7. Frequentism I: model readouts

According to the frequentist school of inference, when we analyse a dataset we should consider the 'counterfactual multiverse' of all the possible ways the dataset might have turned out but didn't.

To be more concrete, let x be the dataset we observe, and let X be a random variable representing all the ways that the dataset might have turned out. Imagine an infinite collection of parallel universes[20], each with its own dataset drawn from distribution X. Perhaps we computed a maximum likelihood estimate using the dataset we have—then in each parallel universe there will be a parallel data scientist who has computed their own estimate, and who is to say that ours is better than theirs? If all the estimates end up very close to each other then we should be confident in the result, and if there's a wide spread then we should not be confident.

The data scientist doesn't just see the data sitting in the database. Like Dr Strange from Marvel Comics, they see 14,000,605 ways that things might have turned out. Not very useful if you're tallying up your bank account for the past month and working out whether you're overdrawn. Very useful if you're making forecasts or inferences.

The difficulty, of course, is that we're stuck inside our own universe and we can't actually see the outcomes in all the parallel universes. As frequentists, we therefore need tools that allow us to use of the data we have available in our universe, in order to reason about the multiverse. Dr Strange really could see the true distribution of X, but the humble frequentist has to find work-arounds to simulate it.

[20]The word 'frequentism' comes from interpreting $\mathbb{P}(X = x)$ as the frequency of outcome x across all these parallel universes.

7.1. Confidence intervals / resampling

Let's suppose we want to report some quantity $h(x)$ that we've computed from the dataset x.

This might be the maximum likelihood estimator for the unknown parameter. Or perhaps our probability model has multiple unknown parameters, and $h(x)$ is the estimator for just one of them. Or perhaps $h(x)$ isn't even based on maximum likelihood estimators at all, perhaps it's just some arbitrary data summary. (The technical term for all of these cases—for any quantity that's a function of the observed data and doesn't involve unknown parameters—is *statistic*.)

Instead of reporting just the value of h computed on the dataset we observed, we should do our best to simulate the parallel universe datasets, and report the spread of values of h. Here are two common approaches for synthesizing such a dataset by leveraging the data we actually have. They're both known as *resampling*.

> *You're writing up a news story about knife crime. You write "A 95% confidence for the mean number of attacks is* $[51, 120]$*". Your colleagues look at you as if you're crazy. One of them, a self-styled data wizard, runs an database query* `SELECT AVG(num_attacks) FROM offences WHERE offence="knife"` *and gets the answer 85, and wonders why you can't give a straight answer to a straight question.*

Let our dataset x consist of observations $(x_1, x_2, \ldots, x_n)$. Here are two ways to generate a random resampled dataset X^*.

- Non-parametric resampling. Let X^* consist of n items chosen with replacement from $(x_1, \ldots, x_n)$. In other words, X^* is a sample of size n drawn from the empirical distribution of the dataset. (As discussed in section 4.2, the best-fitting distribution for a dataset is the dataset itself, i.e. the empirical distribution.)
- Parametric resampling. Suppose we have a probability model $\Pr(x_i \mid \theta)$ for the data, where θ is an unknown parameter, and the observations represent independent samples. First, fit the model i.e. find the maximum likelihood estimator $\hat{\theta}$ from the data we have. Now, we generate a synthetic dataset X^* by sampling n items, generated randomly from our fitted model.

We should generate a large number of synthetic datasets, and compute h on each, and report the spread of values we see.

empirical distribution, section 4.2

There's no universal rule for the best way to simulate the multiverse. How could there be—how could we deduce *what might have happened* when all we know is *what actually happened*? You should think about where the data came from, and how it was collected.

A standard way to express the spread of values of $h(X^*)$ is as a 95% confidence interval, i.e. an interval $[\mathsf{lo}, \mathsf{hi}]$ such that

$$\mathbb{P}\big(h(X^*) \in [\mathsf{lo}, \mathsf{hi}]\big) = 0.95.$$

A common choice is a two-sided interval, where

$$\mathbb{P}\big(h(X^*) < \mathsf{lo}\big) = 0.025 \quad \text{and} \quad \mathbb{P}\big(h(X^*) > \mathsf{hi}\big) = 0.025.$$

There's a handy library routine to find these bounds computationally: if we've sampled a list of values `h_sample` from $h(X^*)$, then all we need is

$$\mathsf{lo},\mathsf{hi} = \mathsf{numpy.quantile}(\mathsf{h\_sample}, [0.025, 0.975])$$

Other choices for the confidence interval are just as legitimate: for example if our statistic $h(x)$ is always ≥ 0 and we hope it's small, then we might report a one-sided interval $[0, \mathsf{hi}]$ with hi chosen so that $\mathbb{P}(h(X^*) > \mathsf{hi}) = 0.05$.

Example 7.1 (Parametric resampling).
We are given a dataset $7.2, 7.3, 7.8, 8.2, 8.8, 9.5$ which we believe is drawn from the Normal(μ, σ^2) distribution, where μ and σ are unknown. The maximum likelihood estimator $\hat{\mu}$ is just the sample mean, 8.13 in this case. Find a 95% confidence interval for $\hat{\mu}$.

Write the dataset as $x_1, \ldots, x_n$. We found formulae for the maximum likelihood estimators $\hat{\mu}$ and $\hat{\sigma}$ in section 1.5 page 15:

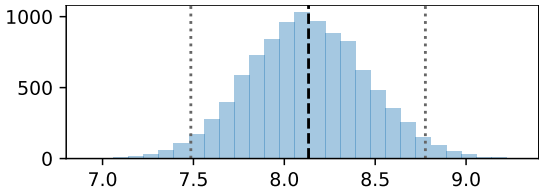
$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i\,, \quad \hat{\sigma} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{\mu})^2}\,.$$

We can synthesize a new dataset X^* by simply calling a Normal$(\hat{\mu}, \hat{\sigma}^2)$ *random number generator n times, and then compute $\hat{\mu}(X^*)$. Now, repeat this to get 10,000 samples of $\hat{\mu}(X^*)$.*

```
x = [7.2, 7.3, 7.8, 8.2, 8.8, 9.5]
def μ̂(x): return numpy.mean(x)
def σ̂(x): return numpy.sqrt(numpy.mean(numpy.power(x − μ̂(x), 2)))
def rx_star(): return numpy.random.normal(size=len(x), loc=μ̂(x), scale=σ̂(x))
μ̂_sample = [μ̂(rx_star()) for _ in range(10000)]
```

We can plot a histogram of $\hat{\mu}(X^*)$, or report a confidence inverval.

```
lo,hi = numpy.quantile(μ̂_sample, [.025, .975])  # [7.48, 8.79]
plt.hist(μ̂_sample, bins=30, alpha=.4)
plt.axvline(lo, linestyle='dotted', color='0.4')
plt.axvline(hi, linestyle='dotted', color='0.4')
plt.axvline(μ̂(x), linestyle='dashed', color='black')
```



■

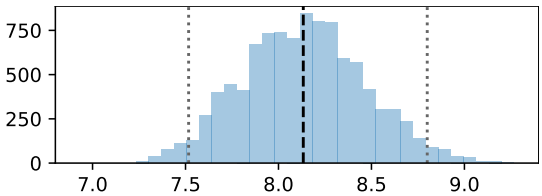
Example 7.2 (Non-parametric resampling).
We are given a dataset $7.2, 7.3, 7.8, 8.2, 8.8, 9.5$ and we computed the sample mean $\bar{x} = 8.13$. Find a 95% confidence interval.

Remember, $\bar{x}$ is a number, not a random variable, so it doesn't really have *a confidence interval. The same is true of $\hat{\mu}(x)$ in the previous exercise: it's a number, not a random variable. What the question is asking is: if we were to collect a new dataset, and compute a new sample mean, what's a range in which this new value is likely to fall?*

To sample n items with replacement from a list x, use numpy.random.choice(x, size=n).

sampling commands: section 1.3.1 page 8

```
x = [7.2, 7.3, 7.8, 8.2, 8.8, 9.5]
def rx_star(): return numpy.random.choice(x, size=len(x))
xbar_sample = [numpy.mean(rx_star()) for _ in range(10000)]
lo,hi = numpy.quantile(xbar_sample, [.025, .975])  # [7.52, 8.80]
```



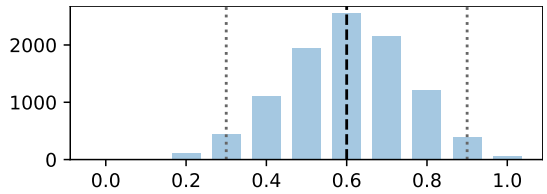
■

Example 7.3 (Parametric resampling).
I toss $n = 10$ coins and get $x = 6$ heads. Using the probability model $X \sim \text{Binom}(n, p)$ where X is the number of heads, and $n = 10$ and p is unknown, I estimate that the probability of heads is $\hat{p} = x/n = 0.6$.

Find a 95% confidence interval for $\hat{p}$.

Write the maximum likelihood estimator as $\hat{p}(x)$, to emphasize that it's a statistic computed from the data x. We can synthesize a new outcome X^* by generating a $\text{Binom}\big(n, \hat{p}(x)\big)$ random variable.

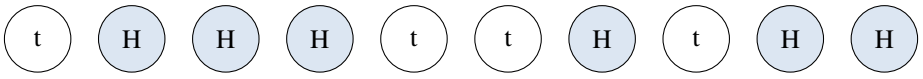
```
x,n = (6,10)
p̂ = x/n
def rx_star(): return numpy.random.binomial(n=n, p=p̂)
p̂_sample = [rx_star()/n for _ in range(10000)]
lo,hi = numpy.quantile(p̂_sample, [.025, .975])  # (0.3, 0.9)

p,c = numpy.unique(numpy.array(p̂_sample), return_counts=True)
plt.bar(p,c, width=0.07, alpha=.4)
plt.axvline(lo, linestyle='dotted', color='0.4')
plt.axvline(hi, linestyle='dotted', color='0.4')
plt.axvline(p̂, linestyle='dashed', color='black')
```



∎

Example 7.4 (Non-parametric resampling).
I toss $n = 10$ coins and get $x = 6$ heads. I estimate that the probability of heads is $x/n = 0.6$. Find a 95% confidence interval for this.

Non-parametric resampling means "pick a value at random from the empirical distribution". All we're given in this question is a single observation $x = 6$, so the empirical distribution is too simple—every time we resample from it we get the same answer, $X^* = 6$. But there's a better way to look at the observed data: look at it not as a count, but as a full-length dataset with 10 records describing the outcomes of 10 coin tosses.



To resample a single coin toss, we pick one of these 10 outcomes at random, and get heads with probability $6/10$. To resample n coin tosses, we pick n of these at random, with replacement. The number of heads from n resampled coin tosses is $\text{Binom}(n, 6/10)$, which is in fact exactly the same answer we got from parametric resampling in example 7.3.

∎

7.2. Measuring differences

Here's typical setting in which we want to measure our confidence in an estimate. Suppose we have two systems, A and B, and we've measured outcomes under both systems, and we want to know which leads to better outcomes. For example, A and B might be different placing of an ad and we're measuring the number of click-throughs; or they might be different compilers and we're measuring compilation time; or they might be placebo and drug and we're measuring the drug's effect size. How do we estimate the difference between the two? More importantly, if the choice between A and B has financial or ethical implications: how confident can we be in our estimate?

Estimating differences is in fact just another application of the tools from the previous section, namely confidence intervals and resampling. Here are two general approaches (though they are just illustrations, and many variations are possible). Suppose we have a dataset $x = (x_1, \ldots, x_m)$ from system A, and $y = (y_1, \ldots, y_n)$ from system B.

- Non-parametric resampling. First decide on a metric for the difference, for example the mean difference $\delta(x, y) = \bar{x} - \bar{y}$. Then create a synthetic dataset X^* by resampling from the x values, and another synthetic dataset Y^* by resampling from the y values, and compute $\delta(X^*, Y^*)$. Repeat this many times, and report the spread of values.

- Parametric resampling. Formulate a probability model to describe both datasets simultaneously. The model should have some parameters that differ between the two datasets, though it may also have parameters that are shared. Decide on a metric for the difference, which should be a function of the parameters. For example, our model might have parameters μ_A and μ_B for the two datasets, and the metric might be $\delta = |\mu_A - \mu_B|$.

  Fit the model, i.e. find the maximum likelihood estimators for all unknown parameters. Use the fitted model to synthesize new datasets X^* and Y^*, compute the maximum likelihood estimators for these datasets, and thence compute the metric of interest[21]. Repeat this many times, and report the spread of values.

> Example 7.5 (Parametric resampling).
> We are given a dataset $x = (7.2, 7.3, 7.8, 8.2, 8.8, 9.5)$ which we believe is drawn from the Normal(μ, σ^2) distribution, and $y = (8.3, 8.5, 9.2)$ which we believe is drawn from Normal$(\mu + \delta, \sigma^2)$. Find the maximum likelihood estimator for δ, and a 95% confidence interval.

Let's use parametric resampling. The first step is to find maximum likelihood estimators for all the unknown parameters.

When we write out the likelihood, the rule is that we include all the data and all the unknown parameters. In this question, "all the data" means "all the x_i and all the y_i", and the unknown parameters are μ, δ, and σ. So the likelihood function is

$$\begin{aligned}
&\text{lik}(\mu, \delta, \sigma \mid x_1, \ldots, x_m, y_1, \ldots, y_n) \\
&\quad = \Pr(x_1, \ldots, x_m, y_1, \ldots, y_n \mid \mu, \delta, \sigma) \\
&\quad = \Pr_X(x_1 \mid \mu, \sigma) \times \cdots \times \Pr_X(x_m \mid \mu, \sigma) \times \Pr_Y(y_1 \mid \mu, \delta, \sigma) \times \cdots \times \Pr_Y(y_n \mid \mu, \delta, \sigma) \\
&\qquad \textit{where } X \sim \text{Normal}(\mu, \sigma^2) \textit{ and } Y \sim \text{Normal}(\mu + \delta, \sigma^2), \\
&\qquad \textit{and assuming all observations are independent} \\
&\quad = \left(\prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x_i - \mu)^2 / 2\sigma^2} \right) \left(\prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y_i - \mu - \delta)^2 / 2\sigma^2} \right).
\end{aligned}$$

Taking logarithms, to make the algebra easier,

$$\log \text{lik}(\mu, \delta, \sigma) = -\frac{m+n}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \left\{ \sum_{i=1}^{m} (x_i - \mu)^2 + (y_i - \mu - \delta)^2 \right\}.$$

To find the maximum likelihood estimators, we need to solve three simultaneous equations:

$$\frac{\partial}{\partial \mu} \log \text{lik} = 0, \qquad \frac{\partial}{\partial \delta} \log \text{lik} = 0, \qquad \frac{\partial}{\partial \sigma} \log \text{lik} = 0.$$

[21] By the plug-in principle, page 2, the maximum likelihood estimator for a derived quantity $\delta = f(\mu_A, \mu_B)$ is just the function applied to the maximum likelihood estimators, $\hat{\delta} = f(\hat{\mu}_A, \hat{\mu}_B)$. Therefore, in parametric resampling, we're resampling the maximum likelihood estimator for the metric of interest, $\hat{\delta}$.

Solving the first two we get $\hat{\mu} = \bar{x} =$ and $\hat{\delta} = \bar{y} - \bar{x}$, where $\bar{x}$ denotes the sample mean $\sum_i x_i/m$ and $\bar{y} = \sum_i y_i/n$. Solving the third equation gives

$$\hat{\sigma} = \sqrt{\frac{1}{m+n}\left\{\sum_{i=1}^{m}(x_i - \hat{\mu})^2 + \sum_{i=1}^{n}(y_i - \hat{\mu} - \hat{\delta})^2\right\}}.$$

(You may spot that this is a special case of linear regression. We could have skipped all this algebra and gone straight for the computational solution: create a vector xy *by concatenating x and y, create a vector* sys *consisting of m copies of* "A" *followed by n of* "B"*, then fit the linear model* xy $\approx \mu + \delta 1_{\text{sys}=\text{"}B\text{"}}$*. See section 2.2.1 for one-hot coding, and section 2.4 for the link between linear models and Gaussian probability models.)*

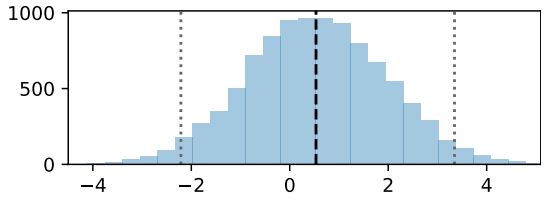
```
x = numpy.array([7.2, 7.3, 7.8, 8.2, 8.8, 9.5])
y = numpy.array([8.3, 8.5, 9.2])
m,n = len(x), len(y)
μ̂ = numpy.mean(x)
δ̂ = numpy.mean(y) - numpy.mean(x)
σ̂ = numpy.sqrt(numpy.sum((x - μ̂)**2) + numpy.sum((y - μ̂ - δ̂)**2)/(m+n))
```

The second step is to use resampling to synthesize new datasets X^* and Y^*, and thence new values for $\hat{\delta}(X^*, Y^*)$:

```
def rxy_star():
    return (numpy.random.normal(loc=μ̂, scale=σ̂, size=m),
            numpy.random.normal(loc=μ̂+δ̂, scale=σ̂, size=n))
def δ̂_star():
    x_star,y_star = rxy_star()
    return numpy.mean(y_star) - numpy.mean(x_star)
δ̂_sample = [δ̂_star() for _ in range(10000)]
```

Now we can report the spread of values of $\hat{\delta}(X^*, Y^*)$, for example by showing a histogram and a 95% confidence interval.

```
lo,hi = numpy.quantile(δ̂_sample, [.025, .975])
plt.hist(deltahat_sample, bins=30, alpha=.4)
plt.axvline(lo, linestyle='dotted', color='0.4')
plt.axvline(hi, linestyle='dotted', color='0.4')
plt.axvline(δ̂, linestyle='dashed', color='black')
```



∎

Exercise 7.6 (Parametric resampling, indirect).
For the data in exercise 7.5, suppose that the x values are drawn from Normal(μ_x, σ^2) and the y values are drawn from Normal(μ_y, σ^2). Estimate the difference $\delta = \mu_y - \mu_x$, and report a 95% confidence interval.

Using similar maths to the previous exercise, the maximum likelihood estimators are

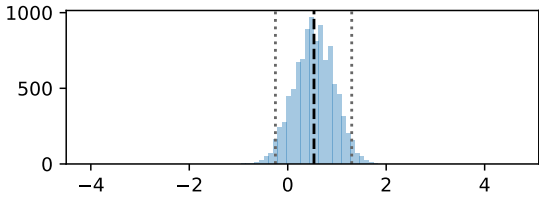
$$\hat{\mu}_x = \bar{x}, \quad \hat{\mu}_y = \bar{y}, \quad \hat{\sigma} = \sqrt{\frac{1}{m+n}\left\{\sum_{i=1}^{m}(x_i - \hat{\mu}_x)^2 + \sum_{i=1}^{n}(y_i - \hat{\mu}_y)^2\right\}}.$$

After a little thought, we realize that the resampling code we have to write here is nearly identical to the resampling code we wrote for the previous exercise. There's only a minor difference in variable names: instead of $\hat{\mu}$ and $\hat{\delta}$ we have $\hat{\mu}_x$ and $\hat{\mu}_y - \hat{\mu}_x$.

∎

Example 7.7 (Non-parametric resampling).
We are given datasets $x = (7.2, 7.3, 7.8, 8.2, 8.8, 9.5)$ and $y = (8.3, 8.5, 9.2)$. The difference between the two sample means is $\bar{y} - \bar{x} = 0.53$. Find a 95% confidence interval using non-parametric resampling.

```
x = [7.2, 7.3, 7.8, 8.2, 8.8, 9.5]
y = [8.3, 8.5, 9.2]

def rxy_star():
    return (numpy.random.choice(x, size=len(x)),
            numpy.random.choice(y, size=len(y)))
def d(xy):
    x_star,y_star = xy
    return numpy.mean(y_star) - numpy.mean(x_star)
d_sample = [d(rxy_star()) for _ in range(10000)]

lo,hi = numpy.quantile(d_sample, [.025, .975])
plt.hist(d_sample, bins=30, alpha=.4)
plt.axvline(lo, linestyle='dotted', color='0.4')
plt.axvline(hi, linestyle='dotted', color='0.4')
plt.axvline(d((x,y)), linestyle='dashed', color='black')
```

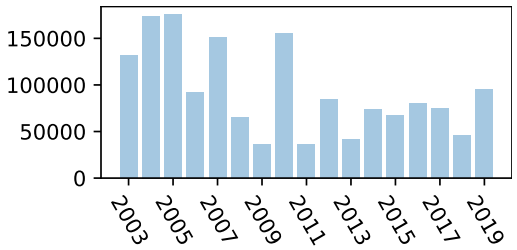


Why is the answer here so different to exercise 7.5? Note that the y datapoints are clustered tightly together, more tightly than the x datapoints. If we resample X^* and Y^* separately, as we did here, then we'll end up seeing very little variation in $\overline{Y^*}$. If we use a parametric model like that of exercise 7.5, in which the two datasets are assumed to share a common variance, then the wider spread of x values will bleed through into higher variation in $\overline{Y^*}$, leading to higher variation in $\overline{Y^*} - \overline{X^*}$.

Which is correct? That's something that we can't answer from the data, we can only answer from the context. Do we have reason to believe that the variability should *be the same in both cases? Anyway, resampling from a dataset of size $n = 3$ is sure to cause problems of interpretation—there's just too little information to make sound inferences.*

■

Exercise 7.8 (Amazon fires).
The fires in the Amazon in summer 2019 were widely reported. According to satellite data from MODIS[22], the number of fires detected from the beginning of the year to 29 August 2019 was 95092, more than double that seen in the same period the previous year. The chart below shows the corresponding numbers over a span of years.



We wish to know if the 2019 value is notably higher than the values since 2011, when Dilma

[22]Data from https://www.globalfiredata.org/forecast.html, csv at https://teachingfiles.blob.core.windows.net/datasets/modis.csv

Rousseff took office as president of Brazil. Assuming that the 2011–2018 values are Normal(μ, σ^2), and the 2019 value is Normal$(\mu + \delta, \sigma^2)$, find a 95% confidence interval for the maximum likelihood estimator $\hat{\delta}$.

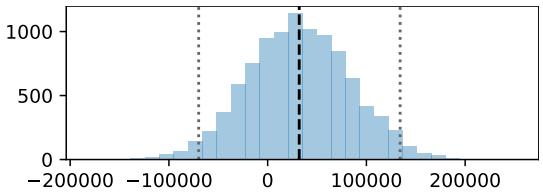
First, here is the code behind the plot. The full dataset includes cumulative number of fires on each day of the year; day 240 is 29 August.

```
modis = pandas.read_csv('https://teachingfiles.blob.core.windows.net/datasets/modis.csv')
df = modis[modis.day_of_year==240]
plt.bar(df.year.values, df.num_fires, width=0.8, alpha=.4)
plt.xticks(numpy.arange(2003,2020,2), rotation=-60)
```

The analysis is exactly the same as for exercise 7.5.

```
x = df.num_fires[(df.year>=2011) & (df.year<=2018)].values
y = df.num_fires[df.year==2019].values
...
```

In the end, the maximum likelihood estimate for δ is 31,000, and the 95% confidence interval is $[-70,000,\ 134,000]$. This is does not support the conclusion that 2019 is notably different to preceeding years.



■

Exercise 7.9.

An exam question was attempted by 68 students from the Computer Science (CS) course, and 40 students from the Natural Sciences (NST) course. The question was marked out of 20. The number of students who attained each mark were

mark	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
CS	0	0	0	2	2	2	1	0	3	2	4	8	6	2	8	8	4	4	6	4	2
NST	1	0	0	1	1	0	1	4	2	0	2	2	8	6	4	3	1	2	0	1	1

The mean mark for CS students is 13.0, and that for NST students is 11.6, a difference of 1.5. Find a 95% confidence interval for the difference.

It's not immediately obvious what probability distribution we might use for this data, so let's use the non-parametric approach.

The data given in the question is pre-aggregated (i.e. it shows the number of students for each mark), but it's perhaps easier to work with unaggregated data. In the code below, cs_star() *generates 68 unaggregated marks for CS students, drawn from the empirical distribution, and* nst_star() *generates 40 unaggregated marks for NST students.*

```
csn = numpy.array([0,0,0,2,2,2,1,0,3,2,4,8,6,2,8,8,4,4,6,4,2])
nstn = numpy.array([1,0,0,1,1,0,1,4,2,0,2,2,8,6,4,3,1,2,0,1,1])
m = numpy.arange(0,21)
def cs_star(): return numpy.random.choice(m, p=csn/numpy.sum(csn), size=numpy.sum(csn))
def nst_star(): return numpy.random.choice(m, p=nstn/numpy.sum(nstn), size=numpy.sum(nstn))
```

Now we simply resample the marks, and for each resampled dataset we record the mean difference. The 95% confidence interval turns out to be $[-0.17, 3.15]$. This suggests that CS students might be cleverer, but the evidence isn't compelling since the confidence interval includes 0.

```
diff_sample = [numpy.mean(cs_star()) - numpy.mean(nst_star()) for _ in range(10000)]
lo,hi = numpy.quantile(diff_sample, [.025, .975])  # returns (-0.17, 3.15)
```

■

7.3. Hypothesis testing and p-values

It's often useful to frame questions as *I have a default model, which I'll stick with unless the evidence says otherwise*. The default model is also known as the *null hypothesis* and written H_0.

For example, "My default assumption is that this proposed drug is ineffective, and the NHS shan't prescribe it unless there is evidence to the contrary." Or "My default assumption is that the fancy new machine learning algorithm is no different to the older simpler one, and the journal I edit will reject a paper proposing the new algorithm unless there's evidence that it's an improvement." Or "I will give the police the benefit of the doubt and assume they are not racially biased, unless there is evidence to persuade me otherwise."

We saw in section 7.2 how to find a confidence interval for the difference in outcomes between two groups. That's the best tool to use, if we're looking to measure a difference. But in other situations it's insufficient:

- If we ask "is there evidence of police bias?" we're not looking for the difference between just two groups—the police dataset from page 35 lists five different ethnic groups, and a difference between *any* pair of these is evidence of bias. The hypothesis-testing approach lets us ask composite questions about multiple groups.
- If we have outcome measurements from two systems A and B, and we ask "does A have better outcomes than B?", then confidence intervals are the right approach. But we could also ask the broader question "does A have *different* outcomes to B?" For example, A might have greater variability than B, even if they have the same mean outcome. The hypothesis-testing approach lets us ask composite questions about multiple dimensions of difference.

Here's a general purpose procedure for hypothesis testing, due to Ronald Fisher.[23]

Fisher's hypothesis testing procedure. Let x be the full dataset, and let H_0 be our null hypothesis.

1. Define a function $t : x \mapsto \mathbb{R}$, called the *test statistic*. It can be whatever function of the data we like (or we are told to use). Just make sure it's a function *only* of the observable data—don't let it use any unknown parameters, because they're unknown!
2. Create synthetic datasets X^* using resampling, either parametric or non-parametric. The important thing is to use a resampling scheme that fits with the null hypothesis H_0. Illustrations are given below.
3. Compute the distribution of the test statistic $t(X^*)$. Plot a histogram of $t(X^*)$, and mark on it the actual observed value $t(x)$. Draw whatever inferences are appropriate. (If $t(x)$ lies at an extreme end of the histogram, it's a sign that something fishy is going on.)

This hypothesis testing procedure is generic. It doesn't tell us how to pick a test statistic, nor how to resample, nor what conclusions are appropriate. In our first illustration, the summer 2019 fires in the Amazon, we're told all these things. In other problems, it's useful to have guidelines …

Amazon fires: exercise 7.10 page 83

Step 1 — the null hypothesis. In many problems, a useful starting point is to write down a general model with many parameters, and to express H_0 as a restriction on the parameters. For example, we might have measurements for individuals from several different groups, and we wish to know whether the groups are all similar: we'd write down a general model with different parameters for each group, and write H_0 as "the parameters for the different groups are all equal". Our second illustration, testing equality of group means, goes into detail.

testing equality of group means: exercise 7.11 page 83

In problems like this, it's sensible to choose a test statistic[24] that's based on maximum likelihood estimators for the general model. Our third illustration, testing for racial bias in policing, illustrates.

testing for racial bias: exercise 7.12 page 84

Step 3 — the alternative hypotheses. To decide what counts as 'extremely fishy', we need to spell out the alternatives to H_0, and consider the likely range of values of the test statistic under these

[23] Ronald Fisher, the "genius who almost single-handedly created the foundations for modern statistical science." He was head of the Department of Eugenics at UCL, and later president of Gonville and Caius in Cambridge. The college has a stained glass window depicting his statistical work.

[24] For the purposes of this course, we'll work with ad hoc test statistics. There are systematic ways to design good test statistics, with close connection to Kullback-Leibler divergence, which you can read about in the extended version of these lecture notes.

alternatives. This guides us to appropriate inferences, and in particular it lets us compute the so-called p-value. Here is the general approach. It is illustrated in our final example, the sign test.

sign test: example 7.13
page 86

Neyman–Pearson approach. Let x be the full dataset, and let H_0 be our null hypothesis.

1. As before, define a test statistic $t : x \mapsto \mathbb{R}$, and create synthetic datasets X^* using resampling under the null hypothesis H_0. Plot a histogram of $t(X^*)$ and mark on $t(x)$
2. Propose an alternative hypothesis H_1. Create resampled versions of the dataset under H_1, and plot a histogram. Repeat for as many alternatives as you have in mind.
3. Use the alternative histograms to decide what counts as an extreme value on the original histogram:
   - If all the alternative histograms are shifted right, then large values of $t(x)$ are grounds for rejecting H_0. This is called a one-sided test.
   - If all the alternative histograms are shifted left, then small values of $t(x)$ are grounds for rejecting H_0. This is also a one-sided test.
   - If some of them shift left and others shift right, then extreme values of $t(x)$ on either side are grounds for rejecting H_0. This is called a two-sided test.
4. Report the probability (assuming H_0 is true) of seing a test statistic at least as extreme as the value $t(x)$ that we actually saw. This is called the p-value.

ILLUSTRATIONS

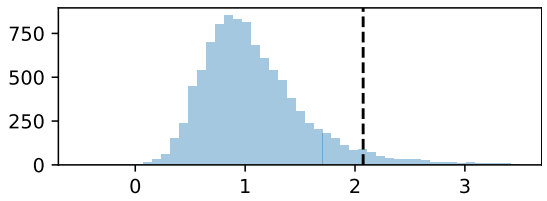
Exercise 7.10 (Amazon fires).
For the Amazonian fire data on page 79, let x_i be the number of fires recorded for year i. The media reported $t = x_{2019}/x_{2018}$, the year-on-year increase in number of fires, and claimed that this value was worryingly large.

Consider the null hypothesis that $(x_{2011}, \ldots, x_{2019})$ are drawn independently from the same distribution, say Normal(μ, σ^2) where μ and σ are to be estimated from the data. Under this hypothesis, what is the probability of seeing a value as large as or larger than t?

The question indicates that we are to use parametric resampling and suggests the Normal distribution. (Parametric resampling is the best choice for problems like this with very few datapoints. Whether or not a Normal distribution is appropriate is something that can be answered either by a domain expert, or a data scientist with much more data.)

Following the parametric resampling approach described in example 7.1 page 75, we first find maximum likelihood estimators $\hat{\mu}$ and $\hat{\sigma}$, and then generate new observations using these parameters.

```
# Load the dataset, and extract the x_i we want as a numpy vector
modis = pandas.read_csv('https://teachingfiles.blob.core.windows.net/datasets/modis.csv')
df = modis[modis.day_of_year==240]
x = df.loc[df.year>=2011].sort_values('year').num_fires.values

# The parametric resampling procedure
μ̂ = numpy.mean(x)
σ̂ = numpy.sqrt(numpy.mean((x-μ̂)**2))
def rx_star(): return numpy.random.normal(size=len(x), loc=μ̂, scale=σ̂)

# The test statistic
def t(x): return x[-1] / x[-2]
t_sample = numpy.array([t(rx_star()) for _ in range(10000)])

# What we'd expect to see if H_0 was true
plt.hist(t_sample, bins=np.linspace(-.5,3.5,50), alpha=.4)
plt.axvline(t(x), linestyle='dashed', color='black')
```



The actual observed year-on-year increase, $t(x) = 2.07$, is very large compared to what we'd expect to see. The probability of seeing a year-on-year increase this large or larger is

```
p = numpy.mean(t_sample>=t(x))  # 0.0511
```

This is known as the p-value.
∎

Exercise 7.11 (Equality of group means).
We are given three groups of observations from three different systems,

$$x = [7.2, 7.3, 7.8, 8.2, 8.8, 9.5]$$
$$y = [8.3, 8.5, 9.2]$$
$$z = [7.4, 8.5, 9.0],$$

and we wish to know whether they all come from the same distribution, or whether there are three different distributions. Start with a general probability model in which they could potentially

come from three different distributions,

$$X_i \sim \text{Normal}(a, \sigma^2), \quad Y_i \sim \text{Normal}(b, \sigma^2), \quad Z_i \sim \text{Normal}(c, \sigma^2)$$

Let H_0 be that the three distributions are identical i.e. that $a = b = c$, or equivalently

$$X_i \sim Y_i \sim Z_i \sim \text{Normal}(\mu, \sigma^2).$$

Consider the test statistic

$$t = (\hat{a} - \hat{\mu})^2 + (\hat{b} - \hat{\mu})^2 + (\hat{c} - \hat{\mu})^2$$

where hats denote maximum likelihood estimators. If H_0 were true, we'd expect $t(x)$ to be small.

Find the value of the test statistic for the data given. What is the probability of seeing a value this large or larger, if H_0 is true?

fitting a Gaussian: exercise 1.8 page 15)

First, let's work out how to resample the dataset. The general rule is "use a resampling scheme that fits with the null hypothesis." Here, the null hypothesis says that all three datasets are drawn from $\text{Normal}(\mu, \sigma^2)$*, so we'll fit* μ *and* σ *to the entire collection of observations, and resample the three datasets from those common values. Note that resampling must create a full dataset mirroring all our data, i.e. it should create* (X^*, Y^*, Z^*)*.*

```
x = [7.2, 7.3, 7.8, 8.2, 8.8, 9.5]
y = [8.3, 8.5, 9.2]
z = [7.4, 8.5, 9.0]

data = numpy.concatenate([x,y,z])
μ̂ = numpy.mean(data)
σ̂ = numpy.sqrt(numpy.mean(data-μ̂)**2))

def rxyz_star():
    return (numpy.random.normal(size=len(x), loc=μ̂, scale=σ̂),
            numpy.random.normal(size=len(y), loc=μ̂, scale=σ̂),
            numpy.random.normal(size=len(z), loc=μ̂, scale=σ̂))
```

Next the test statistic. We've seen the normal distribution enough times by now that we can just write down the maximum likelihood estimators:

Python unpacking syntax: if z is a list or tuple and f is a function, then f(*z) denotes f(z[0],z[1],...).

```
def t(x,y,z):
    μ' = numpy.mean(numpy.concatenate([x,y,z]))
    a',b',c' = [numpy.mean(v) for v in [x,y,z]]
    return (a'-μ')**2 + (b'-μ')**2 + (c'-μ')**2

# The spread of values we'd expect to see if H0 were true
t_sample = numpy.array([t(*rxyz_star()) for _ in range(10000)])

# The probability of seeing t(X*,Y*,Z*) ≥ t(x,y,z), if H0 were true
t(x,y,z)                                  # value: t = 0.159
p = numpy.mean(t_sample >= t(x,y,z))      # answer: p = 0.592
```

∎

Exercise 7.12 (Racial bias in police stop-and-search).

For the police stop-and-search data, page 35, we wish to know if there's evidence of any racial bias in Cambridgeshire policing. The numbers of stops that led to something being found, versus nothing being found, are

	Asian	Black	Other	White
find	116	170	28	1060
nothing	76	100	9	680

Let n_k be the total number of stops for people of ethnicity k, and let x_k be the number where something was found. Consider the general model $X_k \sim \text{Binom}(n_k, \beta_k)$, for which the maxi-

mum likelihood estimator is simply $\hat{\beta}_k = x_k/n_k$. Define the overall bias score to be

$$d = \max_{k,k'} |\hat{\beta}_k - \hat{\beta}_{k'}|.$$

Larger values correspond to more bias.

Compute d for the data given. Let the null hypothesis be that there is no bias, i.e. that the β_k are all equal. Under this hypothesis, what's the probability of seeing a value greater than or equal to the actual d for the data given?

```
# It's a big file, so retrieve it and store locally for future use
import os.path
if os.path.exists('stop-and-search.csv'):
    print("file already downloaded")
else:
    !wget "https://teachingfiles.blob.core.windows.net/datasets/stop-and-search.csv"
police = pandas.read_csv('stop-and-search.csv')

# Preprocess the data
df = police.loc[~pandas.isnull(police['officer_defined_ethnicity']) &
                ~pandas.isnull(police['outcome']) &
                (police['force']=='cambridgeshire')].copy()
df['eth'] = df['officer_defined_ethnicity']
df['y'] = numpy.where(df['outcome'] != 'False', 'find', 'nothing')

# Quick tabulation, to check what data we're working with
tab = df.groupby(['y','eth']).apply(len).unstack()
```

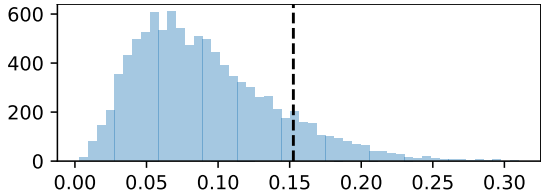
The general rule is "use a resampling scheme that fits with the null hypothesis". Here, the null hypothesis is that all groups share a common parameter, $X_k \sim \text{Binom}(n_k, \theta)$, and it's easy to check that the maximum likelihood estimator for θ is $\hat{\theta} = (\sum_k x_k)/(\sum_k n_k)$.

```
x = tab.loc["find"]
n = tab.loc["find"] + tab.loc["nothing"]
θ̂ = sum(x) / sum(n)
def rx_star(): return [numpy.random.binomial(nk, θ̂) for nk in n]

# The test statistic
def d(x):
    β = numpy.array(x) / n # a vector of length 4
    res = 0
    for e1 in range(len(n)):
        for e2 in range(len(n)):
            res = max(res, abs(beta[e1] - beta[e2]))
    return res
d_actual = d(x)
d_samples = numpy.array([d(rx_star()) for _ in range(10000)])

# What we'd expect to see under H0
plt.hist(d_samples, bins=50, alpha=.4)
plt.axvline(d_actual, color='black', linestyle='dashed')
p = numpy.mean(d_samples > d_actual)  # returns 0.1177
```

The probability of seeing a bias score this high or larger, if H_0 were true, is 11.8%. *This data doesn't provide compelling evidence*[25]

[25] A conventional threshold is $p = 0.05$. Fisher suggested "If p is between .1 and .9 there is certainly no reason to suspect the hypothesis tested. If it is below .02 it is strongly indicated that the hypothesis fails to account for the whole of the facts. We shall not often be astray if we draw a conventional line at .05 and consider that [lower values of p] indicate a real discrepancy." R. A. Fisher. *Statistical methods for research workers*. 1925. URL: http://www.haghish.com/resources/materials/Statistical_Methods_for_Research_Workers.pdf, page 82

When this course was taught in 2018, the police data at the time reported a value of $p = 0.969$ (using data up to 2018-11-01 and excluding ethnicity level "Other" on grounds of insufficient data). Fisher also pointed out that very large values for p are



∎

Example 7.13 (Sign test).
I have a dataset consisting of 500 records, each record a pair (x_i, y_i) where x_i is the predictor variable and y_i is the true label. For example, x_i might be the text of a movie review, and $y_i \in \{☺, 😐, ☹\}$ might be the sentiment.

I have built two classification algorithms, A and B, and I wish to know if one is better than the other. I run them both on each record to get predicted classes $A(x_i)$ and $B(x_i)$. I compare these answers to the true label y_i, and I obtain a grade

$$w_i = \begin{cases} \text{"A" if } A \text{ is correct, } B \text{ not} \\ \text{"B" if } B \text{ is correct, } A \text{ not} \\ \text{"eq" if either both are correct, or both incorrect.} \end{cases}$$

I count up the total number for each grade, to get $n = (n_A, n_B, n_{\text{eq}}) = (70, 50, 380)$.

Consider the general probability model for the grades

$$\mathbb{P}(W = \text{"A"}) = p_A, \quad \mathbb{P}(W = \text{"B"}) = p_B, \quad \mathbb{P}(W = \text{"eq"}) = p_{\text{eq}}.$$

where we require $p_A + p_B + p_{\text{eq}} = 1$. Let the null hypothesis H_0 be "the two algorithms are equally as good", i.e. $p_A = p_B$, and let the alternative hypothesis be the general model, i.e. "the two algorithms might or might not be as good".

Using the test statistic $t = n_A + n_{\text{eq}}/2$, decide whether to use a one-sided or two-sided test, and find the p-value.

Let's implement a general purpose resampler, which can be used for the p parameters under H_0, or for any other p parameters.

```
# The data
n = (nA,nB,neq) = (70,50,380)

# Resample based on an arbitrary p, and compute test statistic
def rn_star(p): return numpy.random.multinomial(n=sum(n), pvals=p)
def t(nA,nB,neq): return nA + neq/2
```

The maximum likelihood estimators under H_0 can be found using the same method as in example 1.3 page 3. The result is

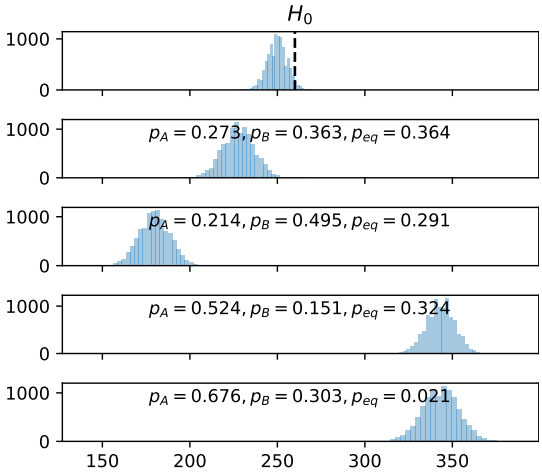
$$\hat{p}_A = \hat{p}_B = \frac{(n_A + n_B)/2}{n_A + n_B + n_{eq}}, \qquad \hat{p}_{eq} = \frac{n_{eq}}{n_A + n_B + n_{eq}}.$$

Python unpacking syntax: t(*rn_star(p)) means: call rn_star(p), get back a tuple (nA,nB,neq), and 'unpack' the call, i.e. call t(nA,nB,neq).

```
# Resampled values of the test statistic, under H_0
p_0 = [(nA+nB)/2/sum(n), (nA+nB)/2/sum(n), neq/sum(n)]
t_samples = numpy.array([t(*rn_star(p_0)) for _ in range(10000)])

fig,axs = plt.subplots(5,1, sharex=True, figsize=(4.5,4))

# Plot the histogram of resampled t values under H_0
axs[0].hist(t_samples, bins=30, alpha=.4)
axs[0].axvline(t(*n), linestyle='dashed', color='black')
axs[0].title('$H_0$')
```

grounds for suspicion; $p = 0.969$ indicates that the $\hat{\beta}_k$ coefficients are much closer together than one would expect by chance. An indication of the police numbers being fudged?

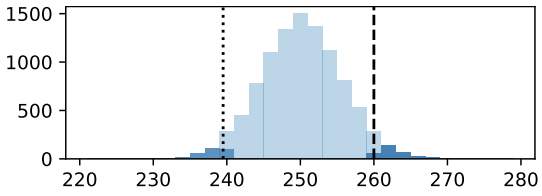
```
17
18 # Plot more alternative histograms for randomly chosen H_1 parameters
19 for ax in axs[1:]:
20     p = numpy.random.uniform(size=3)
21     p = p / sum(p)
22     ax.hist([t(*rn_star(p)) for _ in range(10000)], bins=30, alpha=.4)
```



If an alternative hypothesis is true, the distribution of t might be shifted to the right, or it might be shifted to the left. Either is possible, depending on the alternative. Thus an extreme value of $t(x)$, lying in either tail of the distribution of $t(X^*)$ under H_0, is evidence for rejecting H_0 in favour of the alternative.

Below is another plot, showing the histogram under H_0. It marks out all samples of $t(X^*)$ that are larger than the observed value $t(x)$, as well as an equal number of samples on the left hand side. In other words, a two-sided region. The p-value, i.e. the total probability of lying in this two-sided region, is $2\,\mathbb{P}(t(X^*) > t(x))$ which in this problem turns out to be $\approx 5.8\%$. According to Fisher, a p-value in the range 5%–10% is grounds for suspicion that the two classifiers might be different, but not clear evidence. Best gather more data.

```
23 p = 2 * numpy.mean(t_samples > t(*n))  # 0.0582
```



■