Probabilistic Programming

Marius Popescu

popescunmarius@gmail.com

2019 - 2020

Bayesian Hypothesis Testing

Hypothesis Testing

One of the most frequently encountered scientific procedures is a comparison of two groups. Given data from two groups, researchers ask various comparative questions: How much is one group different from another? Can we be reasonably sure that the difference is non-zero? How certain are we about the magnitude of difference? These questions are difficult to answer because data are contaminated by random variability.

Hypothesis Testing

Part of a data scientist’s goal is to be a champion of experiments, and one of the best tools for a data scientist is a well-designed split-test experiment.

Hypothesis testing is a statistical design pattern for determining the difference of effectiveness between two different treatments:

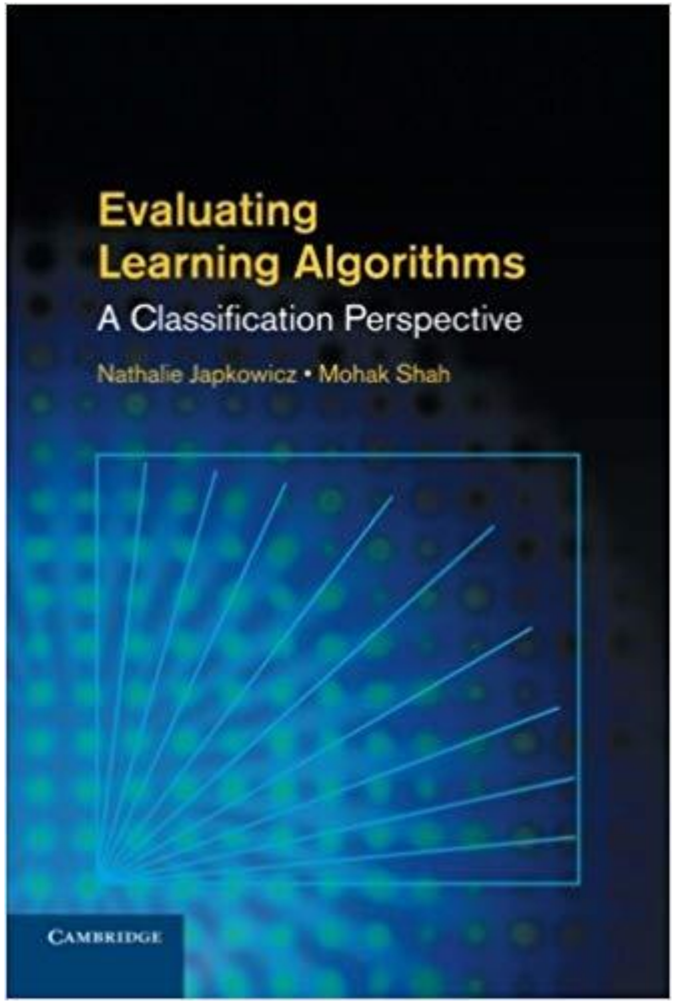
- A pharmaceutical company is interested in the effectiveness of drug A vs drug B. The company will test drug A on some fraction of their trials, and drug B on the other fraction. After performing enough trials, the in-house statisticians sift through the data to determine which drug yielded better results.
- Front-end web developers are interested in which design of their website yields more sales or some other metric of interest. They will route some fraction of visitors to site A, and the other fraction to site B, and record if the visit yielded a sale or not. The data is recorded (in real-time), and analyzed afterwards.

Hypothesis Testing in Machine Learning

Comparing the performance of different machine learning methods on a given problem is another matter that is not so easy as it sounds: to be sure that apparent differences are not caused by chance effects, statistical tests are needed.

…

How can we be confident about whether the difference in measurement for two or more algorithms denotes a statistically significant difference in their performance? Is this statistical difference practically relevant as well? How can we best use the available data to discover whether such differences exist?


Evaluating Learning Algorithms
A Classification Perspective
Nathalie Japkowicz • Mohak Shah
CAMBRIDGE

Bayesian Hypothesis Testing in Machine Learning

The machine learning community adopted the use of null hypothesis significance testing (NHST) in order to ensure the statistical validity of results. Many scientific fields however realized the shortcomings of frequentist reasoning and in the most radical cases even banned its use in publications. We should do the same: just as we have embraced the Bayesian paradigm in the development of new machine learning methods, so we should also use it in the analysis of our own results.

Benavoli, A., Corani, G., Demšar, J., & Zaffalon, M. (2017). Time for a change: a tutorial for comparing multiple classifiers through Bayesian analysis. The Journal of Machine Learning Research, 18(1), 2653-2688.

Bayesian A/B Testing

Bayesian A/B Testing

The fundamental idea in an A/B test is that we consider a perfect counterfactual universe, where the population under study is identical but subject to some treatment, then any differences between the populations after the study must be attributed to the treatment. In practice, we can't spin up other universes, so we rely on using large enough samples in two groups that approximate a counterfactual.

Conversion Testing

We have two Web site designs, called A and B. When a user lands on our Web site, we randomly show them design A or B, and record this assignment. After enough visitors have done this, we join this dataset against some metric of interest (typically, for Web sites, we are interested in a purchase or signup, call it *conversion*). For example, consider the following numbers:

```
visitors_to_A = 1300
visitors_to_B = 1275
```

```
conversions_from_A = 120
conversions_from_B = 125
```

What we are really interested in is the probability of conversion, given site A or B. As a business, we want this probability to be as high possible. So our goal is to determine which site, A or B, has a high probability of conversion.

A First Bayesian Model

We are interested in using what we know, N_A, N_B (the total trials administered) and n_A, n_B (the number of conversions), to estimate what p_A and p_B, the true probabilities of conversion, might are.

We need to assign prior distributions to our unknown quantities. *A priori*, what do we think p_A might be? For this example, we have no strong conviction about p_A, so for now, let's assume p_A is uniform over [0,1]. A conversion from A will be modeled as a Bernoulli with probability p_A.

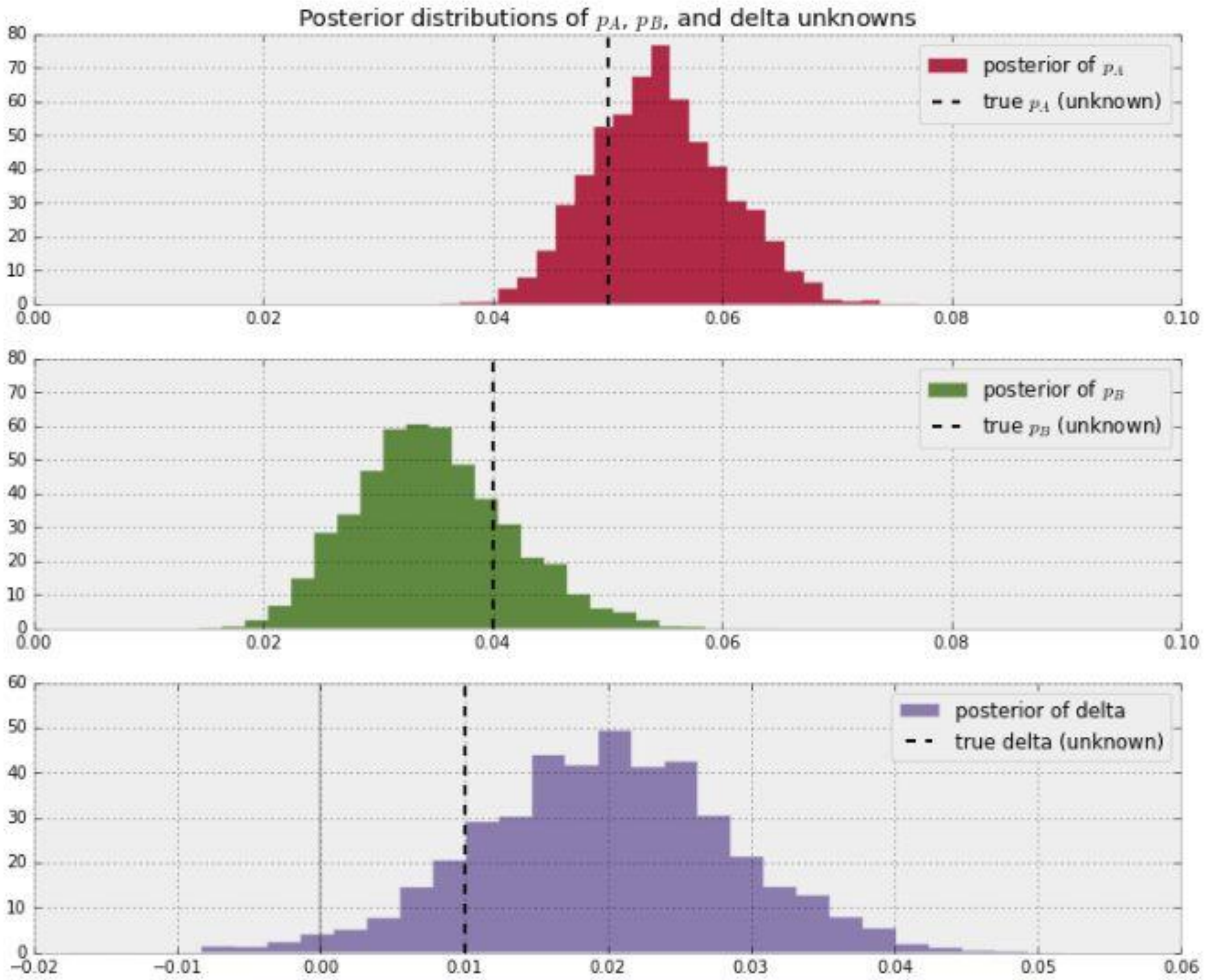
A similar analysis can be done for site B's response data.

But what we are really interested in is the difference between p_A and p_B. We will infer p_A, p_B and $\delta = p_A - p_B$ all at once.

A First Bayesian Model

```
true_p_A = 0.05
true_p_B = 0.04

N_A = 1500
N_B = 750

observations_A = pm.rbernoulli(true_p_A, N_A)
observations_B = pm.rbernoulli(true_p_B, N_B)

p_A = pm.Uniform("p_A", 0, 1)
p_B = pm.Uniform("p_B", 0, 1)

@pm.deterministic
def delta(p_A=p_A, p_B=p_B):
    return p_A - p_B

obs_A = pm.Bernoulli("obs_A", p_A, value=observations_A, observed=True)
obs_B = pm.Bernoulli("obs_B", p_B, value=observations_B, observed=True)

mcmc = pm.MCMC([p_A, p_B, delta, obs_A, obs_B])
mcmc.sample(20000, 1000)
```

The Results



```
print("Probability site A is WORSE than site B: %.3f" % \
    (delta_samples < 0).mean())

print("Probability site A is BETTER than site B: %.3f" % \
    (delta_samples > 0).mean())
```

```
[Output]:

Probability site A is WORSE than site B: 0.017
Probability site A is BETTER than site B: 0.983
```

- Notice that as a result of $N_B < N_A$, i.e. we have less data from site B, our posterior distribution of p_B is fatter, implying we are less certain about the true value of p_B than we are of p_A.
- With respect to the posterior distribution of δ, we can see that the majority of the distribution is above $\delta = 0$, implying there site A's response is likely better than site B's response.

Conversion Testing

We have two Web site designs, called A and B. When a user lands on our Web site, we randomly show them design A or B, and record this assignment. After enough visitors have done this, we join this dataset against some metric of interest (typically, for Web sites, we are interested in a purchase or signup, call it *conversion*). For example, consider the following numbers:

```
visitors_to_A = 1300
visitors_to_B = 1275
```

```
conversions_from_A = 120
conversions_from_B = 125
```

What we are really interested in is the probability of conversion, given site A or B. As a business, we want this probability to be as high possible. So our goal is to determine which site, A or B, has a high probability of conversion.

A Second Bayesian Model

- We'll model the probability of conversion given site A, or site B.
- Since we are modeling a probability, a good choice for a prior distribution is the Beta distribution. (Why? It is restricted to values between 0 and 1, identical to the range that probabilities can take on.)
- Our number of visitors and conversion data are binomial: for site A, out of 1,300 trials, we had 120 successes.
- Recall that a Beta prior and binomial observations have a conjugate relationship; this means we don't need to perform any MCMC!

A Second Bayesian Model

If our prior is $\mathrm{Beta}(\alpha_0, \beta_0)$, and we observe N trials and X successes, then our posterior is $\mathrm{Beta}(\alpha_0 + X, \beta_0 + N)$. Using the built-in `beta` function from SciPy, we can directly sample from the posterior. Our prior is a $\mathrm{Beta}(1{,}1)$; recall that this is identical to a uniform distribution on [0, 1]:

```
from scipy.stats import beta

alpha_prior = 1
beta_prior = 1

posterior_A = beta(alpha_prior + conversions_from_A,
                   beta_prior + visitors_to_A - conversions_from_A)
posterior_B = beta(alpha_prior + conversions_from_B,
                   beta_prior + visitors_to_B - conversions_from_B)
```

A Second Bayesian Model

Next, we'd like to determine which group has a large probability of conversion. To do this, similar to MCMC, we use samples from the posterior and compare the probability that samples from the posterior of A are larger than samples from the posterior of B. We use the `rvs` method to generate samples:

```
samples = 20000 # We want this to be large to get a better approximation.

samples_posterior_A = posterior_A.rvs(samples)
samples_posterior_B = posterior_B.rvs(samples)

print (samples_posterior_A > samples_posterior_B).mean()

[Output]:
0.31355
```

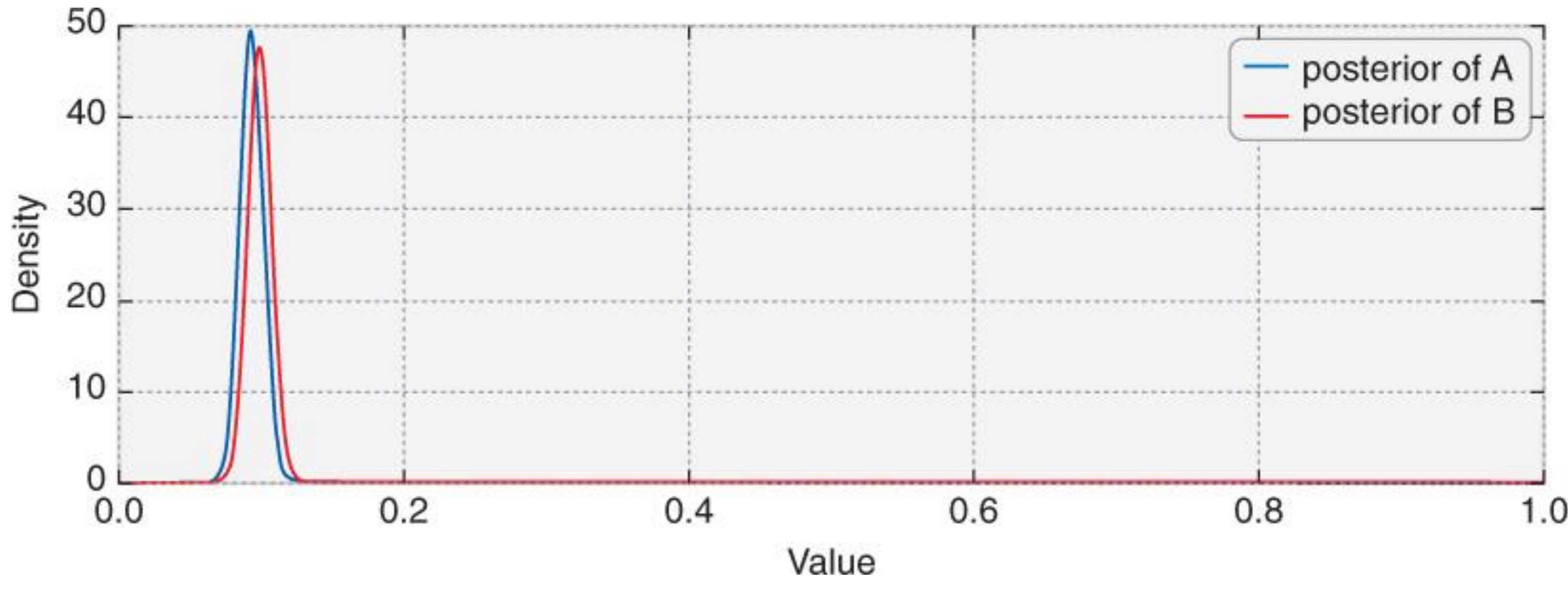
The Results

So we can see here that there is a 31% chance that site A converts better than site B. (Conversely, there is a 69% chance that site B converts better than A.) This is not very significant; consider that if we reran this experiment with identical pages, it would return a probability close to 50%.

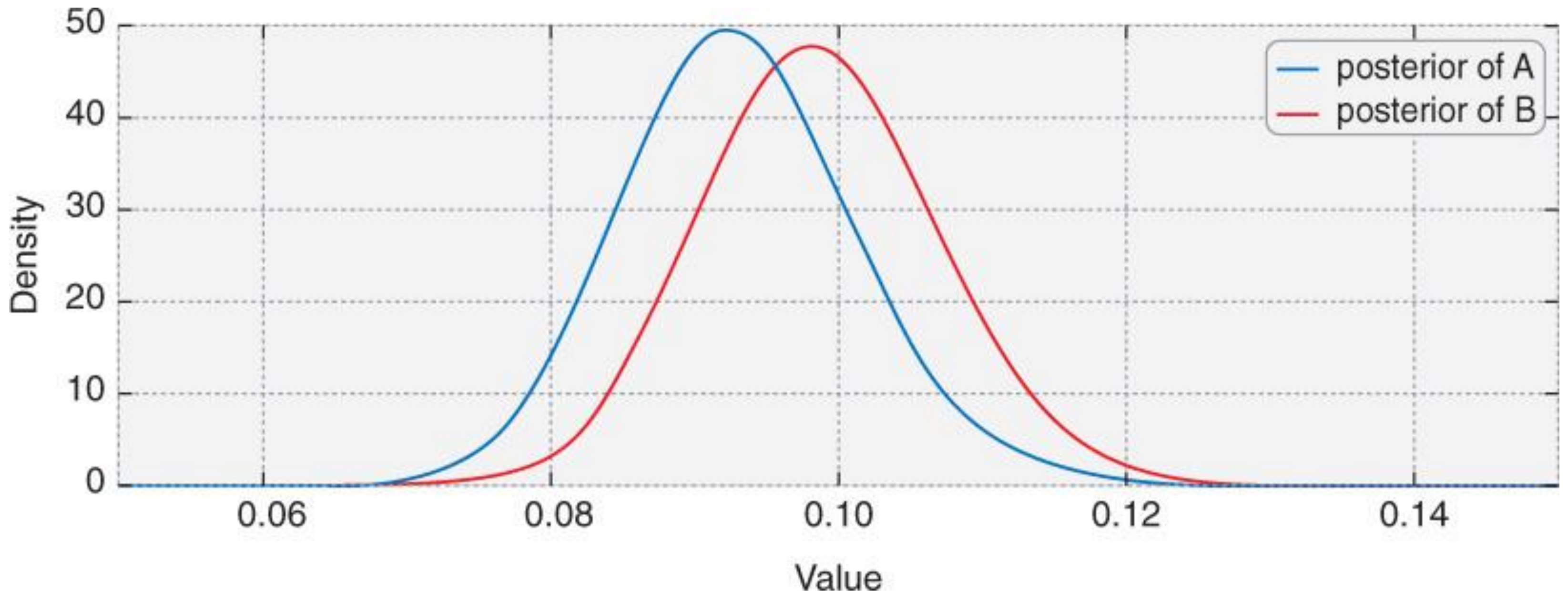
The conversion posteriors of sites A and B

We can visualize the posterior, too, without using histograms. This is done by using the `pdf` method:

```
x = np.linspace(0,1, 500)
plt.plot(x, posterior_A.pdf(x),
         label='posterior of A’)
plt.plot(x, posterior_B.pdf(x),
         label='posterior of B')
```



Zooming in closer to the area of interest



Extending A/B Testing

Adding a Linear Loss Function

Adding a Linear Loss Function

Our previous A/B test analysis was only concerned with whether the user signed up or not; now, we want to know what the expected revenue to be earned is.

A common goal of Internet companies is not only to gain signups, but also to optimize which signup plan the user might select. For example, a business might want new customers to choose a higher-priced plan if presented with two or more plan options.

Suppose users are shown two different versions of this pricing page, and we'd like to determine the *expected revenue* we receive per impression.

Expected Revenue Analysis

For the moment, ignore the A/B test and consider the analysis for a single Web page style. In a completely transparent world, where we knew everything, we could calculate this expected value for this fictional company:

$$E[R] = 79p_{79} + 49p_{49} + 25p_{25} + 0p_0$$

where p_{79} is the probability of selecting the $79 pricing plan, and so on. Also was included a fictional $0 pricing plan for someone who doesn't pick a plan. This is added so that the probabilities sum to 1:

$$p_{79} + p_{49} + p_{25} + p_0 = 1$$

Expected Revenue Analysis

The next step is to estimate these probabilities. We can't use a Beta/binomial model for each probability, as the probabilities are correlated; they must sum to 1. For example, if p79 is high, then the other probabilities must be low. We need to model all the probabilities together.

For our signup page, our observables follow a multinomial distribution, where we do not know the values of the probability vector P.

The Dirichlet distribution returns a vector of positive values that sum to 1.

Luckily, we have a relationship between the Dirichlet and multinomial distributions similar to that between the Beta and the binomial distributions. The Dirichlet distribution is a conjugate prior to the multinomial distribution! This means we have exact formulas for the posteriors of the unknown probabilities.l

If our prior is $\text{Dirichlet}(1, 1, \ldots, 1)$ and our observables are $N_1, N_2, \ldots, N_m$ then our posterior is:

$$\text{Dirichlet}(1 + N_1, 1 + N_2, \ldots, 1 + N_m)$$

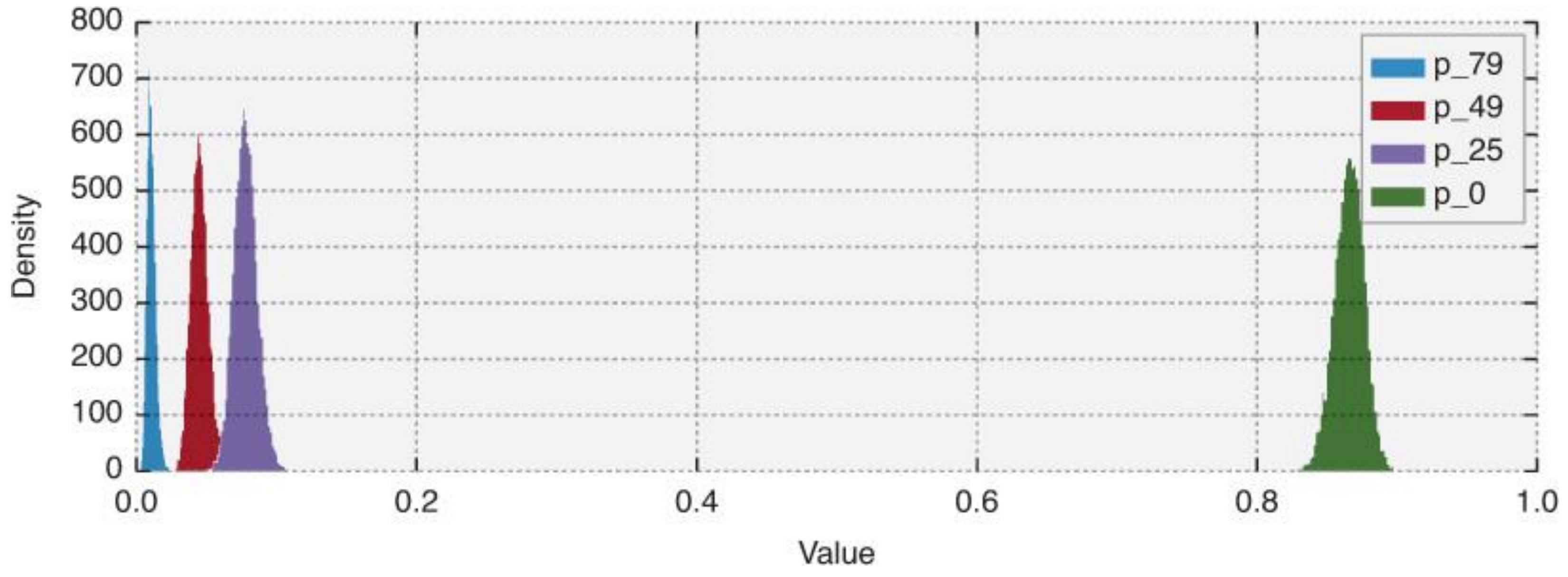
Expected Revenue Analysis

Suppose 1,000 people view the page, and we have the following signups:

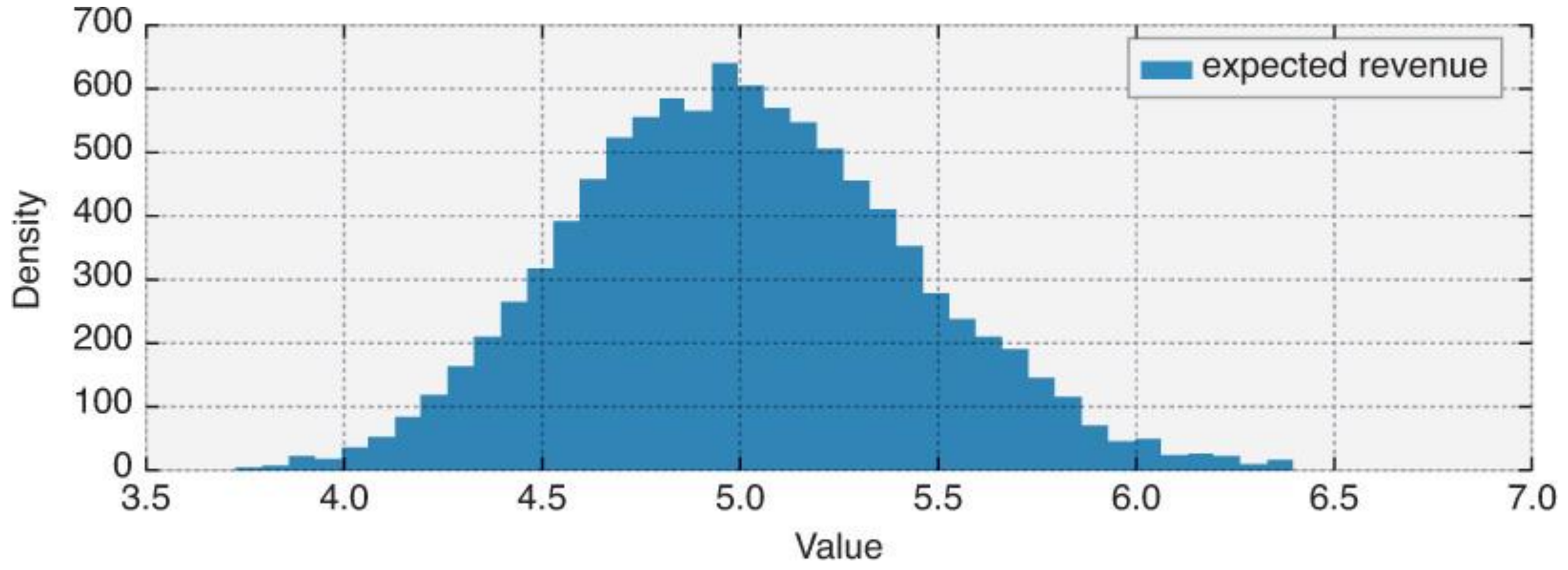
```
N    = 1000; N_79 = 10; N_49 = 46; N_25 = 80
N_0  = N - (N_79 + N_49 + N_49)

observations = np.array([N_79, N_49, N_25, N_0])

prior_parameters = np.array([1,1,1,1])
posterior_samples = np.random.dirichlet(prior_parameters + observations,
                                        size=10000)
print "Two random samples from the posterior:"
print posterior_samples[0]
print posterior_samples[1]
```

```
[Output]:
Two random samples from the posterior:
[ 0.0165  0.0497  0.0638  0.8701]
[ 0.0123  0.0404  0.0694  0.878 ]
```

The Probability Density Function of this Posterior



Posterior Distributions of the Expected Revenue



We can see that the expected revenue is likely between $4 and $6, and unlikely to be outside this range.

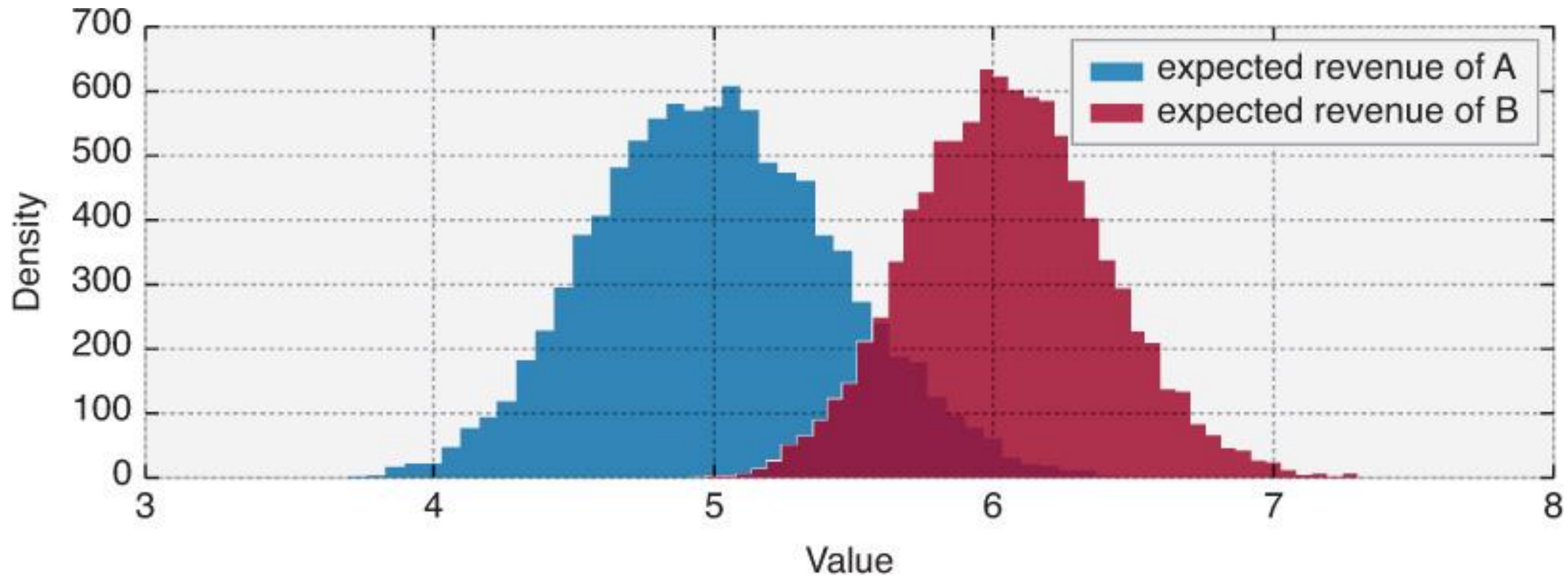
Extending to an A/B Experiment: Some Artificial Data

```
N_A = 1000
N_A_79 = 10
N_A_49 = 46
N_A_25 = 80
N_A_0 = N_A - (N_A_79 + N_A_49 + N_A_49)
observations_A = np.array([N_A_79, N_A_49, N_A_25, N_A_0])

N_B = 2000
N_B_79 = 45
N_B_49 = 84
N_B_25 = 200
N_B_0 = N_B - (N_B_79 + N_B_49 + N_B_49)
observations_B = np.array([N_B_79, N_B_49, N_B_25, N_B_0])
```

Extending to an A/B Experiment: The Model

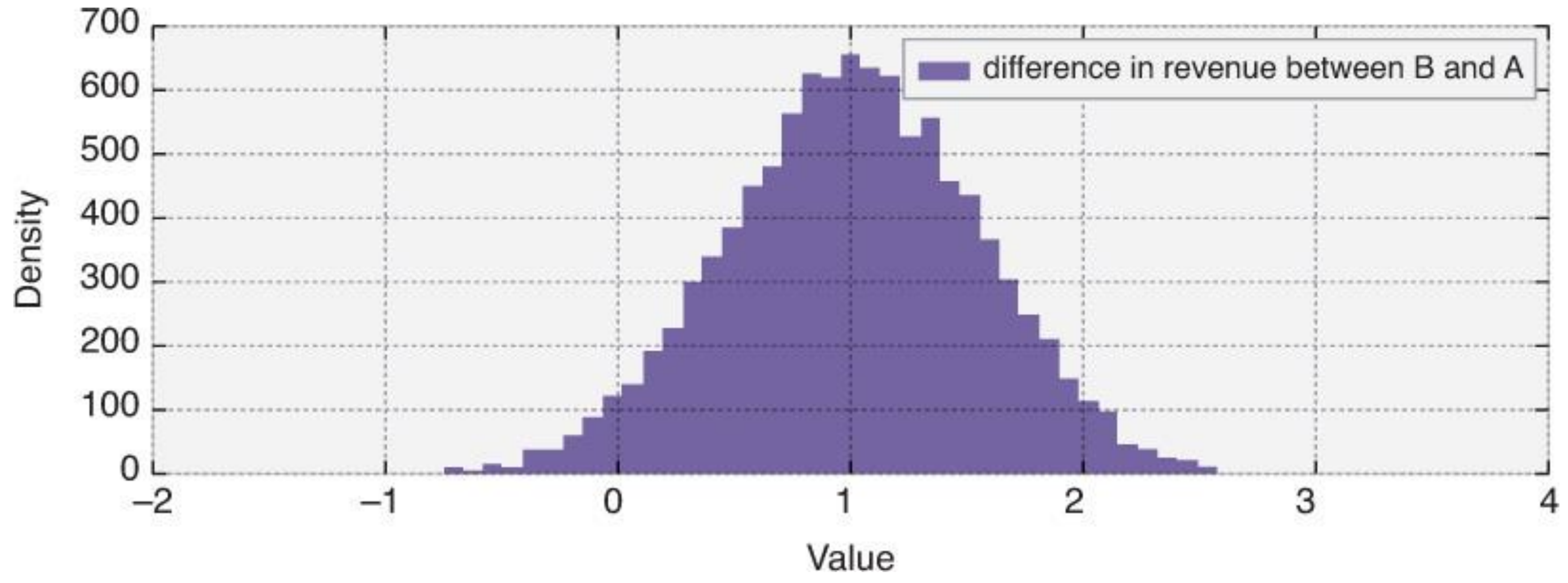
```
prior_parameters = np.array([1,1,1,1])

posterior_samples_A = np.random.dirichlet(prior_parameters + observations_A, size = 10000)
posterior_samples_B = np.random.dirichlet(prior_parameters + observations_B, size = 10000)

def expected_revenue(P):
    return 79*P[:,0] + 49*P[:,1] + 25*P[:,2] + 0*P[:,3]

posterior_expected_revenue_A = expected_revenue(posterior_samples_A)
posterior_expected_revenue_B = expected_revenue(posterior_samples_B)
```

Extending to an A/B Experiment: The Results



```
p = (posterior_expected_revenue_B > posterior_expected_revenue_A).mean()
print("Probability that page B has a higher revenue than page A: %.3f" % p)

[Output]:
Probability that page B has a higher revenue than page A: 0.965
```

Posterior distribution of the delta between expected revenues of pages A and B



Looking at this posterior, we see that there is about a 50% chance that the difference is more than $1, and possibly even larger than $2. And if we are wrong about our choice of B (which is possible), we probably won’t lose much: The distribution doesn’t extend much past −$0.50.

Bayesian Estimation Supersedes the T-Test

BEST

Student’s T-Test

Probably the most taught statistical test in classrooms is the *t-test*. The traditional t-test is a frequentist test to determine whether there is a significant difference between the means of two groups

More generally, the t-test is any statistical hypothesis test in which the test statistic follows a Student's t-distribution under the null hypothesis.

Among the most frequently used t-tests are:

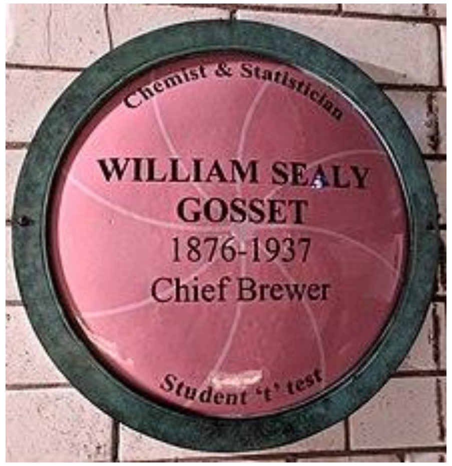
- A one-sample location test of whether the mean of a population has a value specified in a null hypothesis.
- A two-sample location test of the null hypothesis such that the means of two populations are equal.

Some History

The t-statistic was introduced in 1908 by William Sealy Gosset, a chemist working for the Guinness brewery in Dublin, Ireland. "Student" was his pen name.

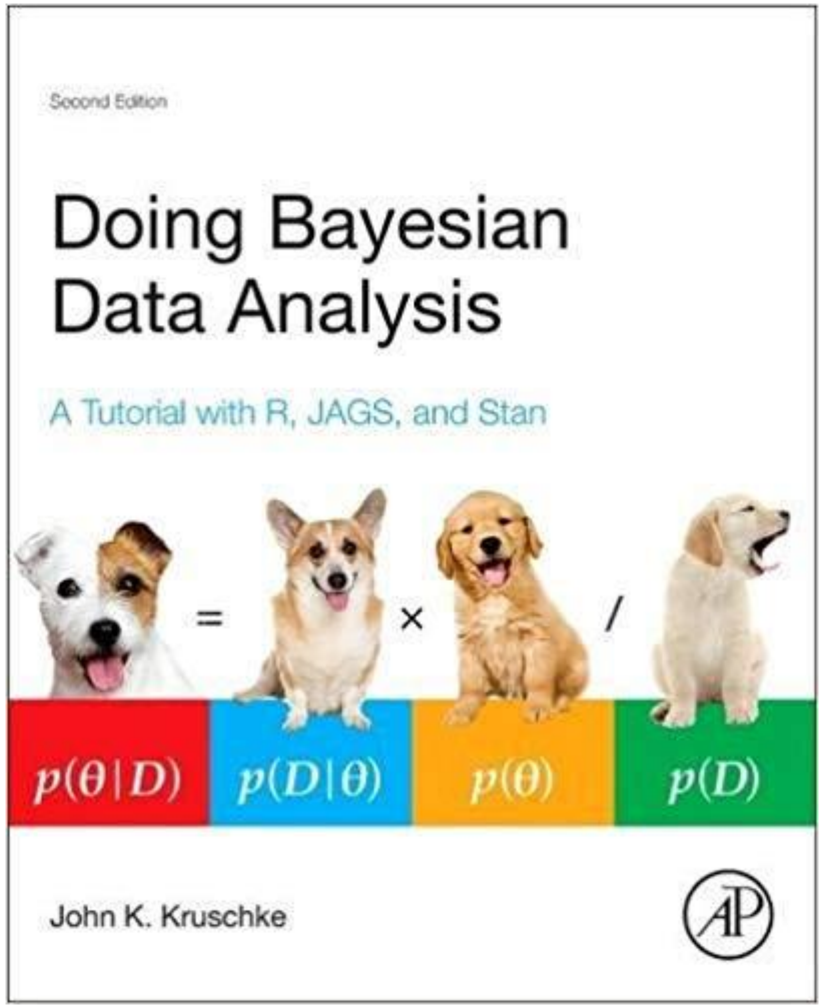
Gosset had been hired owing to Claude Guinness's policy of recruiting the best graduates from Oxford and Cambridge to apply biochemistry and statistics to Guinness's industrial processes. Gosset devised the t-test as an economical way to monitor the quality of stout. The t-test work was submitted to and accepted in the journal Biometrika and published in 1908. Company policy at Guinness forbade its chemists from publishing their findings, so Gosset published his statistical work under the pseudonym "Student“.

Guinness had a policy of allowing technical staff leave for study (so-called "study leave"), which Gosset used during the first two terms of the 1906–1907 academic year in Professor Karl Pearson's Biometric Laboratory at University College London. Gosset's identity was then known to fellow statisticians and to editor-in-chief Karl Pearson.



Bayesian Estimation Supersedes the T-Test

Kruschke, J. K. (2013). Bayesian estimation supersedes the t test. Journal of Experimental Psychology: General, 142(2), 573.



Second Edition

Doing Bayesian Data Analysis

A Tutorial with R, JAGS, and Stan

$p(\theta|D) = p(D|\theta) \times p(\theta) / p(D)$

John K. Kruschke

BEST

“

Bayesian estimation for two groups provides complete distributions of credible values for the effect size, group means and their difference, standard deviations and their difference, and the normality of the data. The method handles outliers. The decision rule can accept the null value (unlike traditional t tests) when certainty in the estimate is high (unlike Bayesian model comparison using Bayes factors).

“

BEST

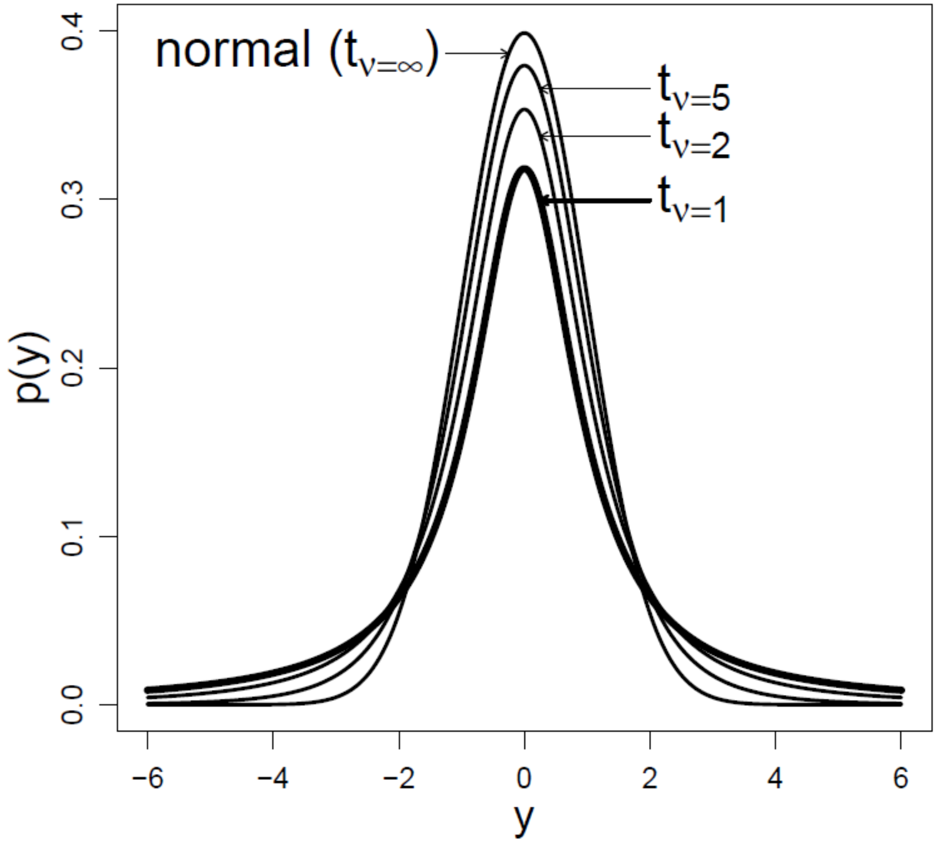
In our model of the data, we will describe each group's data with a t distribution, with each group having its own mean parameter and standard deviation parameter. Because outliers are usually relatively few in number, we will use the same ν parameter for both groups so that both groups' data can inform the estimate of ν. Thus, our description of the data uses five parameters: The means of the two groups (μ_1 and μ_2), the standard deviations of the two groups (σ_1 and σ_2), and the normality of the data within the groups (ν). We will use Bayesian inference to estimate the five parameters.

Student's T-Distribution

$$Z \sim \mathrm{t}(\mu, \lambda, \nu)$$

$$f_Z(z|\mu,\lambda,\nu) = \frac{\Gamma(\frac{\nu+1}{2})}{\Gamma(\frac{\nu}{2})}\left(\frac{\lambda}{\pi\nu}\right)^{\frac{1}{2}}\left[1+\frac{\lambda(z-\mu)^2}{\nu}\right]^{-\frac{\nu+1}{2}}, \qquad \nu > 0$$

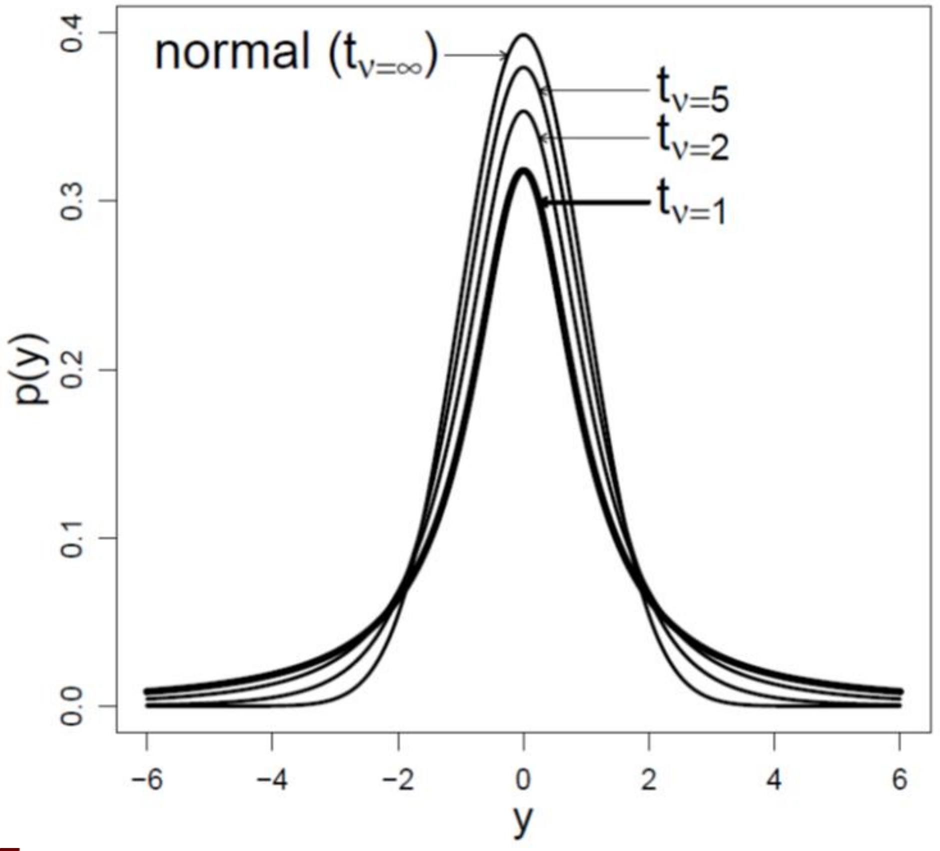
$$E(Z|\mu,\lambda,\nu) = \int_{-\infty}^{\infty} z f_Z(z|\mu,\lambda,\nu)\,dz = \mu$$



```
Z=pymc.NoncentralT("Z", mu, lambda, nu)
```

Student’s T-Distribution

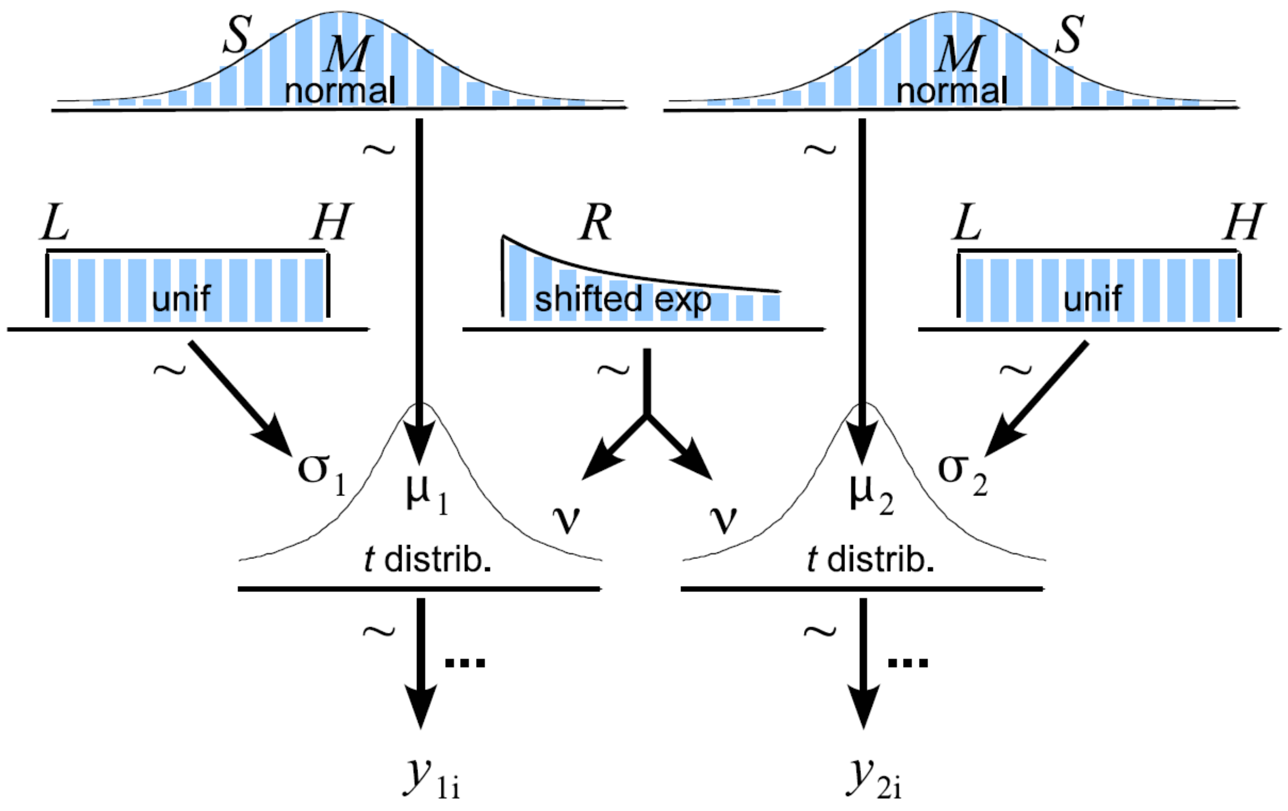
The t distribution is governed by a parameter denoted by the Greek letter ν (nu), which can range continuously from 1 to infinity. When ν is small, the t distribution has heavy tails, and when ν is large (e.g., 100), the t distribution is nearly normal. Therefore I will refer to ν as the *normality* parameter in the t distribution. (Traditionally, in the context of sampling distributions, this parameter is referred to as the *degrees of freedom*. The t distribution can describe data with outliers by setting ν to a small value, but the t distribution can also describe data that are normal, without outliers, by setting ν to a large value. Just like the normal distribution, the t distribution also has a mean parameter μ, and a standard deviation parameter σ.



BEST

According to the BEST model, the priors for the unknowns are as follows:

1. μ_1 and μ_2 come from a Normal distribution with prior mean equal to the pooled mean of data from both groups, and prior standard deviation equal to 1,000 times the pooled standard deviation. (This is a very wide, uninformative prior).
2. σ_1 and σ_2 come from a uniform distribution, restricted to one one-thousandths of the pooled standard deviation, to 1000 times the standard deviation. (Again, a very wide uninformative prior).
3. ν is estimated from a shifted exponential distribution with parameter equal to 29. This prior was selected because it balances nearly-normal distributions $(\nu > 30)$ with heavy-tailed distributions $(\nu < 30)$.



BEST in PyMC: The Setup

Following our A/B testing theme, suppose we have data about the length of time a user is on a test page. This data is not binary; it's continuous. For example, we'll create some artificial data with the following code:

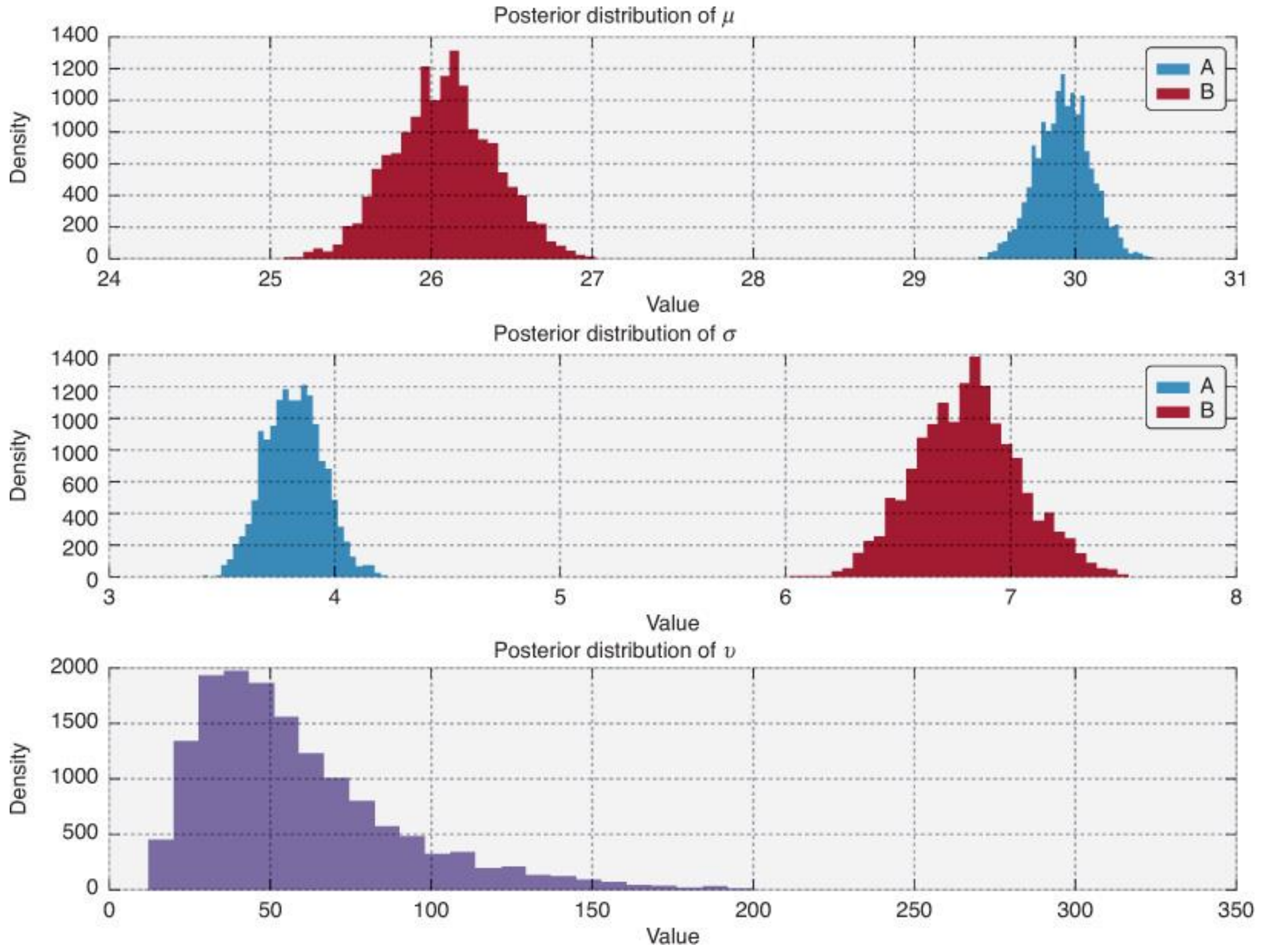
```
N = 250

mu_A, std_A = 30, 4
mu_B, std_B = 26, 7

# create durations (seconds) users are on the pages for

durations_A = np.random.normal(mu_A, std_A, size = N)
durations_B = np.random.normal(mu_B, std_B, size = N)
```

BEST in PyMC: The Model

```
pooled_mean = np.r_[durations_A, durations_B].mean()
pooled_std = np.r_[durations_A, durations_B].std()

tau = 1. / np.sqrt(1000. * pooled_std) # PyMC uses a precision
# parameter, 1/sigma**2
mu_A = pm.Normal("mu_A", pooled_mean, tau)
mu_B = pm.Normal("mu_B", pooled_mean, tau)

std_A = pm.Uniform("std_A", pooled_std / 1000., 1000. * pooled_std)
std_B = pm.Uniform("std_B", pooled_std / 1000., 1000. * pooled_std)

nu_minus_1 = pm.Exponential("nu-1", 1./29)

obs_A = pm.NoncentralT("obs_A", mu_A, 1.0 / std_A ** 2, nu_minus_1 + 1,
                       observed = True, value = durations_A)
obs_B = pm.NoncentralT("obs_B", mu_B, 1.0 / std_B ** 2, nu_minus_1 + 1,
                       observed = True, value = durations_B)

mcmc = pm.MCMC([obs_A, obs_B, mu_A, mu_B, std_A, std_B, nu_minus_1])
mcmc.sample(25000,10000)
```

The Results

We can see that not only does page A have a higher average duration of time spent on the page, but the volatility of each page view is lower (as page A's standard deviation is lower). Furthermore, with these posteriors, we can calculate differences between groups, effect sizes, and so on.


Posterior distribution of μ
Density
Value
A
B
Posterior distribution of σ
Density
Value
A
B
Posterior distribution of υ
Density
Value