Here are marks for IA Algorithms questions last year:

```
Women:   [17, 14, 18, 12, 17, …]
Men:     [18, 18, 11, 17, 17, …]
Other:   [17, 18,  9,  9, 11, …]
```

The mean marks are

Women:	13.22	(n=49)
Men:	12.28	(n=219)
Other:	13.10	(n=10)

Women do better. than either of the other two.

EXERCISE.
How would you critique this analysis?

- This doesn't report significance e.g. confidence.
- It's inappropriate to share this data, or to report unaggregated data for small-n categories.
- It's drawing a general conclusion ("women do better") from just one year of past data.
  (On the other hand, if we restricted ourselves to simply describing what <u>has</u> happened, and never said anything about the future, we'd never be able to influence the future.)

Made-up data

Based on the model

$$\text{Mark} \sim \mu_{\text{gender}} + N(0, \sigma^2)$$

the 95% confidence intervals are

$$\hat{\mu}_F \in [11.8, 14.6]$$
$$\hat{\mu}_M \in [11.6, 12.9]$$
$$\hat{\mu}_O \in [10.0, 16.2]$$

Women tend to do better than Men. There is too little data about Other to be confident in any comparison.

EXERCISE.
How would you critique this revised analysis?

- Marks are not independent (each student does 2 questions)
- A Gaussian dist. is inappropriate

If I want to report differences, I should report a conf. int. for differences.

Based on a model using one-hot coding of gender,

$$\text{Mark} \sim \mu_F + \delta_M 1_{\text{gender}=M} + \delta_O 1_{\text{gender}=O} + N(0, \sigma^2)$$

the 95% confidence intervals are

$$\hat{\mu}_F \in [11.8, 14.6]$$
$$\hat{\delta}_M \in [-2.5, 0.6]$$
$$\hat{\delta}_O \in [-3.6, 3.3]$$

Neither $\hat{\delta}_M$ nor $\hat{\delta}_O$ is convincingly non-zero.

EXERCISE.
How would you implement this analysis?

See Lecture 12 ....

gender	mark
F	17
F	14
M	18
M	11
M	17
⋮	⋮

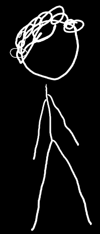
```
# The readout function
def t(marks):
    use sklearn.linear_model to fit the proposed model to marks
    return a triple with the intercept_ (μ_F) and the coef_ (δ_M,δ_O)

# To create a random synthetic dataset of marks
Let μ̂_F,δ̂_M,δ̂_O,σ̂ be the mle estimates from the marks column in the dataset
def rmarks():
    pred = μ̂_F + δ̂_M 1_gender=M + δ̂_O 1_gender=O
    return np.random.normal(loc=pred, scale=σ̂)

# Get lots of samples of the test statistic
t_ = [t(rmarks()) for _ in range(10000)]
np.quantile([θ[0] for θ in t_], [.025, .975])   # confint for μ_F
```

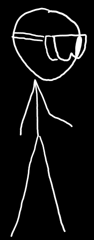
or, we could use nonparametric resampling.

How might we decide whether this simpler model is good enough?



I think everyone gets pretty much the same mark, regardless of gender.
$\text{Mark} \sim \mu + \text{Normal}(0, \sigma^2)$

To answer this, it can be helpful to introduce a richer model.



I think gender affects marks.
$\text{Mark} \sim \mu_{\text{gender}} + \text{Normal}(0, \sigma^2)$

	confidence intervals	**model selection**
FREQUENTIST (The answer might depend on how we resample.)	For just two genders: Consider the richer model with μ_{gender} and find a 95% confidence interval for $\hat{\mu}_M - \hat{\mu}_F$. $\mathbb{P}(\hat{\mu}_M - \hat{\mu}_F \in [-2.5, 0.6]) = 95\%$ so it looks like the simpler model is OK.	Hypothesis Testing
BAYESIANIST (The answer depends on our priors for the unknowns.)	For just two genders: Consider the richer model with μ_{gender} and find a 95% confidence interval for $\mu_M - \mu_F$. $\mathbb{P}(\mu_M - \mu_F \in [-3.1, -0.2]) = 95\%$ so it looks like the simpler model isn't good enough.	If we have prior weights for two models (the simple model, and the richer model with μ_{gender}), we can find posterior weights using Bayes's rule. For prior weights 50%/50%, the posterior weights are 79%/21% in favour of the simpler model.
	This is great if there's a single model parameter that we want to investigate	This is for when we want to evaluate the model as a whole

Bayesianist vs frequentist smackdown

Climate confidence challenge

Find a 95% confidence interval for the rate of temperate increase in Cambridge from 1985 to the present, in °C/year

§9.3 HYPOTHESIS TESTING

Can you taste the difference
between milk-first versus tea-first?

HYPOTHESIS: you can’t.

milk first	tea first

Fisher's hypothesis testing

the full data

Let x be the dataset.

State a null hypothesis H_0, i.e. a probability model for the dataset

1. Choose a test statistic
$$t : \text{dataset} \mapsto \mathbb{R}$$

2. Define a random synthetic dataset X^*, what we might see if H_0 were true.

3. Look at the histogram of $t(X^*)$, and let p be the probability of seeing a value as extreme or more so than the observed $t(x)$.

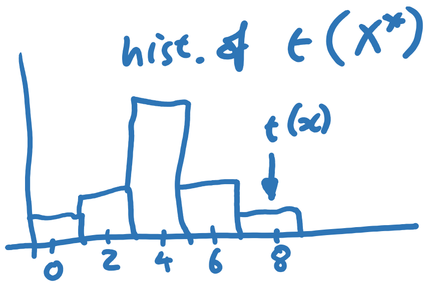
A low p-value is a sign that H_0 should be rejected.

x = taster's assignment of labels

H_0 : taster can't tell the difference,
hence assignment is a random permutation of $\{t,t,t,t,m,m,m,m\}$

$t(x)$ = # correct

def $X^*()$: return random perm of $\{t,t,t,t,m,m,m,m\}$



what would be the dist. of the test statistic, if H_0 were true?

$p = P(t(X^*) \geq t(x)) = 1.4\%$

$p < 5\%$: we'll reject H_0.

Example 9.6.2.

I have a dataset with readings from two groups, $x = [x_1, \ldots, x_m]$ and $y = [y_1, \ldots, y_n]$. Test whether the two groups are significantly different, using the test statistic $\bar{y} - \bar{x}$.

```
# 1. Define the test statistic
def t(x,y): return np.mean(y) - np.mean(x)

# 2. To generate a synthetic dataset, assuming H0, ...
xy = np.concatenate([x,y])
def rxy_star():
    return (np.random.choice(xy, size=len(x)),
            np.random.choice(xy, size=len(y)))

# 3. Sample the test statistic under H0; find p-value for observed data
t_ = np.array([t(*rxy_star()) for _ in range(10000)])
p = …
```

Example 9.3.1.
I have a dataset with readings from two groups, $x = [x_1, \dots, x_m]$ and $y = [y_1, \dots, y_n]$. Test whether the two groups are significantly different, using the test statistic $\bar{y} - \bar{x}$.

H_0: X_i, Y_i both $\sim N(\mu, \sigma^2)$

Equivalently,
assume $X_i \sim N(\mu, \sigma^2)$, $Y_i \sim N(\mu+\delta, \sigma^2)$

H_0: $\delta = 0$

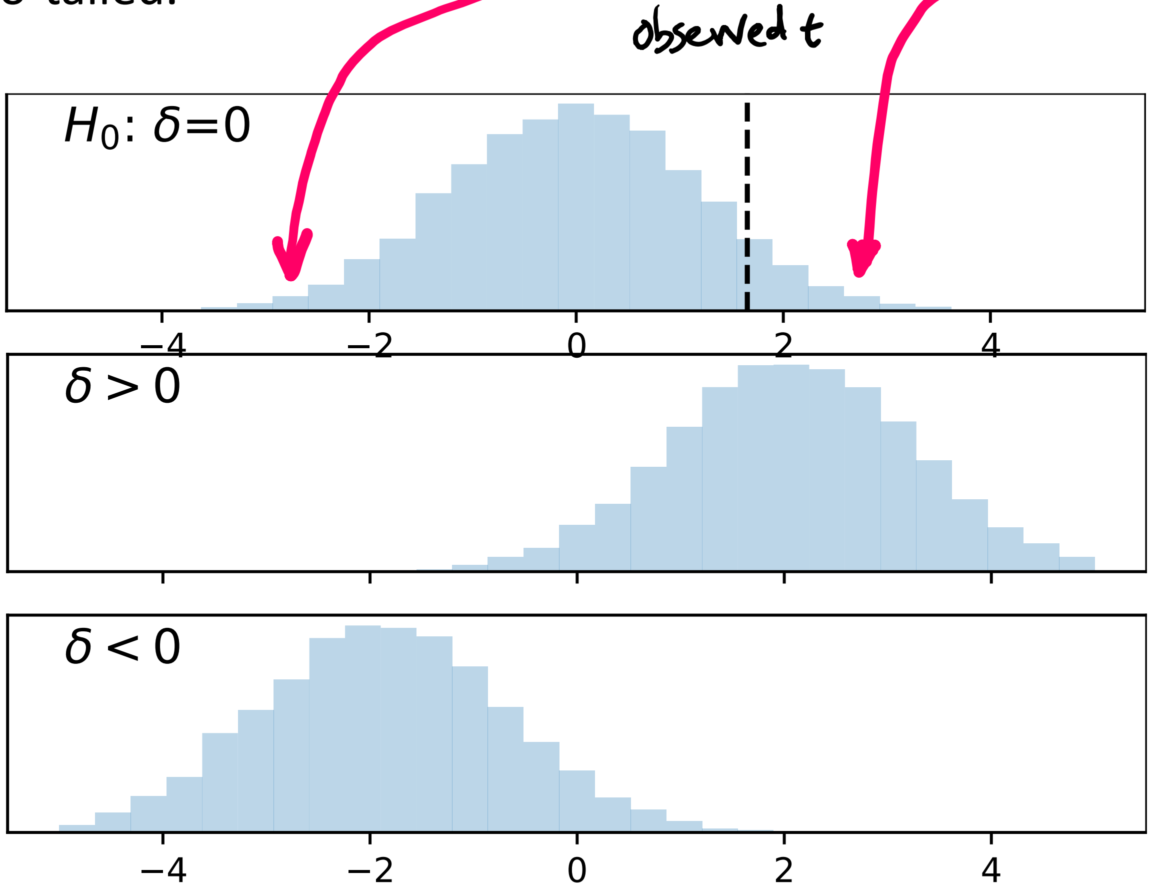
```
# 1. Define the test statistic
def t(x,y): return np.mean(y) - np.mean(x)

# 2. To generate a synthetic dataset, assuming H0, ...
xy = np.concatenate([x,y])
μ̂ = np.mean(xy)
σ̂ = np.sqrt(np.mean((xy - μ̂)**2))
def rxy_star():
    return (np.random.normal(loc=μ̂, scale=σ̂, size=len(x)),
            np.random.normal(loc=μ̂, scale=σ̂, size=len(y)))

# 3. Sample the test statistic under H0; find p-value for observed data
t_ = np.array([t(*rxy_star()) for _ in range(10000)])
p = 2 * min(np.mean(t_ >= t(x,y)), np.mean(t_ <= t(x,y)))
```

What counts as ‘more extreme’?

- Plot the histogram for $t(X^*)$, assuming H_0 is true
- Also plot the histogram for some scenarios where H_0 is false
- Do the alternatives push $t(X^*)$ bigger, or smaller, or either? This determines what ‘more extreme’ means — either one-tailed or two-tailed.

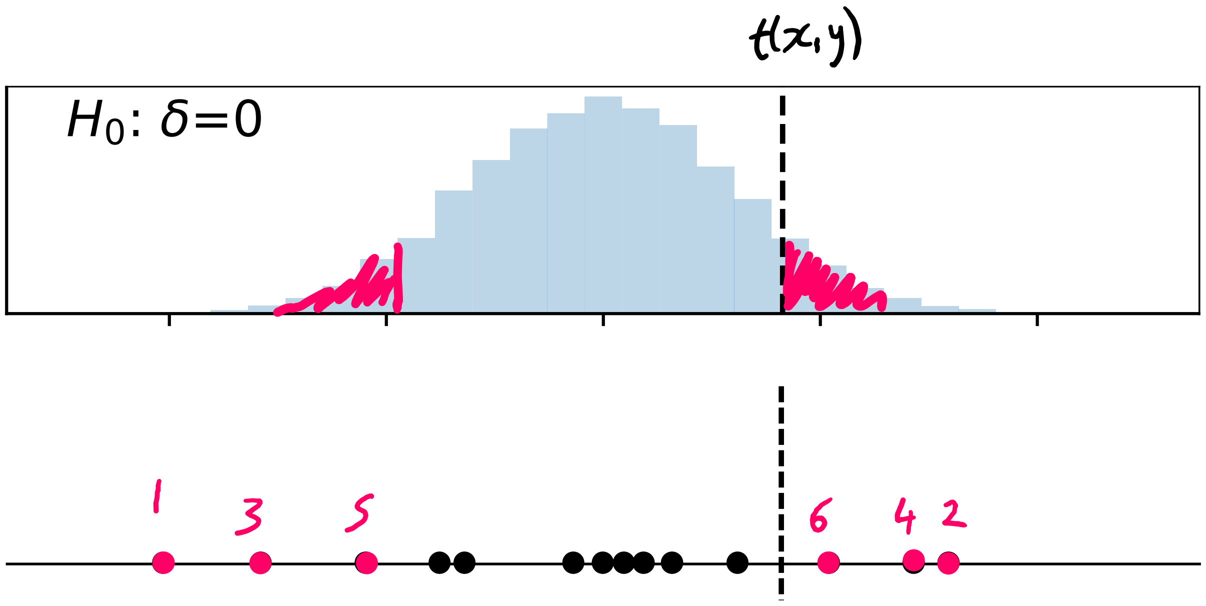


if the observed t lies at either extreme, it's evidence against H_0: δ=0.

How do we compute p for a two-tailed test?

The p-value is

$$\mathbb{P}\left(\begin{matrix} t(X^*) \text{ at least} \\ \text{as extreme as } t(x) \end{matrix} \;\middle|\; H_0 \text{ is true}\right)$$



"6 of my samples of t(X*,Y*) are more extreme than t(x,y)."

p = 2 * min(np.mean(**t_** >= t(**x**,**y**)), np.mean(**t_** <= t(**x**,**y**)))

The beauty of hypothesis testing is that it lets us test whether H_0 is a good enough model for the data, without our having to specify an alternative model. Instead, we specify a test.

Where do test statistics come from?

There are two common scenarios, exploratory and rhetorical.

EXPLORATORY.
You, the modeller, are trying to come up with a good model for the dataset. Suppose you've tried out several models, and H_0 is the best you've come up with. Is it good enough?

- If you settle for H_0 and someone else comes up with a better model, you lose.
- So it's up to you to creatively think up ways to test if H_0 might be deficient.

RHETORICAL.
Sometimes, there's a model H_1 that everyone accepts to be the natural alternative to H_0.

- Example: H_0 = "my drug makes no difference", H_1 = "it makes a difference".
- If so, craft the test statistic to look for evidence pointing in the direction of H_1.