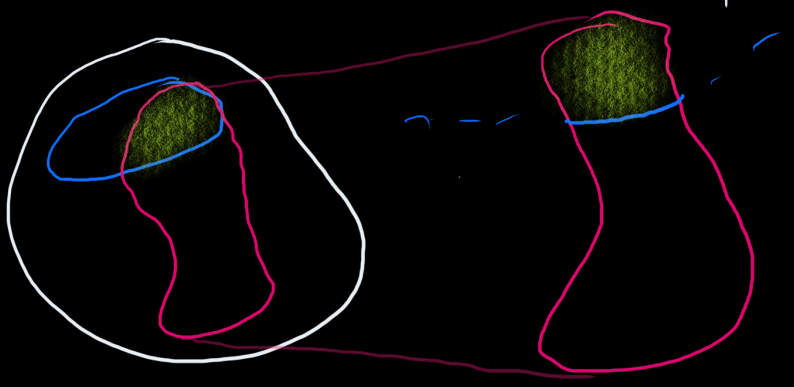
$$? = \Pr(\text{actually} + \mid \text{test} +)$$



You got a + test

That means you know that you live HERE, in one of the universes in which you got a + test.

Because of this knowledge, we can "Zoom In" on this space:



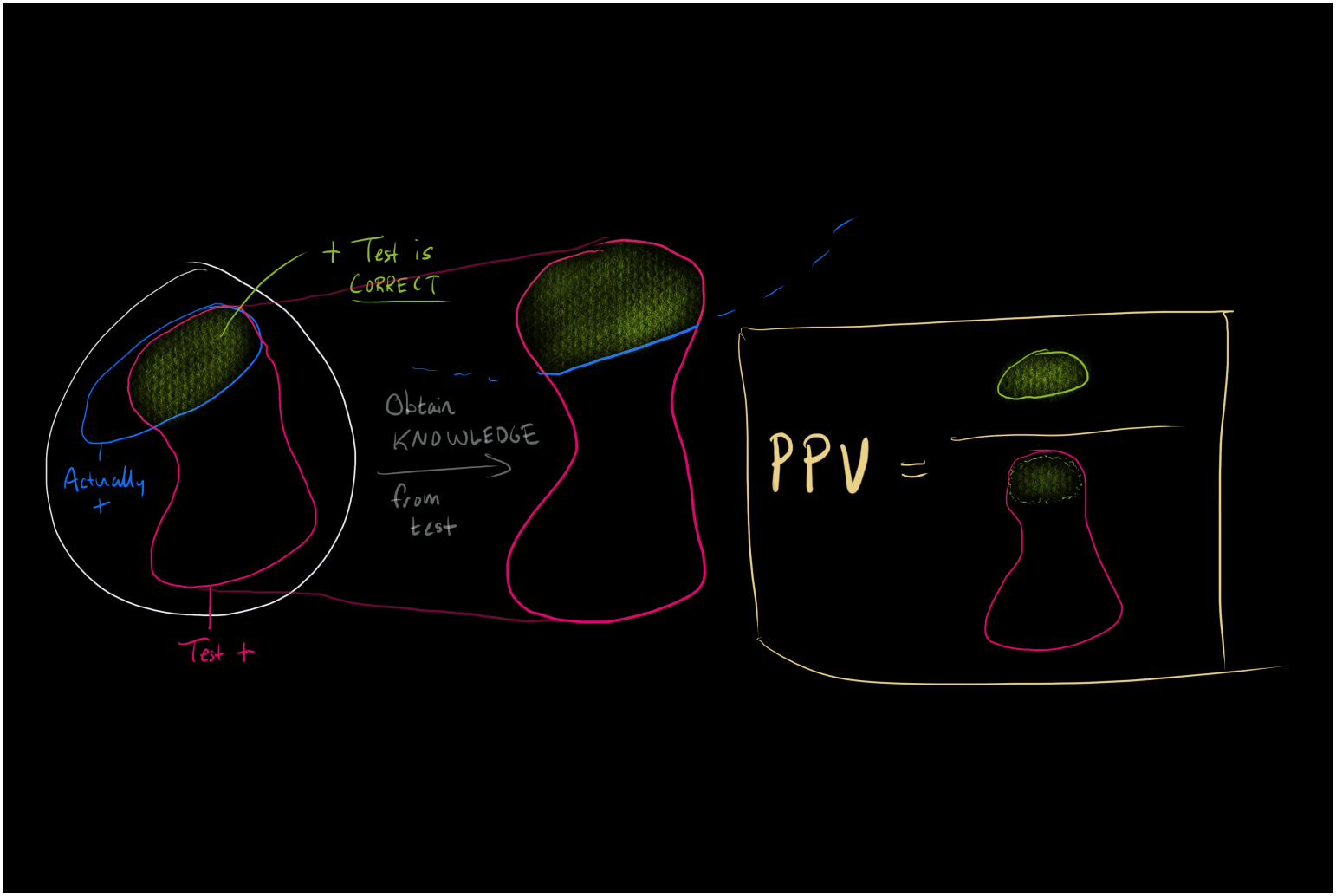
Positive predictive value is the probability one *actually has the disease* **given** a positive test result:

$$\text{PPV} := \Pr(\text{actually} + \mid \text{test} +)$$

> Now that you have a positive Covid test, the **PPV** is the probability that you **actually** have Covid.

Negative predictive value is the probability one *actually does not have the disease* **given** a negative test result:

$$\text{NPV} := \Pr(\text{actually} - \mid \text{test} -)$$


+ Test is
CORRECT
Actually
+
Test +
Obtain
KNOWLEDGE
from
test
PPV =

What do you think are the PPV and NPV for the RT-PCR test?

N.B.: Always write docstrings for your code:

```
1 help( sample_results )
```

```
Help on function sample_results in module __main__:

sample_results(n, prevalence, sensitivity, specificity)
    Sample disease presence and test results with the given characteristics

    Arguments:
    n -- The number of individiuals to sample
    prevalence -- The probability of finding the actual disease in the
population
    sensitivity -- The probability of testing positive given one has the
disease
    specificity -- The probability of testing negative given one does not have
the disease

    Reutrns:
    has_disease -- Shape (n,) whether each individual has the disease
```

Let's simulate one day of Covid tests at UCSF:

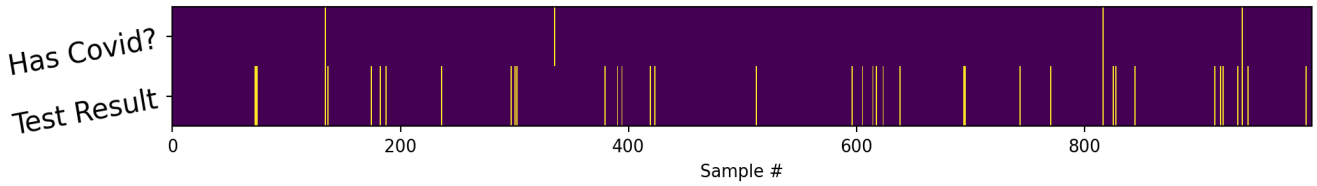
```
# Approximately the number of employees at UCSF
n_employees = 24000
# Run our simulation and grab the results
# (See code for the slides online)
has_covid, rtpcr_positive = sample_results( n_employees,
                                            covid_prevalence,
                                            rtpcr_sensitivity,
                                            rtpcr_specificity )
# If I put an f in front of my string, I can use curly braces to
# put Python code inside of it
print( f'> Cases today: **{np.sum( has_covid )}**\\' )
print( f'> Positive tests today: **{np.sum( rtpcr_positive )}**' )
```

> Cases today: **109**
> Positive tests today: **811**

Clearly something funky is going on: **there are way more positive tests than cases**!

As is always good practice, let's take a look at a subsample of the raw data:

```
# Subplots allows us to lay out some axes (`ax`) inside of a
# figure (`fig`) that we can manipulate later
fig, ax = plt.subplots( figsize = (12, 1.2) )

# Use the plotting function we just made on these axes
# (See code for the slides online)
sample_plot( ax, has_covid, rtpcr_positive )

# We're all done building the plots, we want to displaly them now
plt.show()
```



By eye, it *definitely* looks like there are a lot of false positives.

What are the PPV and NPV?

```
# (See code for the slides online)
ppv_naive, npv_naive = predictive_value( has_covid, rtpcr_positive )

# We can do math in our curly braces; here I convert the ratios into
# percents. The `:0.2f` at the end tells Python that I want 2 digits
# after the decimal point (not at all intuitive; blame the C people
# who wrote the original `printf` function).
print( f'> Naive PPV: **{ppv_naive * 100:0.2f}%**\\' )
print( f'> Naive NPV: **{npv_naive * 100:0.2f}%**' )
```

> Naive PPV: **9.86%**
> Naive NPV: **99.87%**

Let's talk about the good news first:

- A *negative* test is **extremely informative**: if you receive a negative test result, you can say with almost certainty that you are in the clear.

And now, the bad news:

- In this setup, a *positive* test is **not very informative**: in fact, if you were to receive a positive RT-PCR test, *there is a ~90% chance that you still don't have Covid*.
- Put another way, **more than 9 out of every 10 positive tests are actually false positives**.

Do you think this will scale well?

Suppose quarantine lasts for 10 days (8 work days). What fraction of the workforce would have to be quarantining at any given moment from false positive tests alone?

```
# 10 days is the policy, but a couple of those will be a weekend
quarantine_workdays = 8
# False positives *do not* have Covid *and* test positive
quarantine_fp_per_day = np.sum( ~has_covid & rtpcr_positive )
quarantine_fp_simultaneous = quarantine_fp_per_day * quarantine_workdays
quarantine_fraction = quarantine_fp_simultaneous / n_employees

print( f'> Fraction of workforce quarantined: **{quarantine_fraction * 100:
```

> Fraction of workforce quarantined: **24.37%**

But, **n.b.**! This test also has an extremely high **accuracy**—that is, the probability of the test being *correct*:

```
# == operates element by element on arrays
# `a == b` is an array where each element is `True` if the
# corresponding elements of `a` and `b` are equal
accuracy_naive = np.sum( has_covid == rtpcr_positive ) / len( has_covid )

print( f'> Naive accuracy: **{accuracy_naive * 100:0.2f}%**' )
```

> Naive accuracy: **96.83%**

The accuracy of the test is hiding the fact that we're actually doing quite a terrible job with the *positive predictive value*—the much more actionable piece of information—because **the prevalence of Covid cases is so low**!

What's going wrong?

Is this a bad test?

Do you still think you have Covid?

A better way to use Covid tests

A better way to use Covid tests

In our simulation, the **prevalence** of Covid was thought of as **the probability that someone in our sample had Covid**. We took this to be the prevalence of Covid in San Francisco, under the assumption that we were testing people essentially randomly, and had no criteria for how we were selecting people to test.

A better way to use Covid tests

After briefly quarantining a substantial portion of its workforce, UCSF has decided to try a new Covid testing schema.

> In medicine, we call this "*continual improvement*"!

Now, **only people who have an exposure to a confirmed symptomatic case** will be tested.

A better way to use Covid tests

Before, we had no clue about the people being tested. This new criterion means we have more **knowledge** (of the exposure).

This knowledge changes our **prior belief**—before even running the test!—of whether or not the person being tested has Covid: we *expect* that someone who has had an exposure is more likely to have Covid than someone random from the general population.

A better way to use Covid tests

All of the characteristics of the test itself are still exactly the same—the same reagents, the same technique, everything.

The only thing we'll change to simulate this new scenario is the **proportion of tested people who have Covid**.

Let's say that an exposure to a symptomatic case of Covid carries an associated risk of infection of 5%:

```
1 covid_belief_sus = 0.05
```

What do you think will happen to the PPV of our test in this new scheme?

How about the NPV?

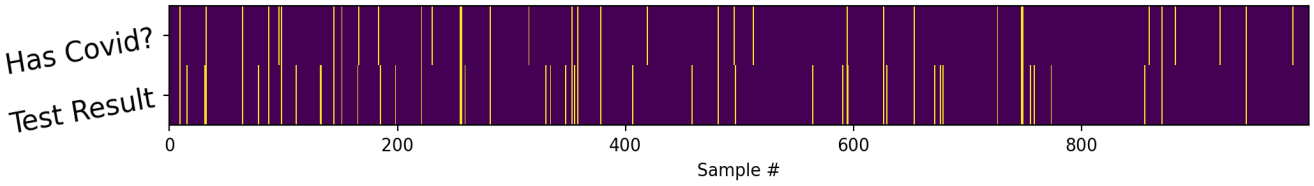
Let’s run a simulation of a group of employees with suspected Covid exposure:

```
n_sus = 1000
has_covid_sus, rtpcr_positive_sus = sample_results( n_sus,
                                                    covid_belief_sus,
                                                    rtpcr_sensitivity,  # S
                                                    rtpcr_specificity ) # S

# Python is fully Unicode compatible for strings, so there's no
# reason not to include emoji 😜
print( f'> Cases among the suspicious 🤔: **{np.sum( has_covid_sus )}**\\'
print( f'> Positive tests among the suspicious 🤔: **{np.sum( rtpcr_positiv

# You can't use emoji in variable names, though 😢
# Got to switch to Haskell for that ...
```

> Cases among the suspicious 🤔: **35**
> Positive tests among the suspicious 🤔: **51**

Dummy check, round 2

```
fig, ax = plt.subplots( figsize = (12, 1.2) )
sample_plot( ax, has_covid_sus, rtpcr_positive_sus )
plt.show()
```



Already this is looking much more reasonable. **What are the PPV and the NPV**?

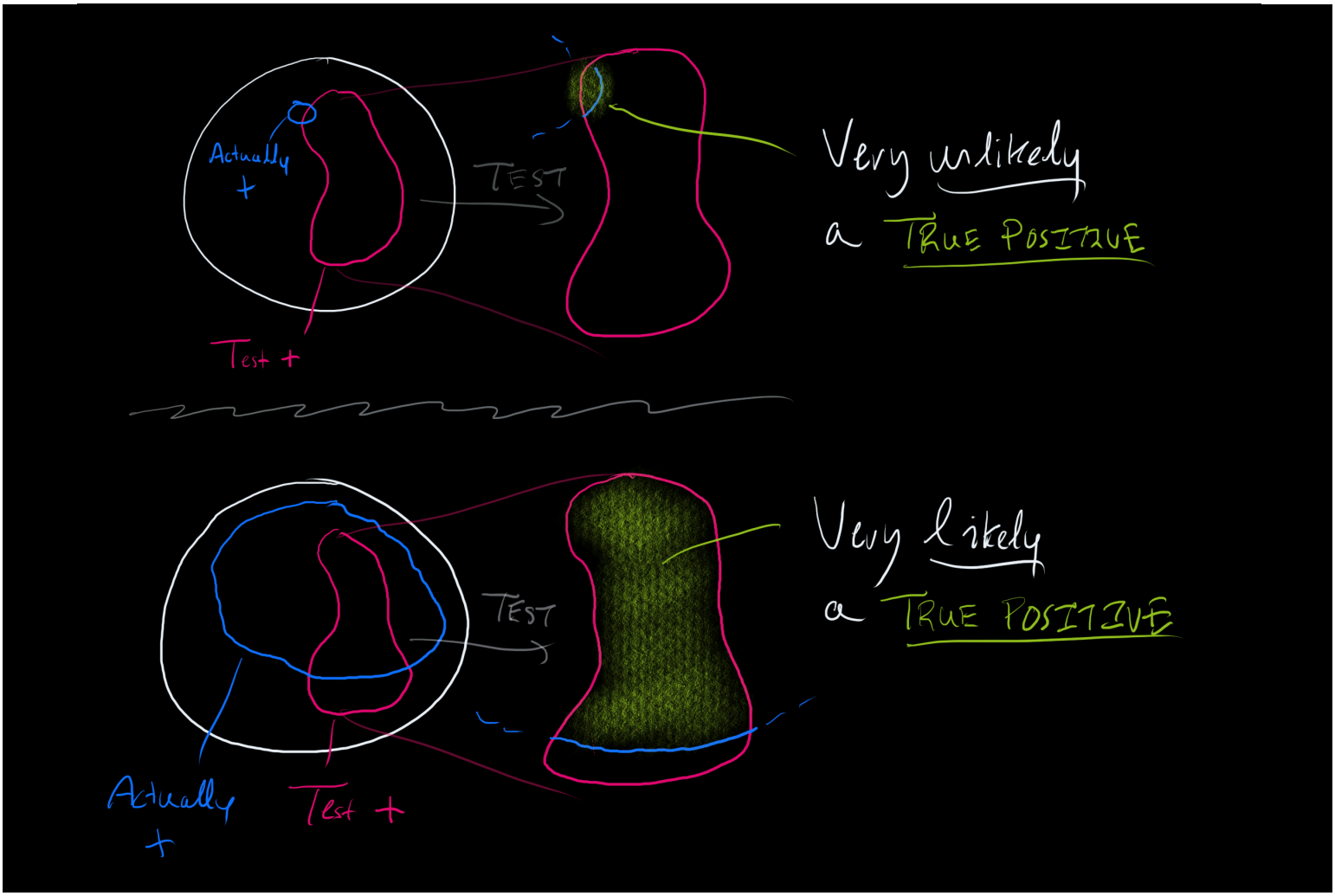
```
ppv_sus, npv_sus = predictive_value( has_covid_sus, rtpcr_positive_sus )

print( f'> Suspicious PPV: **{ppv_sus * 100:0.2f}%**\\' )
print( f'> Suspicious NPV: **{npv_sus * 100:0.2f}%**' )
```

> Suspicious PPV: **45.10%**
> Suspicious NPV: **98.74%**

Aha!

The PPV has jumped dramatically, from about 10% to about **60%**!

In this setting—where we have some **additional knowledge** about who we're testing—the **same exact test** has become **a lot more useful**.


Actually +
TEST
Test +
Very unlikely a TRUE POSITIVE
TEST
Actually +
Test +
Very likely a TRUE POSITIVE

> **N.B.**: There is no free lunch, of course!
>
> The cost we pay is that the NPV in this scenario is now *slightly* lower!

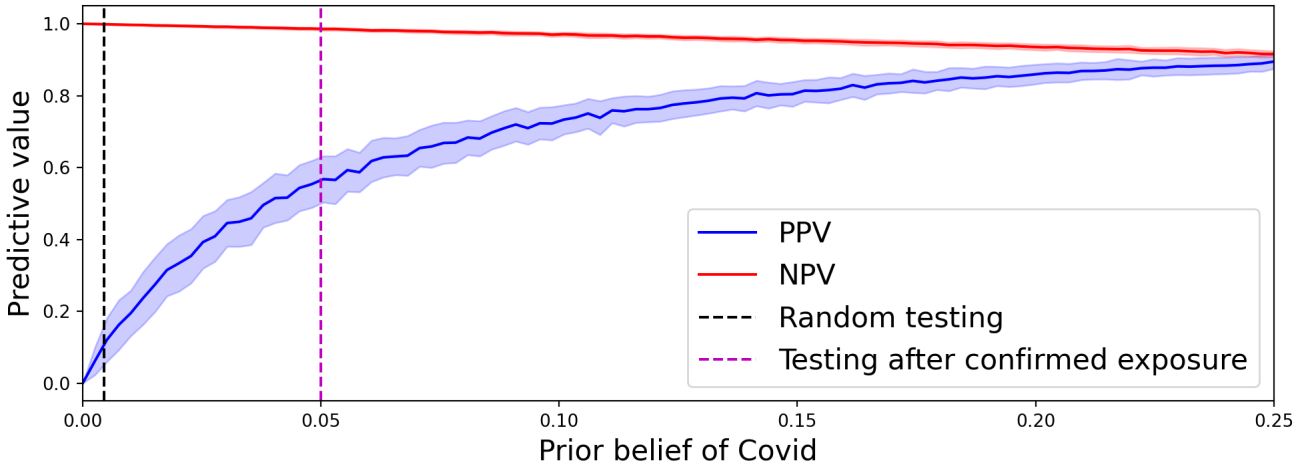
The critical takeaway message is this:

How useful a test is depends on our prior belief about the thing we're testing for.

How much knowledge do we need?

Let’s see how this whole picture plays out as we “sweep through” a bunch of different possible priors.

```
# (See code in slide repo for implementations)

# Run our simulation
# Pick beliefs linearly spaced between 0 and 0.25
covid_beliefs_sweep = np.linspace( 0, 0.25, 100 )
ppvs_sweep, npvs_sweep = belief_sweep( rtpcr_sensitivity,
                                       rtpcr_specificity,
                                       covid_beliefs_sweep )
# Build our plot
fig, ax = plt.subplots( figsize = (12, 4) )
plot_sweep( ax, covid_beliefs_sweep, ppvs_sweep, npvs_sweep,
            annotations = [(covid_prevalence, 'k--', 'Random testing'),
                           (covid_belief_sus, 'm--', 'Testing after confirmed exposure')],
            xlabel = 'Prior belief of Covid' )
plt.show()
```



In this example, we saw that the PPV of our Covid test depended strongly on the scheme we used to select which people to test, even while **the test itself remained entirely the same**.

From the previous slide, even **very weak improvements in our prior belief** can lead to **vast changes in the utility of our test**!

This means that **PPV is not just a statement about the test**.

PPV is a statement about how the test is used.

What does a test do?

What does a test do?

Before we ran the test, we had some **prior knowledge** about whether we had Covid:

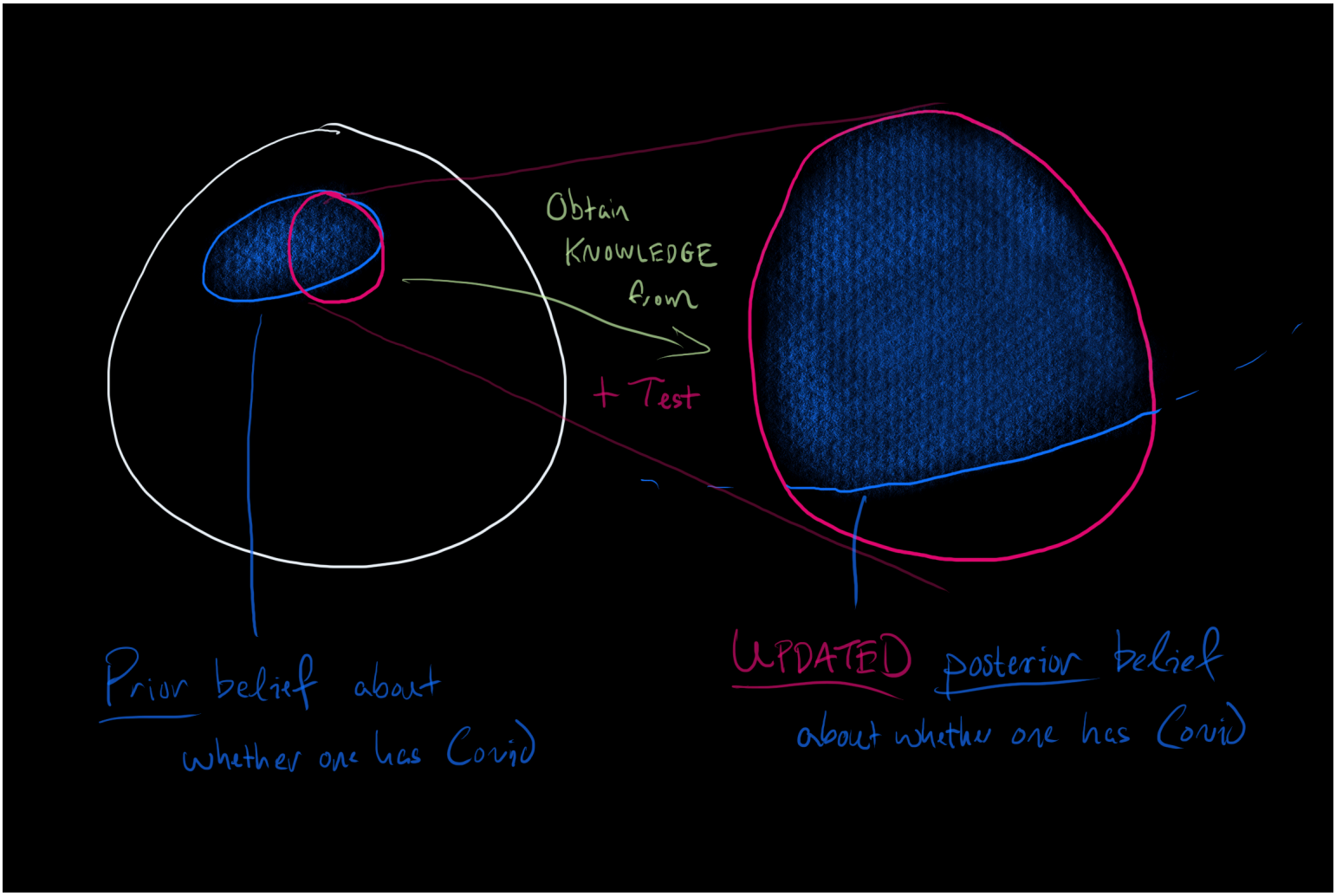
$$\Pr(\text{actually } +)$$

What does a test do?

When we get **new knowledge**—the result of the test—we **update** our belief of whether we have Covid. Our knowledge about whether we have Covid *after* the update caused by the test is precisely the **PPV**:

$$\Pr(\text{actually } +)$$
$$\downarrow$$
$$\Pr(\text{actually } + \mid \text{test } +)$$

This is also known as the **posterior probability** (as in, the probability that we have *after*) given a positive test.


Obtain
KNOWLEDGE
from
+ Test
Prior belief about
whether one has Covid
UPDATED posterior belief
about whether one has Covid

N.B.: The amount of information provided by a test result is called the result's **Bayes factor**, K.

While the PPV—which is the **posterior probability** of disease given a positive test—changes depending on the prior, the following **ratios** are always proportional:

$$\frac{\Pr(\text{actually } + \mid \text{test } +)}{\Pr(\text{actually } - \mid \text{test } +)} = K \frac{\Pr(\text{actually } +)}{\Pr(\text{actually } -)}$$

So, the Bayes factor tells us how much receiving a positive test result **changes** our prior belief. Test results with larger Bayes factors **change our beliefs more**.

Tests in science

Tests in science

For Covid testing, we obtained some bit of noisy data (a positive test result), and sought to **infer** something about what was **actually** happening (whether we truly have Covid).

We made this inference quantitative using the Covid RT-PCR test's **PPV**:

$$\text{PPV} = \Pr(\text{actually} + \mid \text{test} +)$$

Tests in science

In science, we use **statistical tests**.

Just like the Covid RT-PCR, statistical tests are **noisy** bits of data. What we would like to do is to **infer** something about what is **actually** happening in the world on the basis of these test results.

Example—Correlation

Let's say that you're analyzing some data. You plug two variables you're working on, `x` and `y`, into a magic black box like `scipy.stats.pearsonr`, and it pops out:

```
>>> scipy.stats.pearsonr( x, y )
(0.21365304326850618, 0.03281172835181021)
```

> (The first value is the correlation r, the second is the p-value.)

Cool, it's significant! We got a **positive test result**.

Example—Correlation

Given this:

```
>>> scipy.stats.pearsonr( x, y )
(0.21365304326850618, 0.03281172835181021)
```

… what do you now know about whether there is actually a relationship between **x** and **y**?

Example—Correlation

```
>>> scipy.stats.pearsonr( x, y )
(0.21365304326850618, 0.03281172835181021)
```

This information, by itself, is **not a statement about whether the null hypothesis is actually true or false**—just as a positive Covid test is not, by itself, a statement about whether you actually have Covid.

We must infer the truth from the test

What does this inference depend on?

Example—Correlation

```
>>> scipy.stats.pearsonr( x, y )
(0.21365304326850618, 0.03281172835181021)
```

With our shiny positive test result, what we really care about is the **posterior probability**, given we saw this test result, of whether there is a relationship between x and y. This is precisely the **PPV**:

$$\text{PPV} := \Pr(\text{actually a relationship} \mid \text{test rejects null hyp.})$$

Example—Correlation

```
>>> scipy.stats.pearsonr( x, y )
(0.21365304326850618, 0.03281172835181021)
```

$$\text{PPV} := \Pr(\text{actually a relationship} \mid \text{test rejects null hyp.})$$

But, just as before with Covid RT-PCR,

PPV depends on our prior knowledge of whether there is a real relationship between `x` and `y`—

not just the characteristics of `pearsonr`!

Example—Correlation

```
>>> scipy.stats.pearsonr( x, y )
(0.21365304326850618, 0.03281172835181021)
```

Suppose I got this result by going out into the world and running the `pearsonr` function between every pair of two datasets I could get my hands on.

What is the probability that any two of those randomly chosen datasets are **actually** causally related to one another? That is, what is the **prior**,

$$\Pr(\text{actually a relationship})$$

Example—Correlation

```
>>> scipy.stats.pearsonr( x, y )
(0.21365304326850618, 0.03281172835181021)
```

What is the probability that any two of those randomly chosen datasets are **actually** causally related to one another? That is, what is the **prior**,

$$\Pr(\text{actually a relationship})$$

It is astronomically small.

Example—Correlation

```
>>> scipy.stats.pearsonr( x, y )
(0.21365304326850618, 0.03281172835181021)
```

In this experimental setup of **randomly testing** correlations, what is the **PPV**?

It is exceedingly low.

Example—Correlation

```
>>> scipy.stats.pearsonr( x, y )
(0.21365304326850618, 0.03281172835181021)
```

So, how do I interpret this result I just saw, that `pearsonr` rejected the null hypothesis of no relationship between `x` and `y`?

It is probably a false positive.

Does that mean that
`scipy.stats.pearsonr` is a bad test?

Is Covid RT-PCR a bad test?

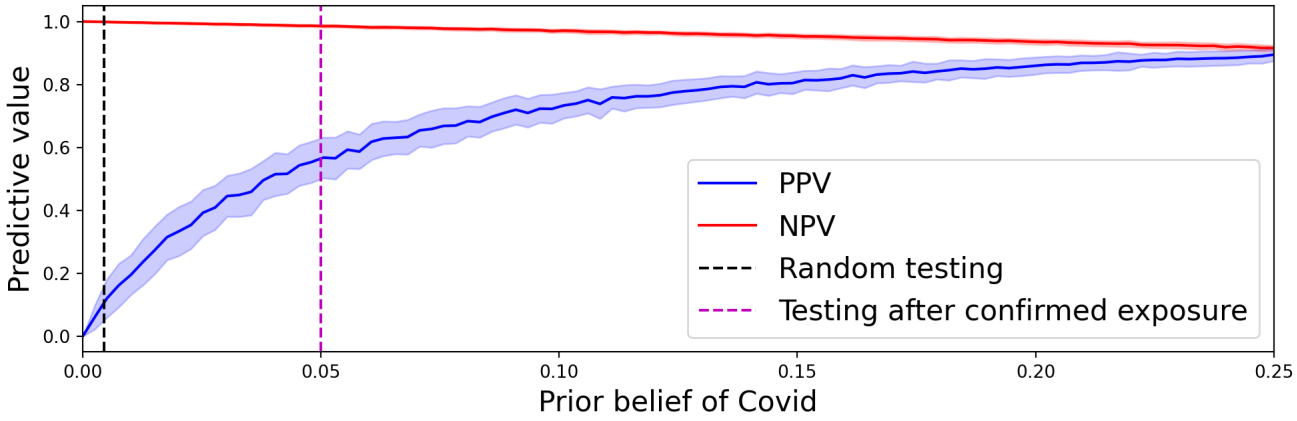
No.

PPV isn't just about the test.

PPV depends on **how the test is used**.

Recall

When looking at Covid testing, **increasing our prior belief of Covid infection to just 5%** raised the PPV from 5% to 60%, **for the exact same test**. To make the test's results more meaningful, we used **knowledge** of prior exposure to Covid to **choose who to test**.



Even very weak information can dramatically improve the utility of a test!

So, how do you **improve the positive predictive value** of testing for significant correlations?

Choose the right questions to ask.

A clairvoyant mouse?

You look at your data, and you see that you **reject the null hypothesis** that *manipulated mice do not predict the future* with a p-value of 0.014.

What do you now know about whether the mouse is actually clairvoyant?

Why do you think this?

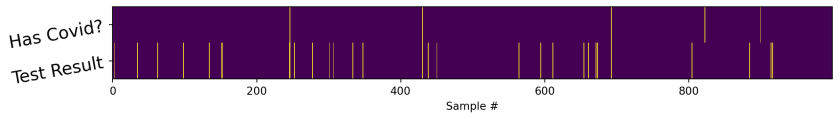
A clairvoyant mouse?

You look at your data, and you see that you **reject the null hypothesis** that *manipulated mice do not predict the future* with a p-value of 0.014.

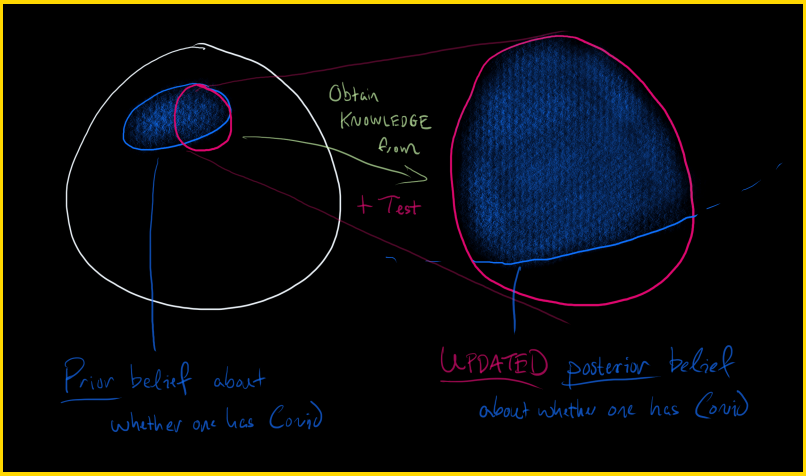
What do you do with this information?

Summary

Tests do not tell us the truth.



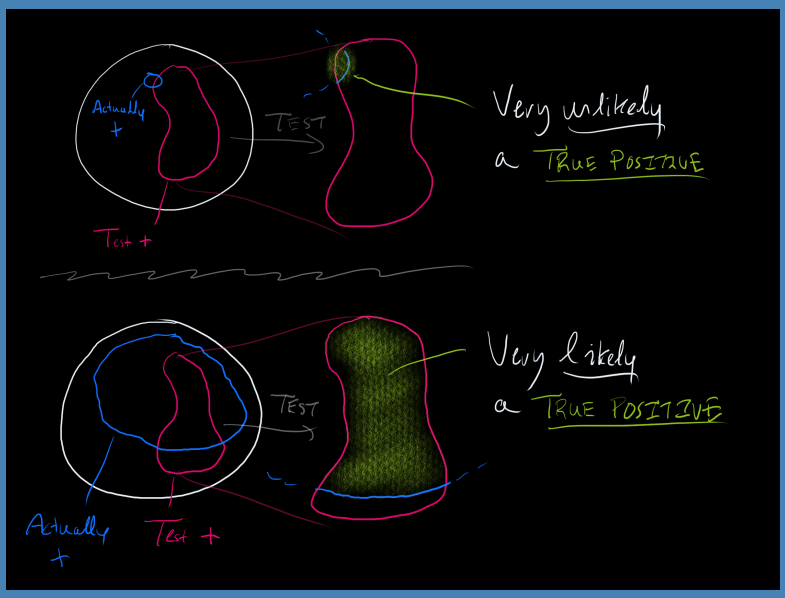
We try to infer the truth from tests.



Tests update our beliefs about the world.

Because we live in a special part of the multiverse—the part where we saw what we saw!

The predictive value of a test depends on how the test is used.



In particular, it depends on our prior knowledge of what we are testing for.

Statistical testing is not just a matter of selecting the "correct" test for a given question.

Being judicious about selecting the correct **questions** to apply the test **to** is a more powerful way to

- strengthen the inferences gained through statistics;
- improve the reproducibility of science; and
- more responsibly use resources, like experimental animals, that we have an ethical obligation to minimize.