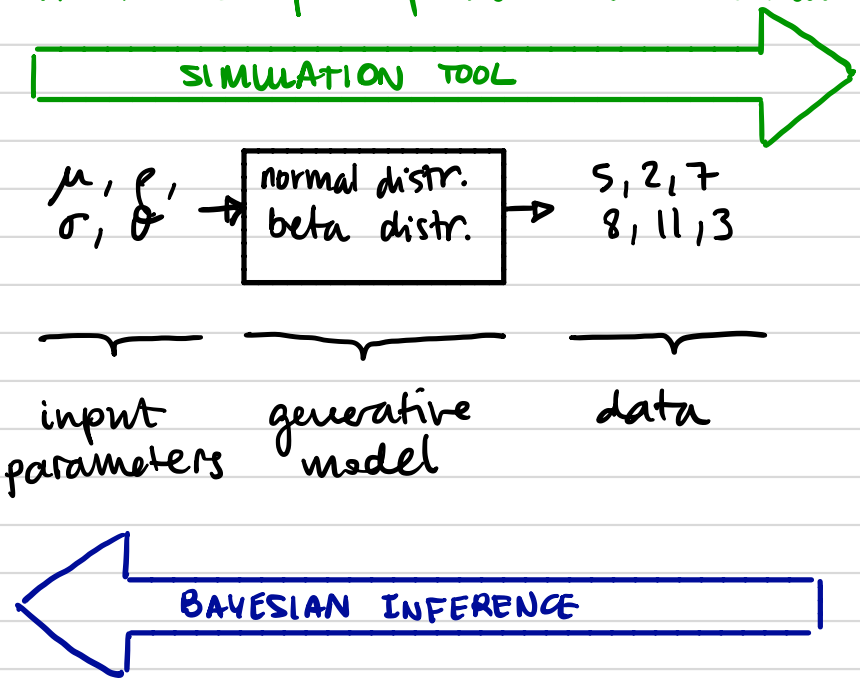
Introduction to Bayesian data analysis

Part 1: What is Bayes data analysis

- It is when you use probability to represent uncertainty
- A method for figuring out unknowns that requires 3 things:
  1) Data
  2) A generative model
  3) Priors. → what information the model has before seeing the data.

2) Generative model - what?

- With input parameters and known distributions, we can simulate data.
- Classical example of monte carlo simulations.



- But we can also think of it as discovering the underlying patterns given certain data.
- What are the parameters and distributions that will most likely generate the given data.

↳ motivating example

EXAMPLE: Bayesian A/B testing for Swedish Fish. LTD.

- S.F. L.T.D sells fish on a subscription model.
- They want to expand into the danish market.
- 2 strategies:

Treatment A:
Brochure

Treatment B:
Brochure + salmon sample.

Analysing Treatment A:

THE DATA

- Tried on 16 persons
- 6 signed up for subscription

} If we scaled this method to a large population, what can we expect?

option: to calculate the current sign up proportion

↳ 6/16 = 38%. → it might be a good guess, but it is also quite uncertain, specially due to the small sample.

↳ Once we have the data, let's move towards a generative model of people signing up to our fish subscription.

↳

THE GENERATIVE MODEL. (simulation).

- To start with, let's assume that there is an underlying true sign up rate → 55%.
- When we simulate this rate for 16 people, we might get that 7/16 people would theoretically sign up.

55% → [try for 16 people] → 7/16.

↳ PROBLEM
- we don't want to simulate data
- we already know what happened when we asked 16 people.
- not only that, it's pretty much impossible to know the true underlying sign-up rate.

¿ θ ? → [try for 16 people] → 6 sign ups

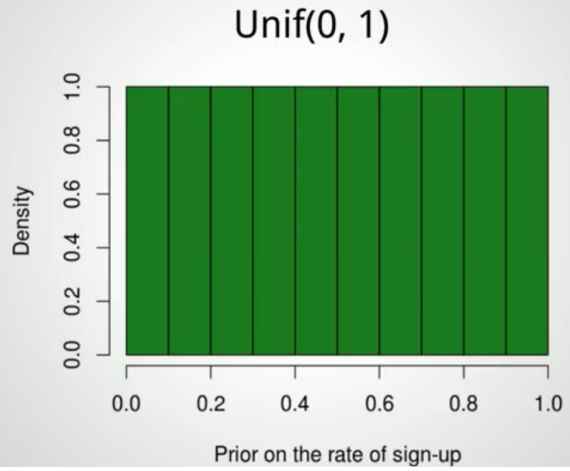
↳ what we really want to know is a reasonable sign up rate

what params ← uncertainty → what distribution. (in this case it could be a Binomial (16, θ).

↳ to understand uncertainty, let's look at the priors

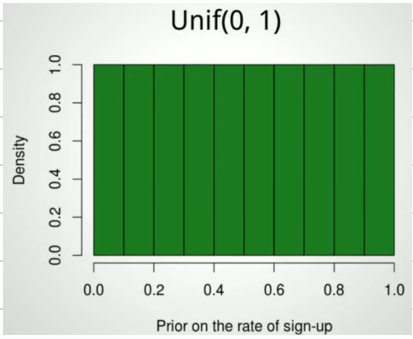
THE PRIORS what information does the model have before seeing the data

- At time 0, it is sensible to assume that the model doesn't know absolutely anything
  - ↳ in other words, the model can assume that any rate of sign up, is equally likely.
    - ↳ can be represented by the uniform distribution



PUTTING IT TOGETHER

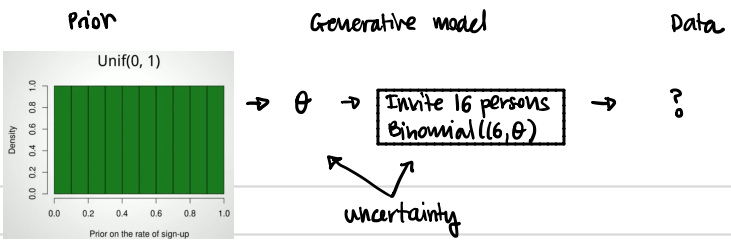
Prior → Generative model → Data



→ θ → Invite 16 persons
Binomial(16, θ) → ?

uncertainty

↳ now that we have all the components we need, let's run the model.



RUNNING THE MODEL

- Round 1 → we draw a parameter θ based on prior distribution → θ = 21%
  - → push through Binomial (16, 0.21) → 4 sign-ups.
- Repeat many times (100k times)

θ = 21% → Invite 16 → 4 ✗
θ = 54% → Invite 16 → 6 ✓
θ = 95% → Invite 16 → 15 ✗
θ = 21% → Invite 16 → 6 ✓

we also know that 6 out 16 actually signed-up.

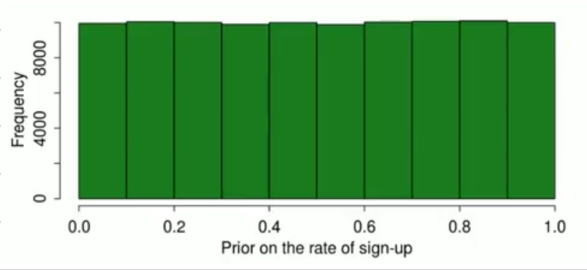
remove parameters that didn't result in this

- Let's look at this by analysing the distributions:

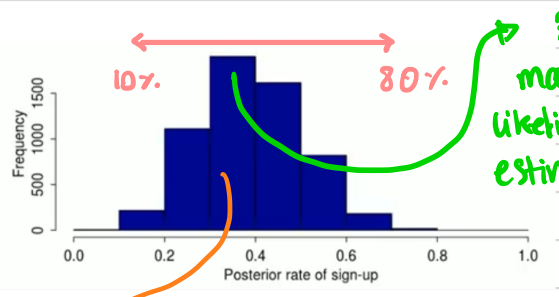
Before removing parameters

After removing parameters.

PRIOR



POSTERIOR



Answers the question of what parameter will likely provide the data we have captured.
- very unlikely that signup rate is <10% or >80%
- most likely being 30% or 40%.

At the same time, uncertainty is quite high!

1900 / 5700 draws in 30-40% draws in total = 33% chance that sign up rate is between 30%-40%.

The Bayes Theorem

- Represents what we have done in a mathematical form which generalises for any parameters and distributions.

$$P(\theta|D) = \frac{P(\theta) \cdot P(D|\theta)}{\Sigma P(\theta) \cdot P(D|\theta)}$$

$\theta \rightarrow$ parameters
$D \rightarrow$ data

→ Problem → this method is using approximate bayesian
  → super slow
    → but there are tricks that we can use to make this faster.

Part 2 : Why use Bayesian analysis

- Recall our example of Swedish Fish LTD, who wanted to expand into the danish market.
- We analysed through bayes how could we describe, with a level of uncertainty, how good would our expansion campaign A do when scaling to mass market.
- The CEO wants to try a new campaign.

- We can use Bayes to help us answer questions like the ones below.

EXAMPLE: Bayesian A/B testing for Swedish Fish. LTD.

- S.F. L.T.D sells fish on a subscription model.
- They want to expand into the danish market.
- 2 strategies:

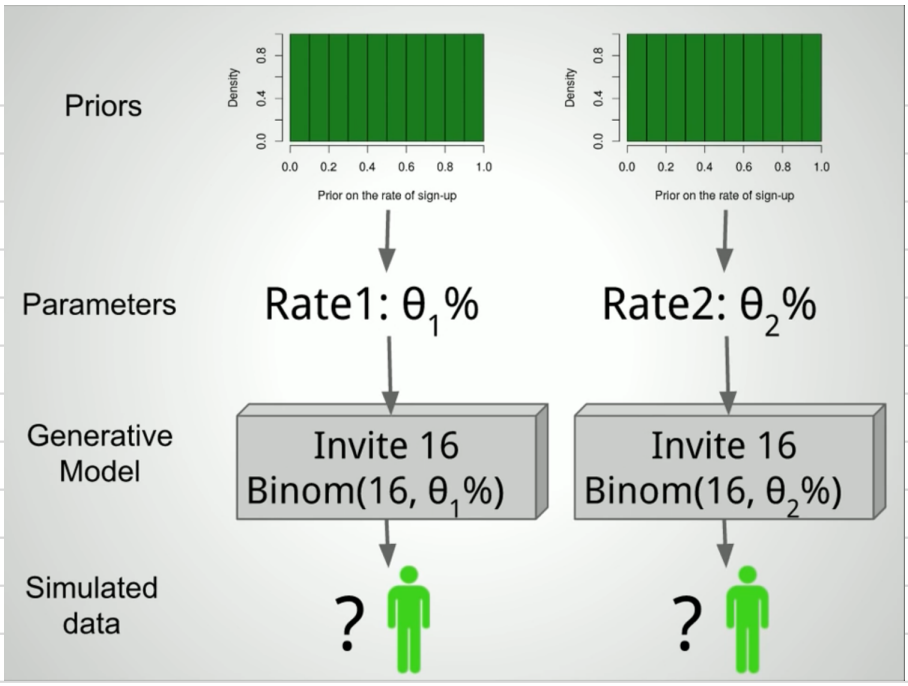
Treatment A :
Brochure

Treatment B :
Brochure + salmon sample

6/16 sign ups

10/16 sign ups

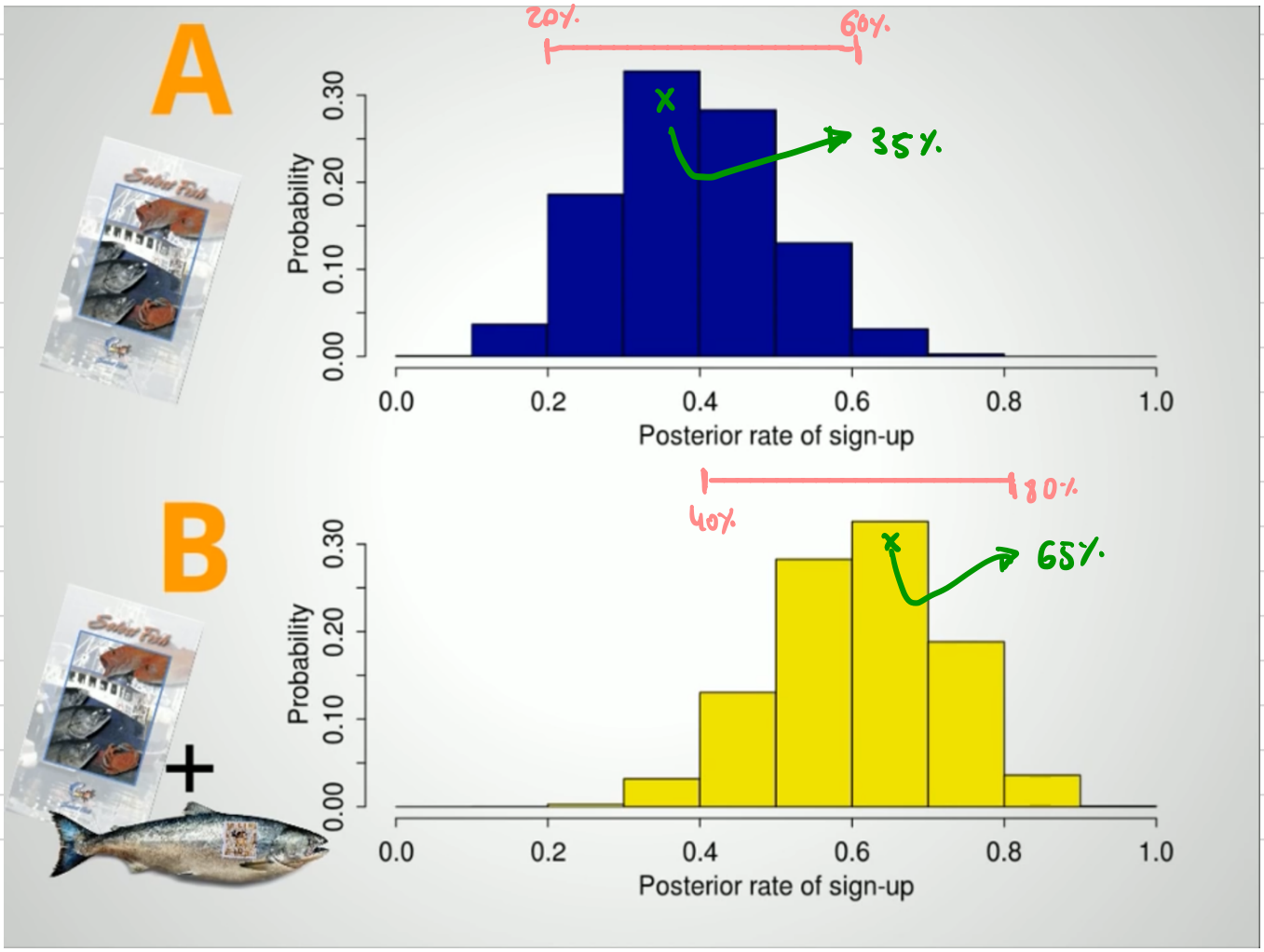
Which method is better?
How certain or uncertain are we about this?

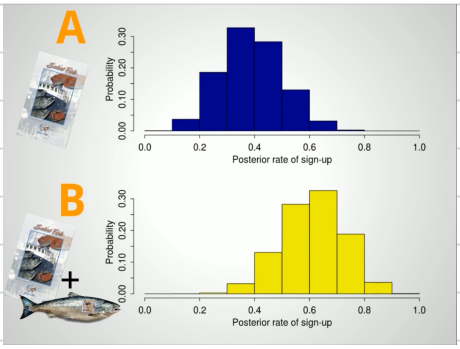


Run 1 $\theta_1 = 20\% \rightarrow 4$ $\theta_2 = 72\% \rightarrow 10$ ✗

↳ although θ_2 matches, we want the model to match reality for both

Run 2 $\theta_1 = 35\% \rightarrow 6$ $\theta_2 = 64\% \rightarrow 10$ ✓





By eyeballing the results of the 2 prior distributions, we do get the intuition that method B does achieve better rates.

But, as mentioned, this is an intuition

↳ we would want to calculate the probability that B is better than A.

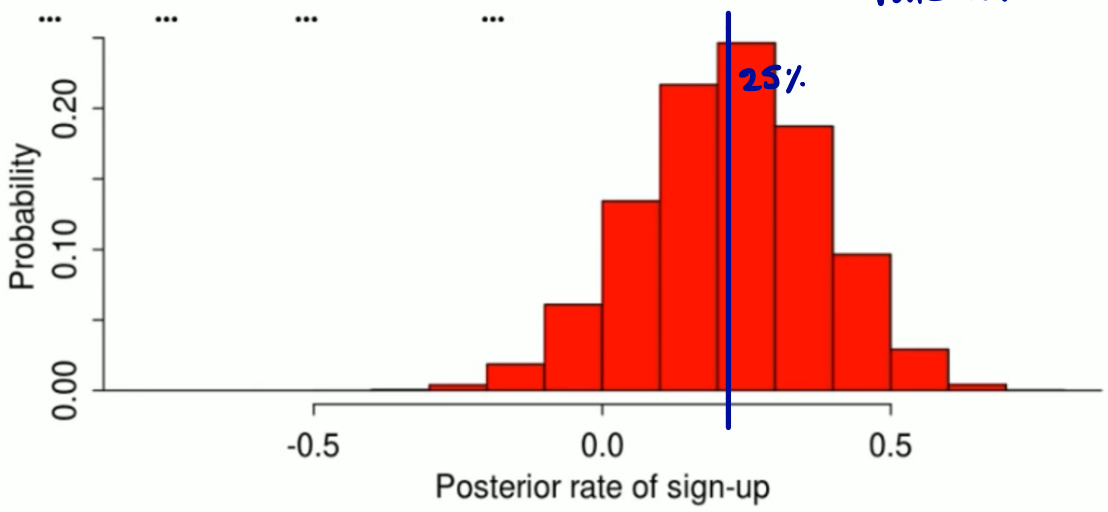
Let's lean on the underlying data for these 2 posterior distributions.

θ →	rateA	rateB	rate_diff
1	0.33	0.76	0.43
2	0.34	0.42	0.08
3	0.58	0.65	0.07
4	0.42	0.50	0.08
5	0.23	0.47	0.23
6	0.28	0.75	0.47
7	0.49	0.78	0.29
8	0.39	0.54	0.15
...	...	...	...

```
>rate_diff = rateB - rateA

> sum(rate_diff > 0) /
    length(rate_diff)
0.92
```

↳ 92% of the times, rate B is bigger than rate A.

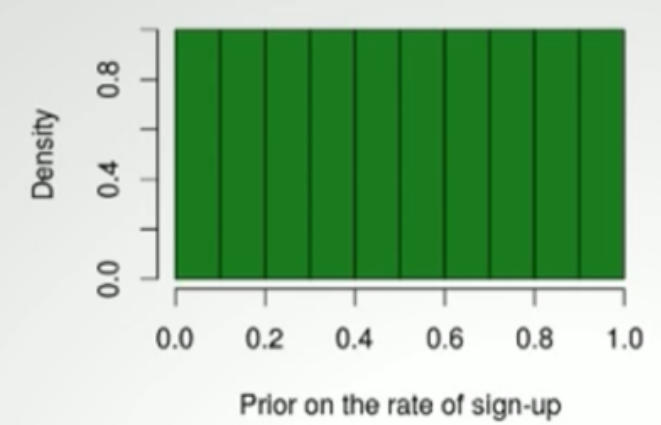


- Bayes also allows us to incorporate expert opinion (think of it as prior knowledge) to use in our analysis.

- Swedish Fish LTD CEO says: "sign up rate has never been higher than 20%. It's usually between 5% and 15%."

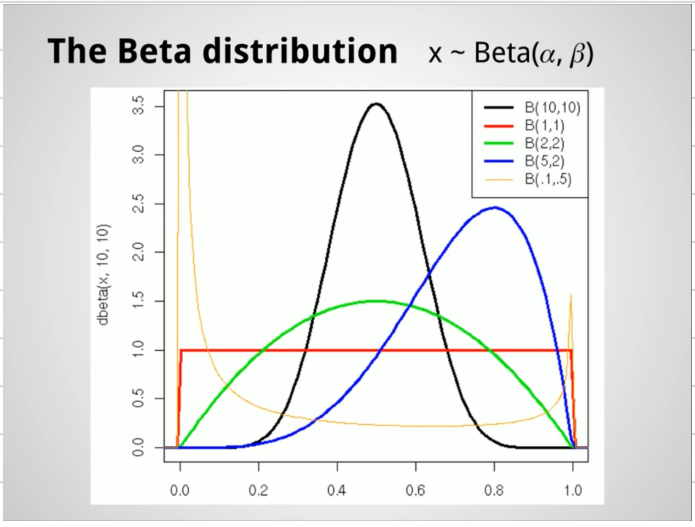
  ↳ Could this be useful information?
  ↳ How much can we trust this information?

↳ A natural place to include this information would be in the prior



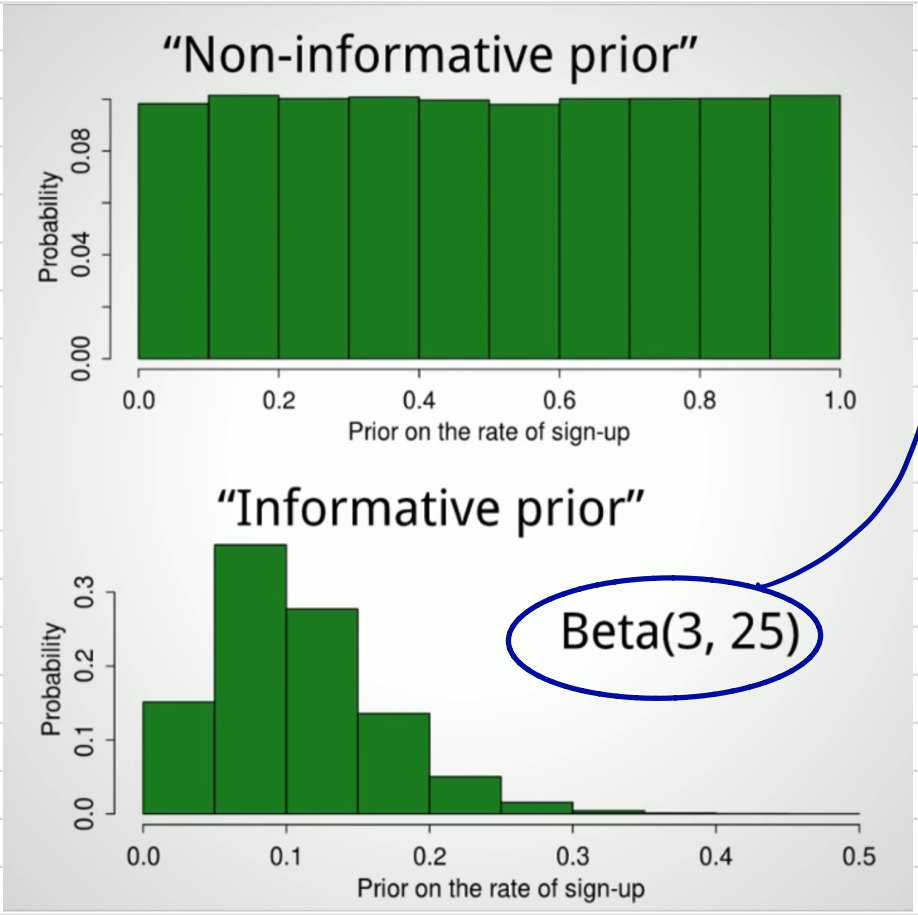
Instead of using a distrib. where any parameter has an equal probability of being selected

↳ we can model our desired prior distribution through the BETA DISTRIBUTION.



- Continuos distrib. bounded between 0 and 1.
  ↳ helpful because sign up rates can only be between 0,1

- You play around with the α and β parameters.

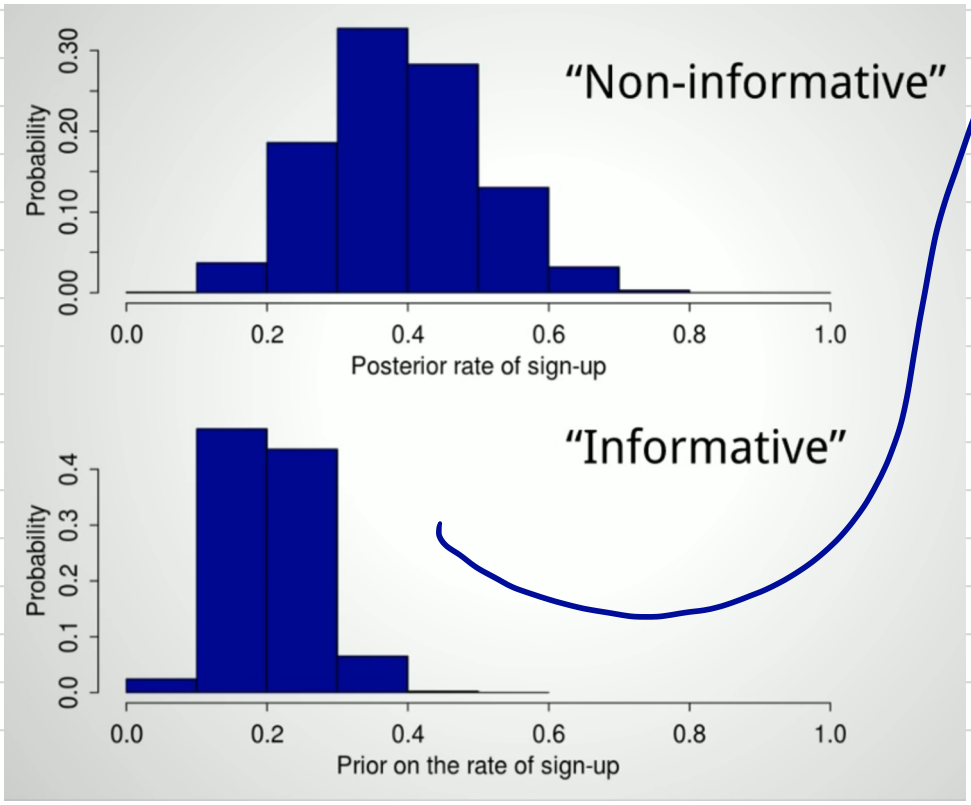


The actual params are not that important.

The key is finding a distribution that:

- Places a big probability that the rate of sign up is between 5% and 15%.
- Also doesn't rule out the chance that it could be as high as 35%.

Pushing this through the model we have been working with, we get:



→ The posterior looks now like a mix between actual sign-up rate and prior knowledge.

- the more data you have (now only 16 people), the less importance is given to the prior.
- The less data you have, the more importance the prior would have.

- Finally, the result of a Bayesian analysis retains the uncertainty of the estimated parameters, which is very useful in prediction and decision analysis.

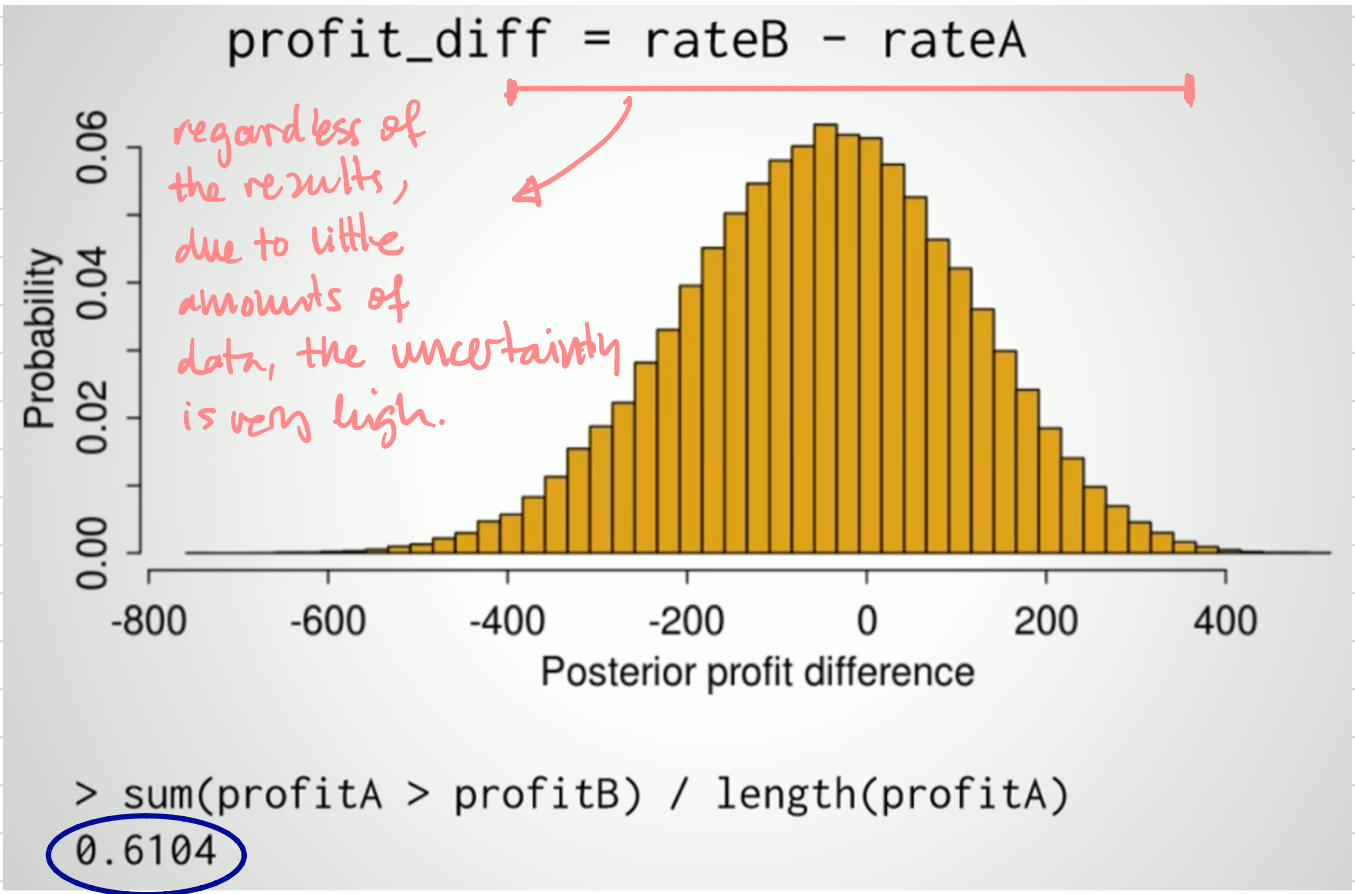
Our CEO returns with extra considerations:

- Cost of sending a brochure is 30kr
- Cost of sending a salmon is 300kr
- If a person signs up, we make 1000kr on average.

What should we do?

```
> profitA = rateA * 1000 - 30
> profitB = rateB * 1000 - 300
> profit_diff = profitB - profitA
```

	rateA	rateB	profitA	profitB	profit_diff
1	0.3316	0.7582	301.6	458.2	156.58
2	0.3424	0.4197	312.4	119.7	-192.71
3	0.5800	0.6496	550.0	349.6	-200.43
4	0.4171	0.4996	387.1	199.6	-187.46
5	0.2338	0.4683	203.8	168.3	-35.50
6	0.2810	0.7488	251.0	448.8	197.77
7	0.4881	0.7804	458.1	480.4	22.23
8	0.3888	0.5360	358.8	236.0	-122.78

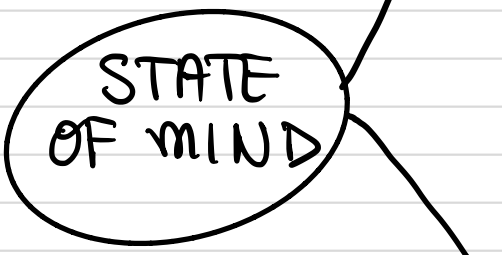



```
> sum(profitA > profitB) / length(profitA)
0.6104
```

method B might be better in terms of sign up rates, but it looks like method A is better in terms of profit!

Chapter 1 Notes.

The philosophy of Bayesian Inference



Frequentist → assumes probability is the long-term frequency of events

✓ When dealing with high frequency events, you have lot's of data to analyse long term frequency.

✗ When the event is rare, frequentist approaches may have problems.

Bayesian → interpret probability as a measure of belief or confidence of the event occuring

- When the facts change, I change my mind.
- A bayesian would update his or her beliefs after seeing evidence

For N → ∞, the more data we gather:

- statistical inference becomes more objective
- Bayesian results often align with frequentist results.

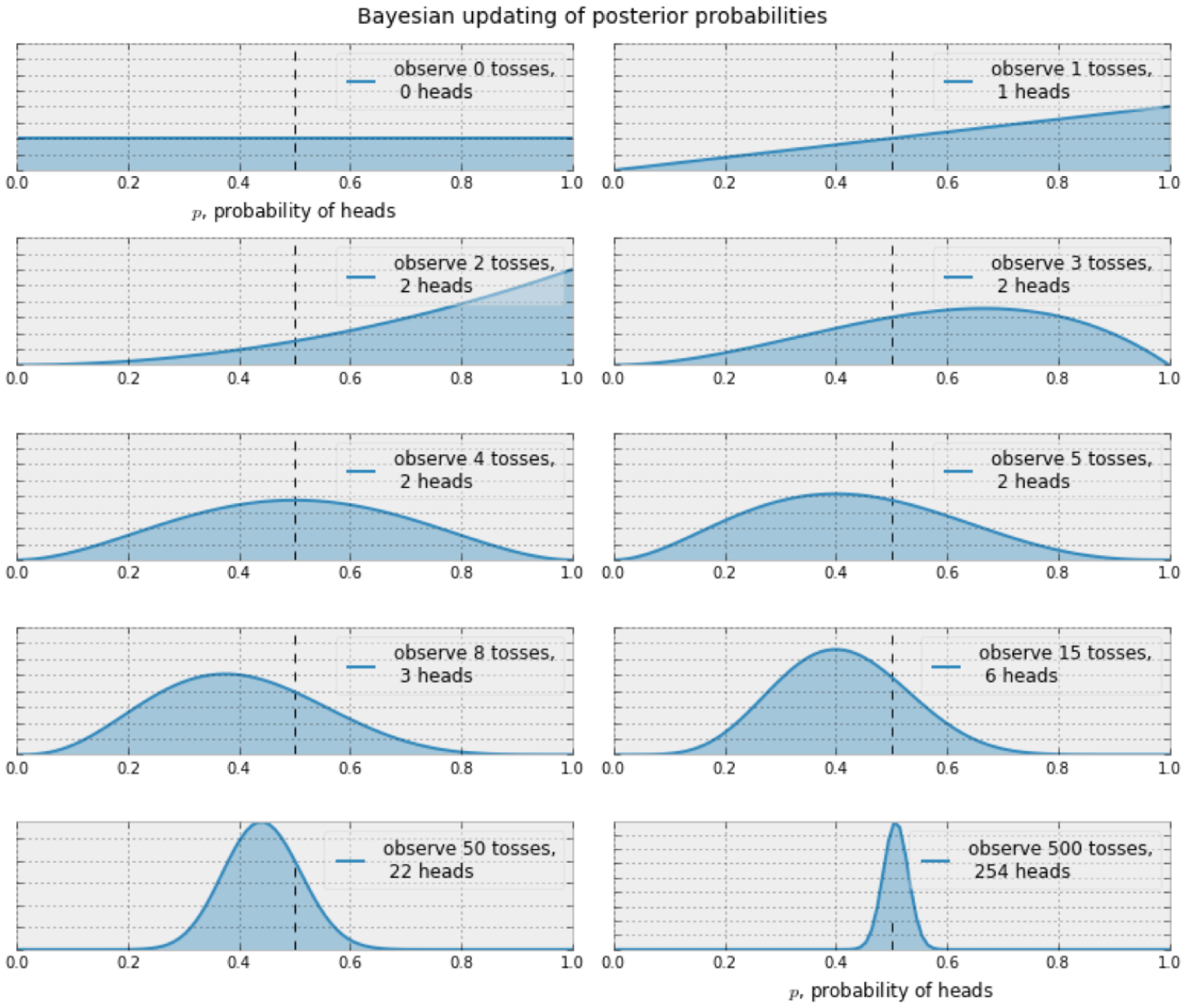
For N ↓↓, statistical inference is much more unstable

- This is where Bayesian excels.
- NOT because it provides ground truth.
- But because it returns probabilities (instead of pure scalars), and preserves uncertainty.

Example 1. - simple

How would a Bayesian thinker adapt their beliefs on a flip-coin example?

↳ We want to know what would the probability of heads be?

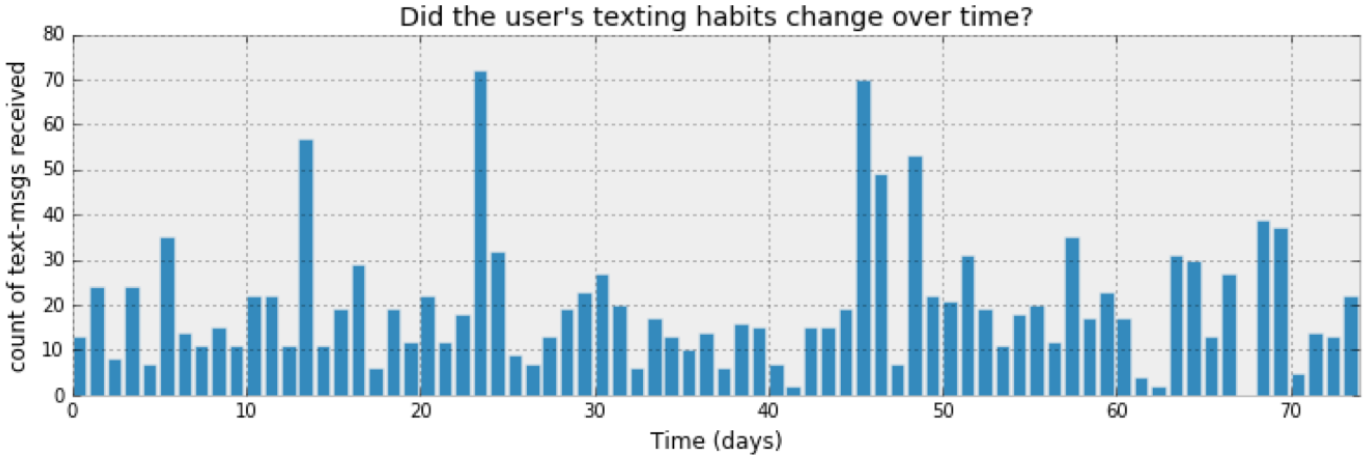


- Bayesians can update their prior beliefs with new piece of information.
- The more data you get, the more certain you are.
  ↳ But you will always have a range of possibilities (distribution), compared to a 50% scalar.

Example 2 - Bit more complicated

How would a Bayesian analyse if a user's texting habits change over the course of 70+ days?

① The data



② Distribution to model behaviour.

- Given we are dealing with count data → Poisson

$$C_i \sim \text{Poisson}(\lambda)$$

- For there to be a change in user behaviour, there must be a certain day where count patterns start changing

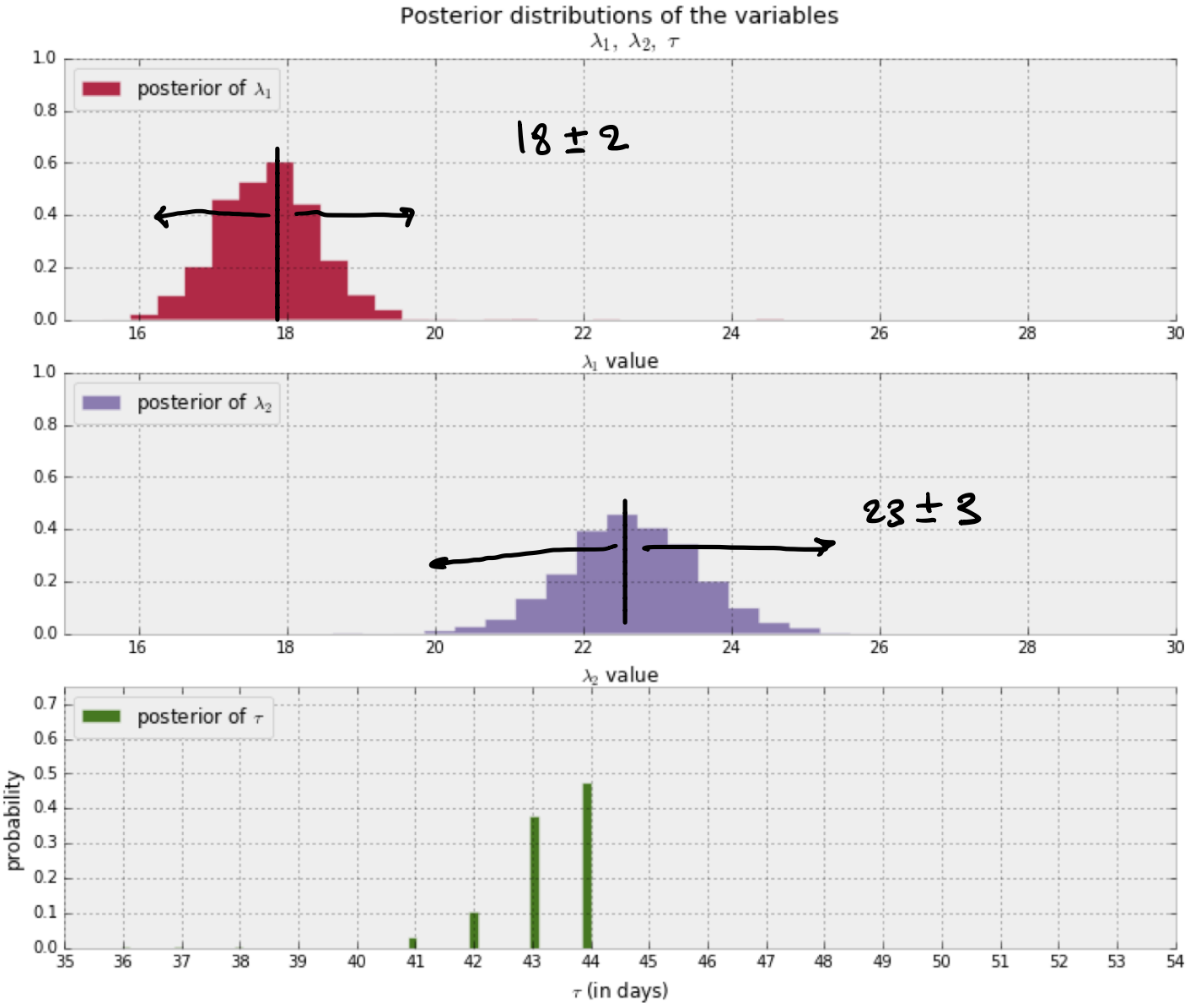
$$\lambda = \begin{cases} \lambda_1 & \text{if days} < \tau \\ \lambda_2 & \text{if days} \geq \tau \end{cases}$$

- let's assume too that λ can be described as an exponential

$$\lambda_1 \sim \text{Exp}(\alpha) \qquad \lambda_2 \sim \text{Exp}(\alpha)$$

→ we will run multiple Poisson distributions by changing these $\lambda_1, \lambda_2, \tau$ parameters and check what the simulation throws out.

③ λ_1, λ_2, τ results



- Immediately we can see:

1. λ_2 tends to be higher than λ_1
2. There is higher degree of uncertainty in λ_2 than λ_1
3. It seems that the possible change in texting behaviours occured in day 43 or 44. with a probability of ~39% and ~49% respectively.

Chapter 2 Notes.

Introduction of PyMC3.

MODEL CONTEXT

```
import pymc3 as pm

with pm.Model() as model:
    parameter = pm.Exponential("poisson_param", 1.0)
    data_generator = pm.Poisson("data_generator", parameter)
```

we can handle everything related with experimentation using a pymc3 MODEL context.

```
Applied log-transform to poisson_param and added transformed poisson_param_log_
to model.
```

```
with model:
    data_plus_one = data_generator + 1
```

you can add as many extra parameters to the context using WITH.

```
parameter.tag.test_value
```

→ you can access the defined objects outside of the model context.

```
array(0.693147177890573)
```

VARIABLES

```
print("parameter.tag.test_value =", parameter.tag.test_value)
print("data_generator.tag.test_value =", data_generator.tag.test_value)
print("data_plus_one.tag.test_value =", data_plus_one.tag.test_value)
```

Any variable defined in pymc3 will have an interval test value used for later sampling.

```
parameter.tag.test_value = 0.693147177890573
data_generator.tag.test_value = 0
data_plus_one.tag.test_value = 1
```

```
with pm.Model() as model:
    parameter = pm.Exponential("poisson_param", 1.0, testval=0.5)

print("\nparameter.tag.test_value =", parameter.tag.test_value)
```

This can be changed if one wants to play with it.

```
Applied log-transform to poisson_param and added transformed poisson_param_log_
to model.

parameter.tag.test_value = 0.49999999904767284
```

PyMC3 deals with 2 types of variables

- **Deterministic** → if known parameters, then always same results
  $y = x + 1$
- **Stochastic** → even if you know all parameters, results would still be random → Poisson, Discrete Uniform, etc.

Working with Stochastic variables in pyMC3.

Definition: `some_variable = pm.DiscreteUniform("discrete_uni_var", 0, 4)`

name — params.

For multivariate problems where we have multiple distributions defined by different parameters:

```
beta_1 = pm.Uniform("beta_1", 0, 1)
beta_2 = pm.Uniform("beta_2", 0, 1)
...
```

→ `betas = pm.Uniform("betas", 0, 1, shape=N)`

Working with deterministic variables in pymc3

```
deterministic_variable = pm.Deterministic("deterministic variabl
e", some_function_of_variables)
```

For example, $\lambda = \begin{cases} \lambda_1 & \text{if } t < \tau \\ \lambda_2 & \text{if } t \geq \tau \end{cases}$ — if we known $\lambda_1, \lambda_2, \tau$ then λ is completely known and specified.

```
import numpy as np

n_data_points = 5  # in CH1 we had ~70 data points
idx = np.arange(n_data_points)
with model:
    lambda_ = pm.math.switch(tau >= idx, lambda_1, lambda_2)
```

PyMC3 is built on top of Theano.

- Whilst Numpy directly executes a+b, theano builds a computation graph only executed when evaluation is required.
- This computation graph helps PyMC3 be run on GPUs.

Including observations in the model

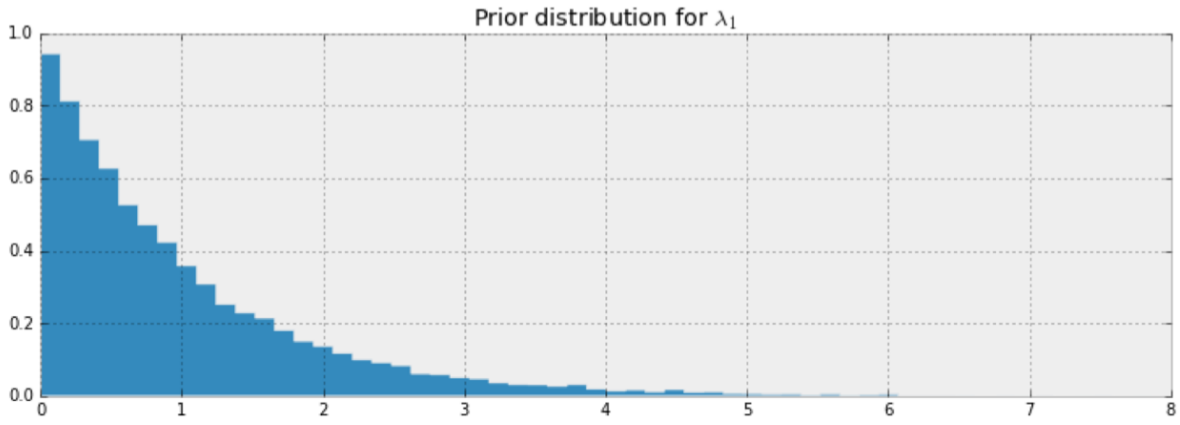
Deterministic

```
import numpy as np

n_data_points = 5  # in CH1 we had ~70 data points
idx = np.arange(n_data_points)
with model:
    lambda_ = pm.math.switch(tau >= idx, lambda_1, lambda_2)
```

```
%matplotlib inline
from IPython.core.pylabtools import figsize
import matplotlib.pyplot as plt
import scipy.stats as stats
figsize(12.5, 4)

samples = lambda_1.random(size=20000)
plt.hist(samples, bins=70, normed=True, histtype="stepfilled")
plt.title("Prior distribution for $\lambda_1$")
plt.xlim(0, 8);
```



Stochastic.

Stochastic variables have a keyword argument OBSERVED

It is used to fix the variable's current value to the given known data.

```
# We're using some fake data here
data = np.array([10, 25, 15, 20, 35])
with model:
    obs = pm.Poisson("obs", lambda_, observed=data)
print(obs.tag.test_value)
```

```
[10 25 15 20 35]
```

Defining an A/A example.

- 1. set the prior knowledge (in this case no prior knowledge)

```
import pymc3 as pm

# The parameters are the bounds of the Uniform.
with pm.Model() as model:
    p = pm.Uniform('p', lower=0, upper=1)
```

→ PRIOR

```
Applied interval-transform to p and added transformed p_interval_ to model.
```

- 2. Consider the true success rate is 5% and we testing on 1500 users we will simulate this using a Bernoulli distribution (as the outcome is buy or not buy)

```
#set constants
p_true = 0.05  # remember, this is unknown.
N = 1500

# sample N Bernoulli random variables from Ber(0.05).
# each random variable has a 0.05 chance of being a 1.
# this is the data-generation step
occurrences = stats.bernoulli.rvs(p_true, size=N)

print(occurrences) # Remember: Python treats True == 1, and False == 0
print(np.sum(occurrences))
```

nothing here pymc3. This is just data simulation.

```
[1 0 1 ..., 0 0 0]
77
```

```
# Occurrences.mean is equal to n/N.
print("What is the observed frequency in Group A? %.4f" % np.mean(occurrences))
print("Does this equal the true frequency? %s" % (np.mean(occurrences) == p_true))
```

```
What is the observed frequency in Group A? 0.0513
Does this equal the true frequency? False
```

- 3. Combine simulated data with our prior knowledge

```
#include the observations, which are Bernoulli
with model:
    obs = pm.Bernoulli("obs", p, observed=occurrences)
    # To be explained in chapter 3
    step = pm.Metropolis()
    trace = pm.sample(18000, step=step)
    burned_trace = trace[1000:]
```

→ DATA

↳ SAMPLING.

```
 [-------100%-------] 18000 of 18000 in 1.7 sec. | SPS: 10329.7 | ETA: 0.0
```

4. Checking the posterior results for A.

```
figsize(12.5, 4)
plt.title("Posterior distribution of $p_A$, the true effectiveness of site A")
plt.vlines(p_true, 0, 90, linestyle="--", label="true $p_A$ (unknown)")
plt.hist(burned_trace["p"], bins=25, histtype="stepfilled", normed=True)
plt.legend();
```

= 0.05

The nice thing about Bayesian thinking is that we return a possible maximum likehood result given certain prior knowledge, but also how uncertain are we about this.

A/B test example

Defining observations (no need for pymc3)

```
import pymc3 as pm
figsize(12, 4)

#these two quantities are unknown to us.
true_p_A = 0.05
true_p_B = 0.04

#notice the unequal sample sizes -- no problem in Bayesian analysis.
N_A = 1500
N_B = 750

#generate some observations
observations_A = stats.bernoulli.rvs(true_p_A, size=N_A)
observations_B = stats.bernoulli.rvs(true_p_B, size=N_B)
print("Obs from Site A: ", observations_A[:30], "...")
print("Obs from Site B: ", observations_B[:30], "...")
```

← key!

```
Obs from Site A:  [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
...
Obs from Site B:  [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0]
...
```

```
print(np.mean(observations_A))
print(np.mean(observations_B))
```

```
0.042
0.0346666666667
```

```
# Set up the pymc3 model. Again assume Uniform priors for p_A and p_B.
with pm.Model() as model:
    p_A = pm.Uniform("p_A", 0, 1)
    p_B = pm.Uniform("p_B", 0, 1)

    # Define the deterministic delta function. This is our unknown of interest.
    delta = pm.Deterministic("delta", p_A - p_B)

    # Set of observations, in this case we have two observation datasets.
    obs_A = pm.Bernoulli("obs_A", p_A, observed=observations_A)
    obs_B = pm.Bernoulli("obs_B", p_B, observed=observations_B)

    # To be explained in chapter 3.
    step = pm.Metropolis()
    trace = pm.sample(20000, step=step)
    burned_trace=trace[1000:]
```

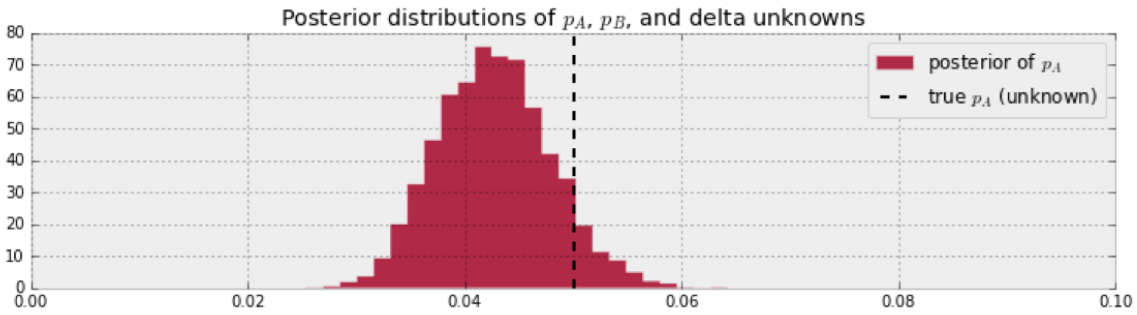
prior definition

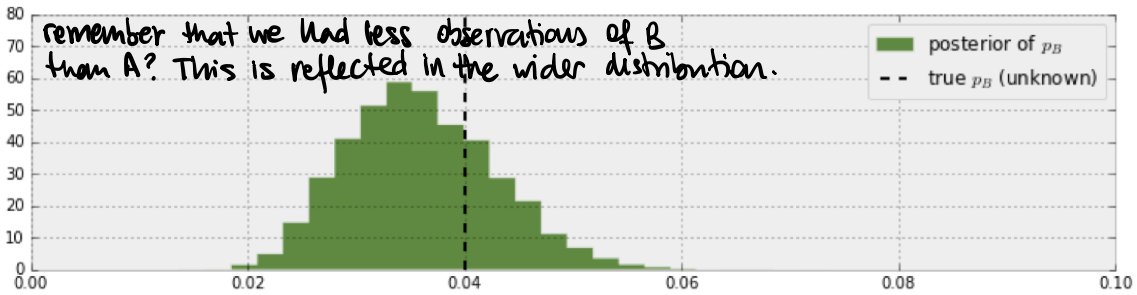
difference between the obtained sampling results

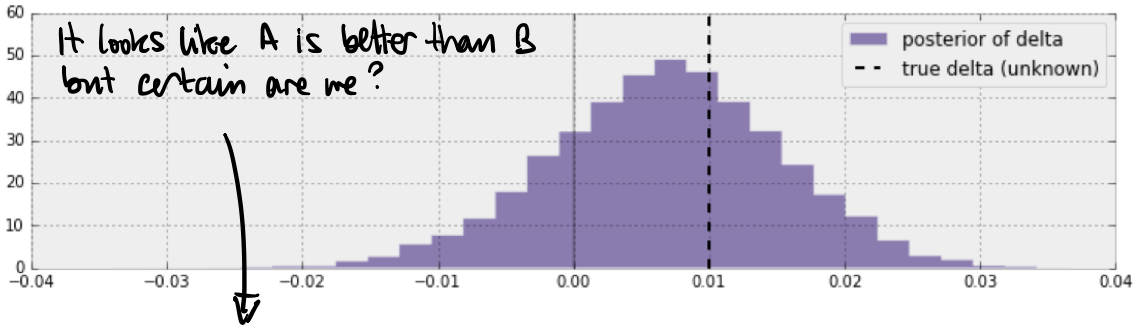
data samples from before

sampling.

```
p_A_samples = burned_trace["p_A"]
p_B_samples = burned_trace["p_B"]
delta_samples = burned_trace["delta"]
```








```
# Count the number of samples less than 0, i.e. the area under the curve
# before 0, represent the probability that site A is worse than site B.
print("Probability site A is WORSE than site B: %.3f" % \
    np.mean(delta_samples < 0))

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

```
Probability site A is WORSE than site B: 0.208
Probability site A is BETTER than site B: 0.792
```

Chapter 3 Notes.

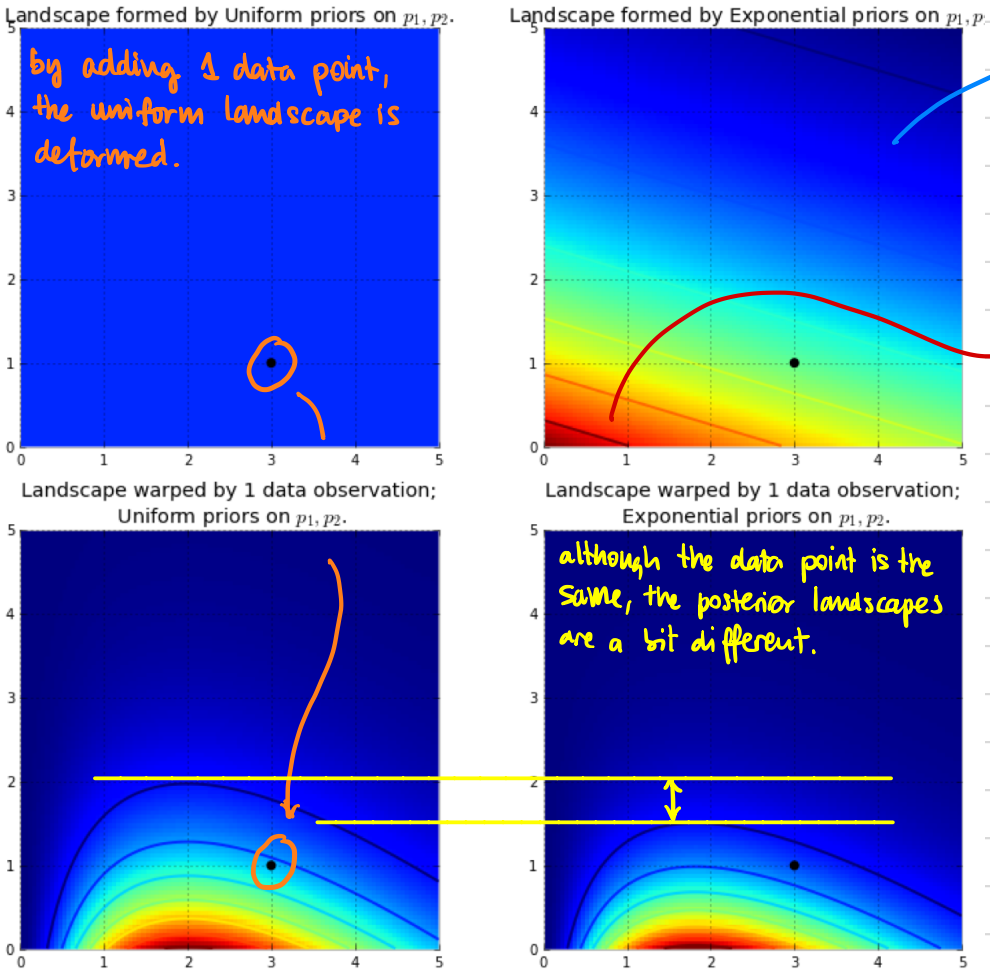
Opening the black box of MCMC (Markov Chain - Monte carlo).

Why understand MCMC?

1) Any book about Bayesian inference must discuss MCMC
2) Knowing the process of MCMC gives you an insight on whether your process has converged.
3) Understand why we work with thousands of samples from the posterior as a solution.

As a reminder to what we have seen.

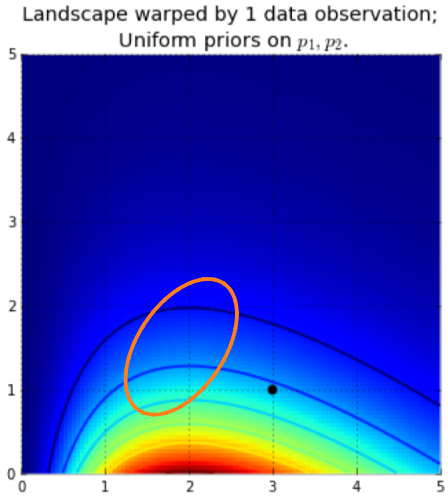
- Prior beliefs or knowledge can be described with a certain distribution.
- When we begin to incorporate data, our beliefs change and the distribution gets pulled or pushed on different areas.



Both posteriors try to capture the data point within their higher probability space of parameters.

The question is, if 1 data point already has this effect, what would happen if we had 25 or 2500 data points?

Exploring the landscape using MCMC.



- We should explore the deformed posterior space generated by our prior surface + observed data to find the posterior mountain.
- However, it is nearly impossible computationally (N-dimensional space is exponentially difficult in N) to naively search ALL parameter space.
- MCMC tries to perform an intelligent search of the space.

- MCMC returns SAMPLES from the posterior distribution, NOT the actual distribution itself.
  (in MCMC lingo, the collection of SAMPLES is called TRACES)

  ↳ What MCMC is trying to do is explore positions that are close by and move into the areas with higher probability.

  ↳ MCMC hopes then to "converge" towards these "real" areas of high probability.
  (converging might not be a great terms as it implies moving towards a point in space, whereas MCMC moves towards a BROADER AREA).

Question → Isn't exploring thousands of samples inefficient?
Counter argument.

1) The alternative of returning a mathematical formula that quasi-fully describes the whole space would involve describing an N-dimensional surface with arbitrary peaks and valleys.
   ↳ Difficult.
2) You could mathematically only look for the 'peak of the mountain', but you would ignore the shape of the landscape (wouldn't be able to get a view of certainty)
3) The Law of Large Numbers is on our side (Chapter 4).

MCMC exploration in simple pseudo-code.

1. Start at current position
2. Propose moving to a nearby close position
3. Accept/Reject moving to this position based on the adherence to the prior and the data.
4. A → If you accept move to new position.
   B → If you reject, don't move.
5. Return to step 1.
6. After large number of iterations, return all accepted positions.

- At beggining (random location in space) the algorithm will probably move towards areas that are not a good reflection of the "real" posterior, but they are definetily better than random
- It is when N ↑↑, that we are hopefully moving towards the right positions.

Maximum a posterior (MAP) to the rescue

From step 1 in the pseudo code, we observe that the results of MCM are dependant on the starting position
- ↳ Different MCMC runs can provide different results.
- ↳ Poor starting values can prevent or slow down convergence.

Ideally, we would like MCMC at the peak of the landscape, as we would avoid a lengthy burn-in period.
- ↳ of course this peak is unknown, but PyMC3 provides a function to approximate it's value.

```
find_MAP(fmin=scipy.optimize.fmin_powell)
```

you can also define other optimization algorithms.

```
start = pm.find_MAP()
trace = pm.sample(2000, step=pm.Metropolis, start=start)
```

Consideration → discard the first samples as they "introduce" noise.

```
with pm.Model() as model:
    start = pm.find_MAP()

    step = pm.Metropolis()
    trace = pm.sample(100000, step=step, start=start)

burned_trace = trace[50000:]
```

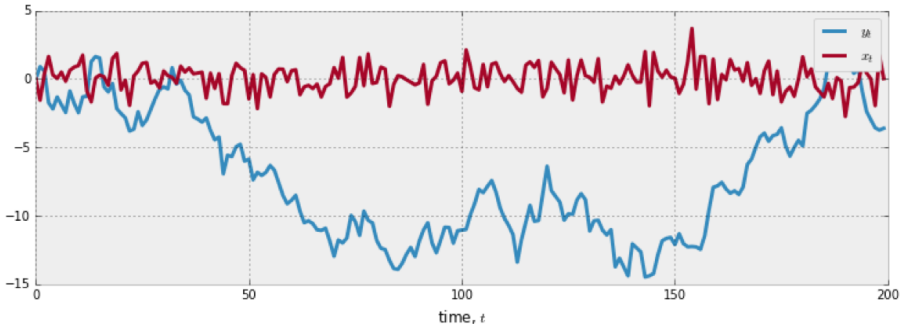
discard the first 50% of samples.

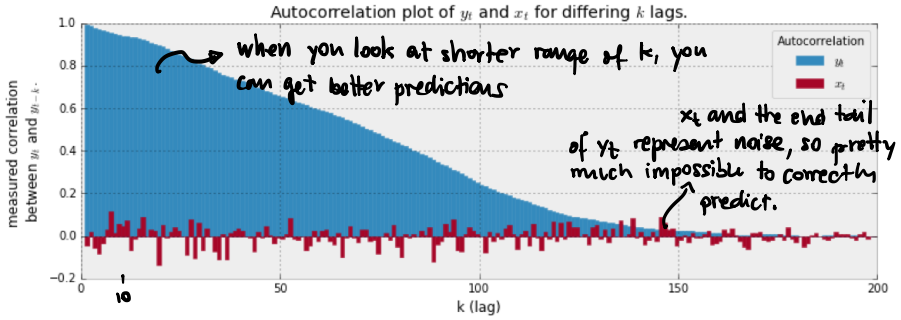
perfect positive correlation

perfect negative correlation

Diagnosing convergence.

1. Autocorrelation → how related is a series of numbers with itself. [-1, 1]

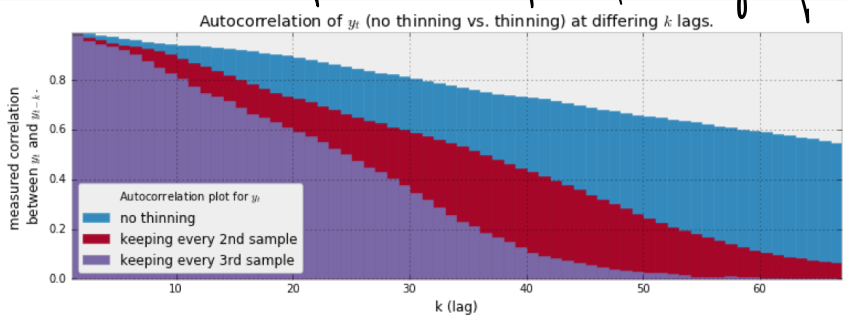




How does this related to MCMC?
→ MCMC will always return samples that exhibit autocorrelation.
→ This is because you are dependant and moving towards nearby positions.
→ A MCMC chain that is not exploring space well will have high autocorrelation (very dependant on previous parameter space).
→ If autocorrelation is low, this might indicate that a broader area of params have been explored and return confident results.

=> Converge of MCMC DOES NOT imply low autocorrelation, but it is an indicative.
↳ Therefore → not necesary, but sufficient.

2. Thinning. → Having high autocorrelation between posterior samples can introduce problems if post-procesing algorithms require samples to be independent from each other. By thinning (selecting every n-th sample) you reduce autocorrelation a bit.



x What thinning? 10k samples with a thinning of 10 requires 100k samples.

x Any amount of thinning will have a level of autocorr. The key is checking that it tends to 0.

→ 4 different plots.

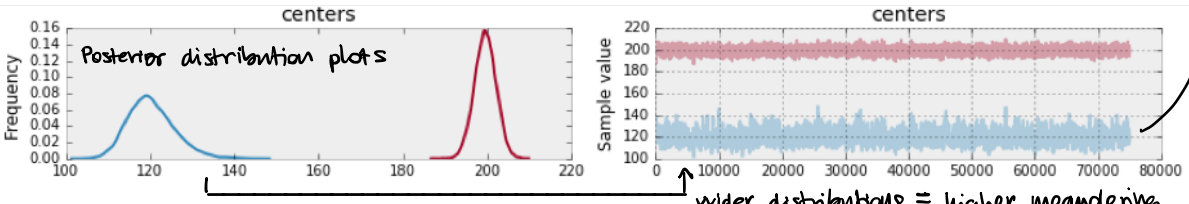
Some PyMC3 plots.

pass a trace returned from sampling

variable names you are interested in

```
1) pm.plots.traceplot(trace=trace, varnames=["centers"])
2) pm.plots.plot_posterior(trace=trace["centers"][:,0])
3) pm.plots.plot_posterior(trace=trace["centers"][:,1])
4) pm.plots.autocorrplot(trace=trace, varnames=["centers"]);
```

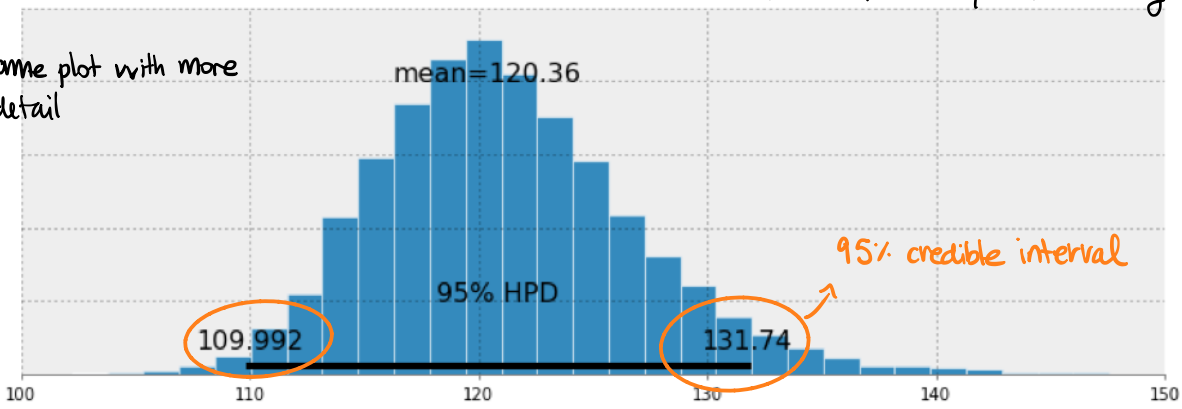
1)



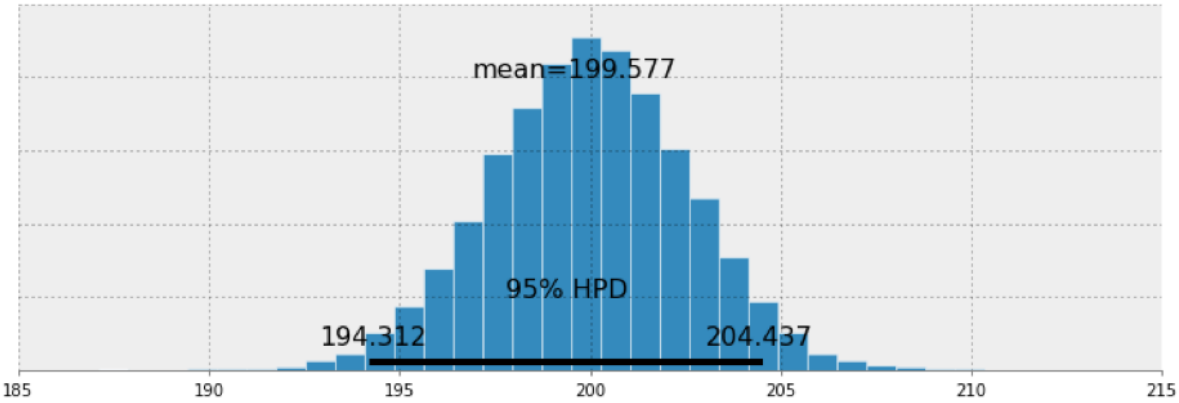
the more "meandering", the higher the chances of non-convergence

wider distributions ≡ higher meandering

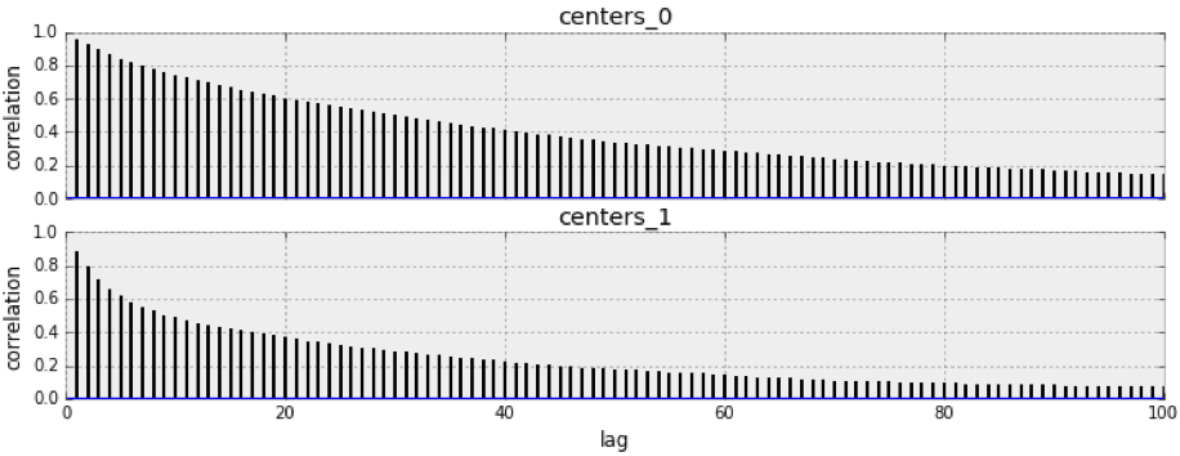
2) same plot with more detail



3)



4)

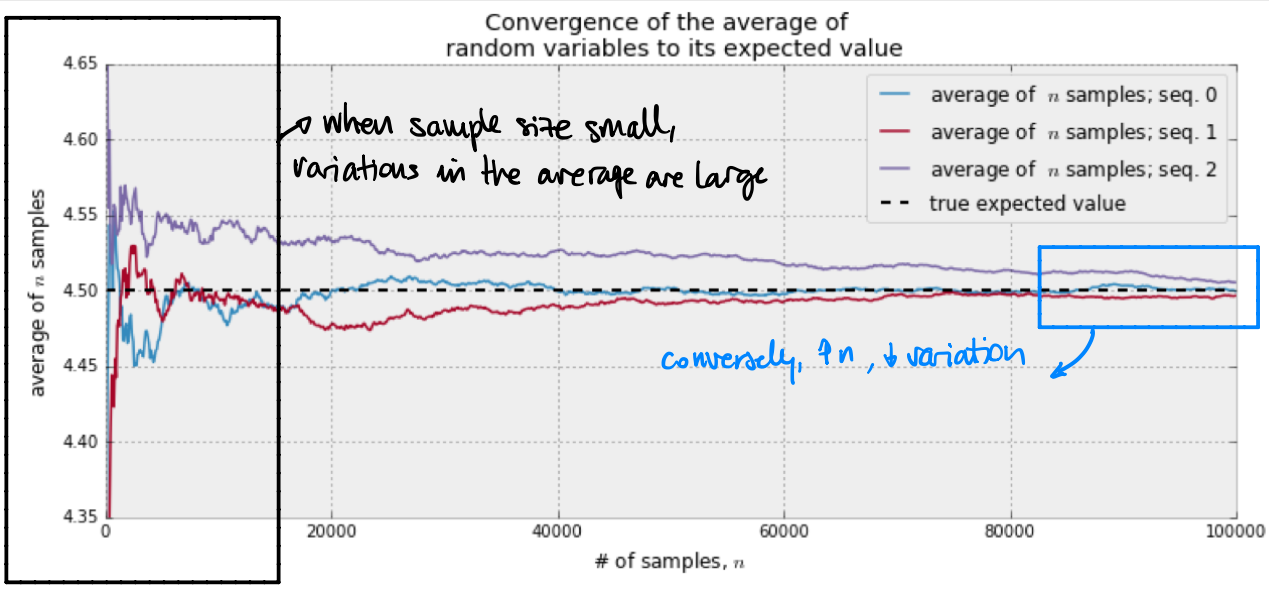


Simple auto-correlation plot

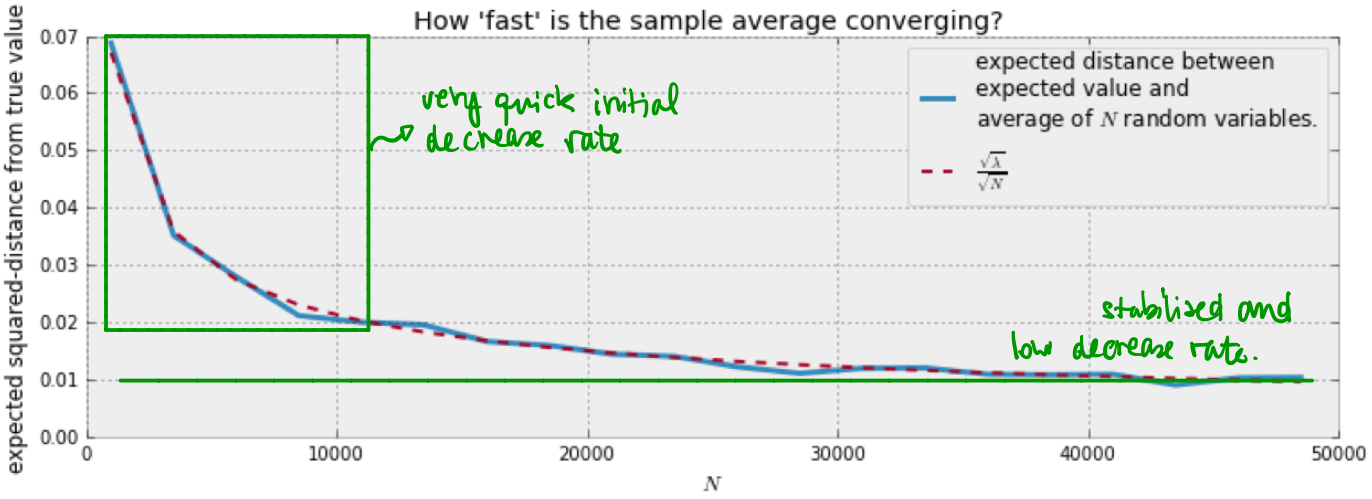
Chapter 4 Notes.

The Law of Large Numbers.

A very simple look.



But, what is a big number of samples? In the limit (∞), we would expect variation to be 0, but we can't sample ∞ times! So, how quickly is my avg. sampling converging?



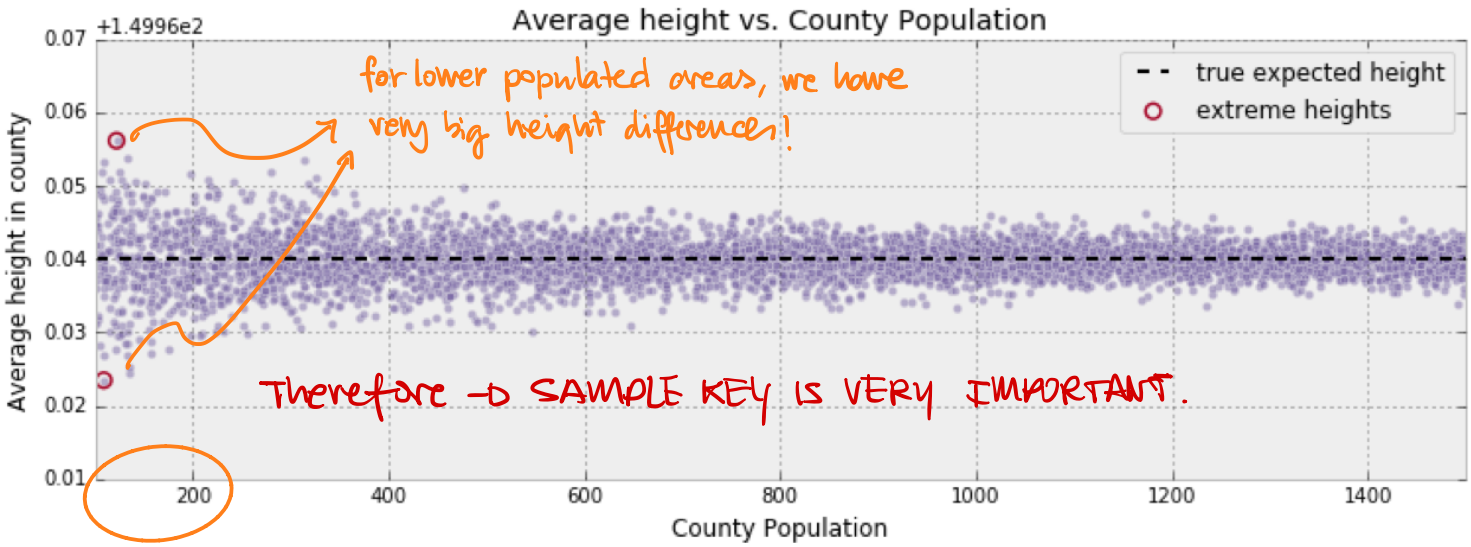
How does this relate to Bayesian statistic?

- Expected values require, most of the time, evaluation of complicated multi-integrals.
- Bayesian point estimates (chapter 5), help here.
  - ↳ By sampling from the posterior distribution, we simple need to evaluate averages
  - ↳ Again, when N is not large, we know these average sampling will be noisy, that is why uncertainty is key in Bayesian statistics.

Example where Law of Large Numbers fail.

Aggregated demographic data. Imagine:

- → 5k geographic areas.
- → Population in each range from 100 to 1500
- → We know that height does NOT vary across county.
- → If we aggregate heights in each area:

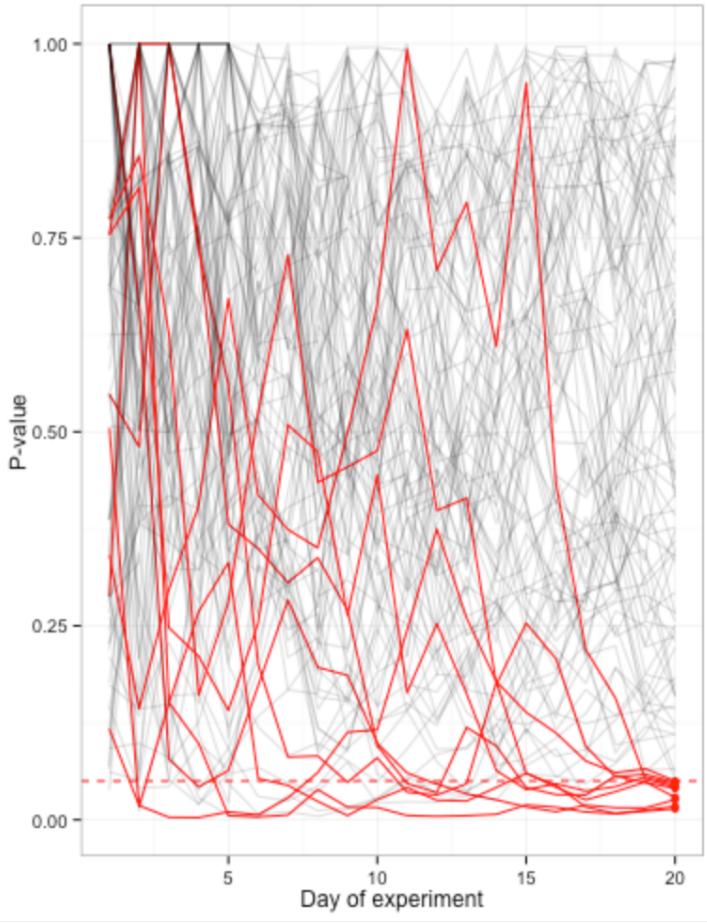


Stopping rules in Bayesian Testing

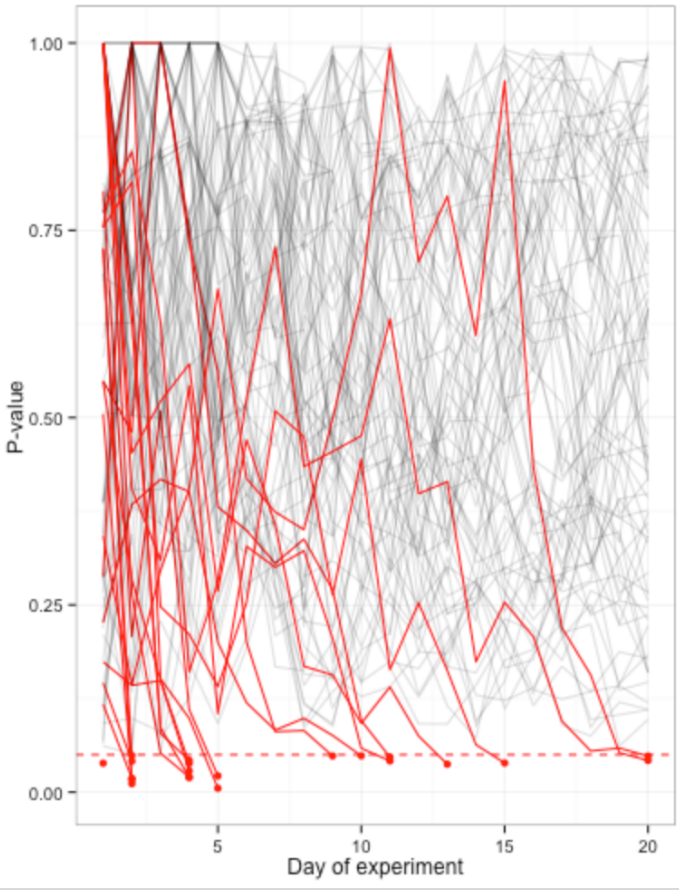
Is Bayesian A/B testing immune to peeking?

1. Review of classical A/B testing stopping considerations.

- Classical A/B testing relies on p-values as a decision criterion
- If one evaluates the p-value every day, one will incur in higher probability of finding a "significant" p-value, which might in fact only be produced due to noise.
- One solution is to pre-commit to running your experiment for a particular amount of time.
  - ↳ but again
    - ↳ what if there is a clear winner?
    - ↳ what if we are clearly doing some damage?



↳ Leaving our experiment to run for 20 days, only 5.34% of the test simulations fell below 0.05 threshold.



↳ If we stopped early (one 0.05 has been reached), 22.68% of the simulations would have been called "significant"!!

2. Bayesian Expected loss - doesn't fully avoid the peeking problem.

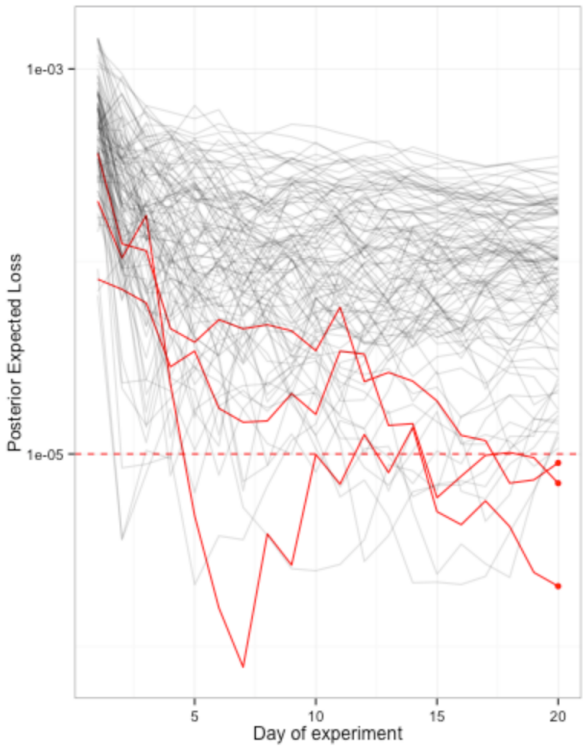
What is expected loss?

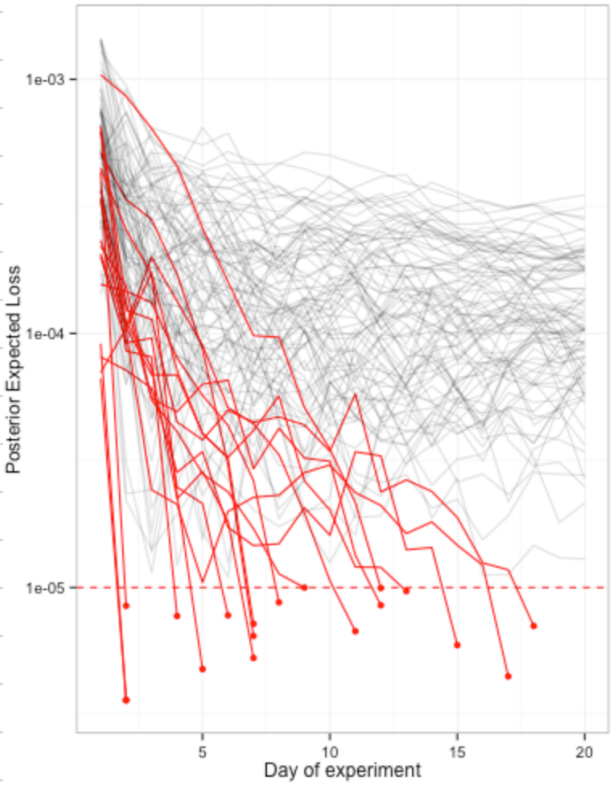
- It is a combination of how probable B has a lower success rate than A and, if B is worse, how much worse it is on average.
  - ↳ If there is a 10% chance that B is worse than A, and when it is worse, it decreases clickthrough rate by 0.0001
    - ↳ average loss → $10 \cdot 0.0001 = 10^{-5}$

- As a side comment, notice that we are totally ignoring any potential upside → the rule doesn't consider how much better A could be.

Testing with expected loss.

- Suppose we are testing a new feature that would actually decrease success rate (10.1% → 0.09%)
- Our threshold of caring is 10^{-5}





- Waiting 20 day → 2.5% cases
- Looking every day → 11.8% cases

↳ Peeking not solved as we have increased x4 our chances of passing a test that we know would decrease our success rate!

3. Why is this happening.

- Bayesian methods never claimed to fix type I error rates.
  - ↳ their goals is set about an expected loss
- Bayesian can heavily depend on the priors.
  - ↳ in the previous experiment, we knew that B was worse than A, but we allowed our initial priors to think that either could be better than the other.

4. First conclusion

- Focusing solely on expected loss can still lead to problems.
- A possible approach would be to include also the idea of how many times is B better than A (posterior probability $Pr(B > A)$.

Stopping based on a precision Bayesian credible interval

- In the previous section we briefly introduced that Bayes is not immune to peeking when focusing solely on expected loss.
- Nevertheless, Bayesian is still better than classical A/B testing in terms of improving immunity to peeking:
  - ↳ using Bayesian HDI with ROPE or Bayes factor, the false alarm rate asymptotes at a level far less than 100%. (maybe around 20-25%)
- However, let's test 4 methods through simulations and check what the results return.

1. Experiment.

- We are going to carry out peeking in a series of coin flips
- 4 stopping rules (forget for the moment on how to calculate certain parameters):
  1. NHST (sequential peeking). For every new flip, stop and reject the null hypothesis that $\theta = 0.50$ if $p < 0.05$. Otherwise flip again.
  2. Bayes Factor. For every flip, conduct a Bayesian model comparison of the null hypothesis that $\theta = 0.50$ versus the alternative hypothesis that there is a uniform prior. If $BF > 3$, accept null and stop. If $BF < 1/3$, reject null and stop. Otherwise flip.
  3. Bayesian HDI with ROPE. For every flip, compute the HDI on θ. If HDI is completely in a ROPE from 0.45 to 0.55, stop and accept null. If HDI falls completely outside of ROPE, stop and reject null. Otherwise, continue.
  4. Precision: For every flip, compute the 95% HDI on θ. If it's width is less than 0.8 * width of ROPE, stop. Otherwise flip. Once stopped, check whether null can be accepted or rejected according to HDI with ROPE criteria.

- Run MonteCarlo simulations of flipping coin with bias $\theta = 0.5$.
- 1000 simulated experiments.
- Sample size to maximum of 1500 flips.

- The set of charts presented in the next page, show the results for when the null hypothesis is true.
  - ↳ we would like to see, out of the 4 methods, which one rejects the null hypothesis the least.
  (we know that the distribution of coins flips is $\theta = 0.5$).

— undecided → keep flipping.
— proportion decisions to accept null
— proportion decisions reject null.

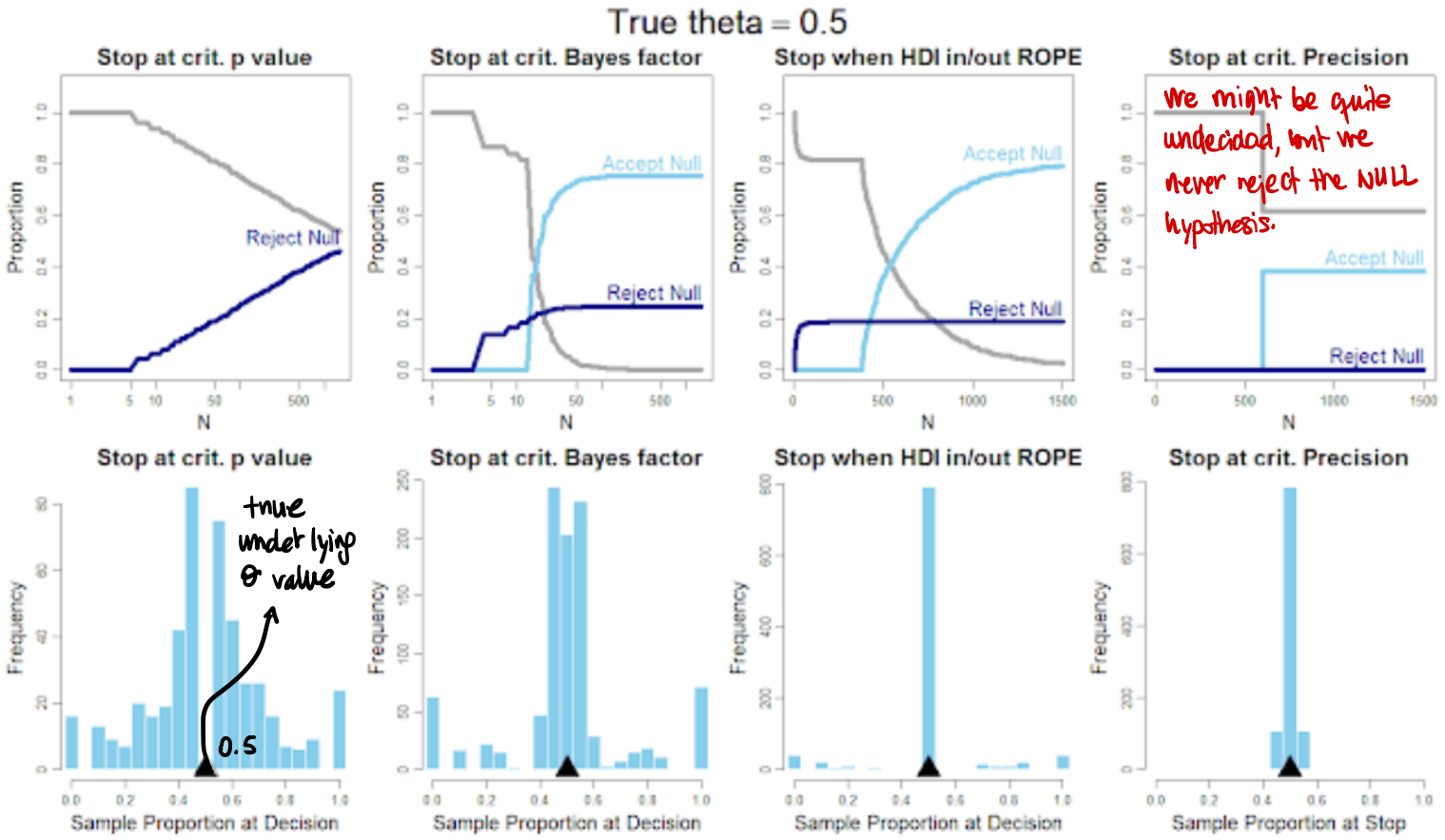


Figure 1

- p-values never accept the null hypothesis (only reject).
- you can see how this rejection increases on a logarithmic scale.

- Both stopping criterias yield similar results in terms of rejecting the (true) null hypothesis.
  - ↳ Bayes factor does it with very small samples.
- The moment we capture larger N, the rejection never happens and it only accepts the null.

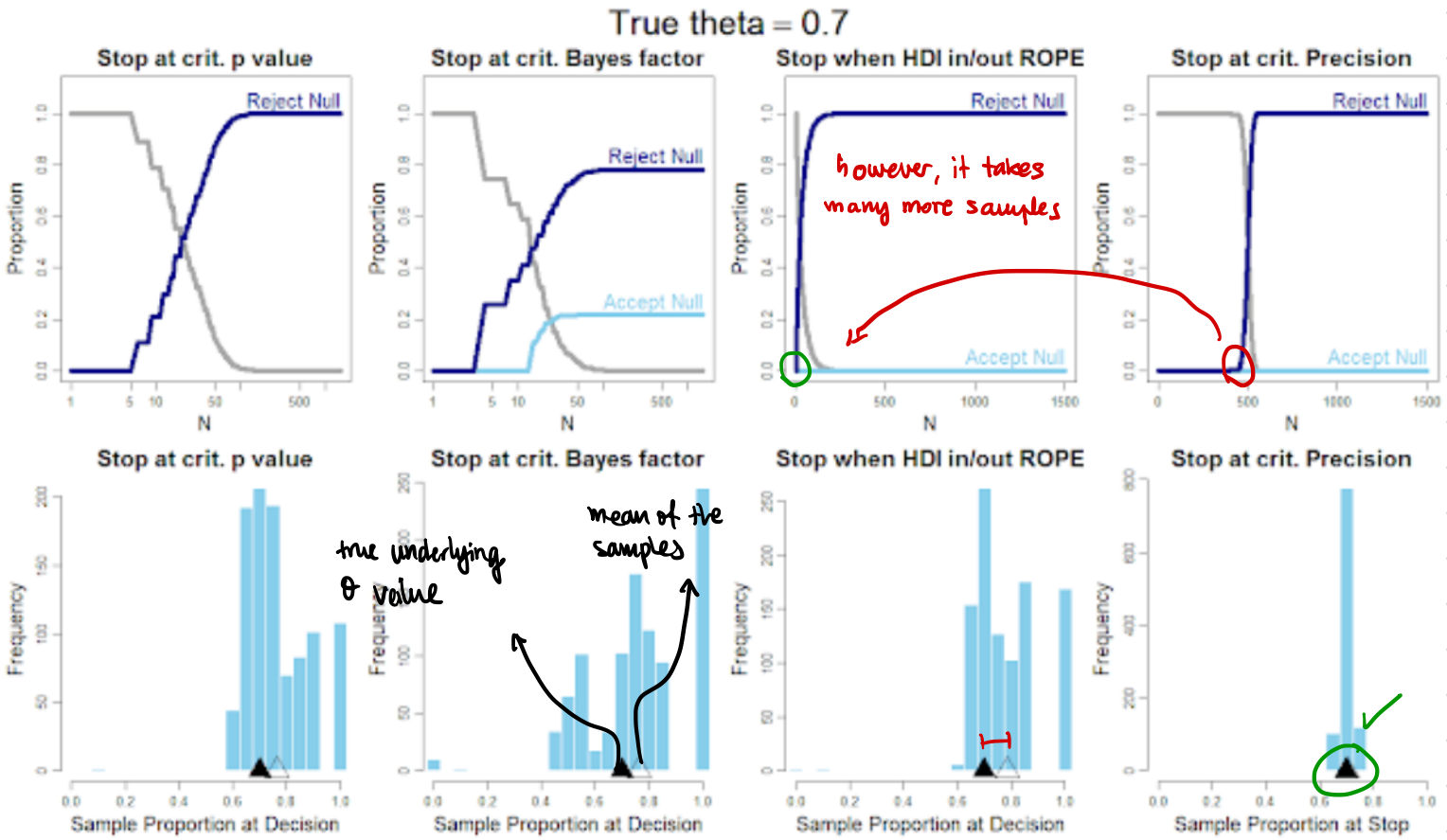
- Compared to the 2 methods on the left, a bigger sample size is needed to begin taking decisions.
- However, decisions are much more accurate with very low rates of null hypothesis rejection.

✓ winner

But what if our underlying succes rate (θ) was biased?

↳ we know that flipping a coin $\theta = 0.5$, but let's test what the methods throw when $\theta = 0.7$.

↳ we would expect a high proportion of rejecting the null hypothesis that $\theta = 0.7$.



↳ Bayes factor still accepts the null hypothesis 20% of the time!

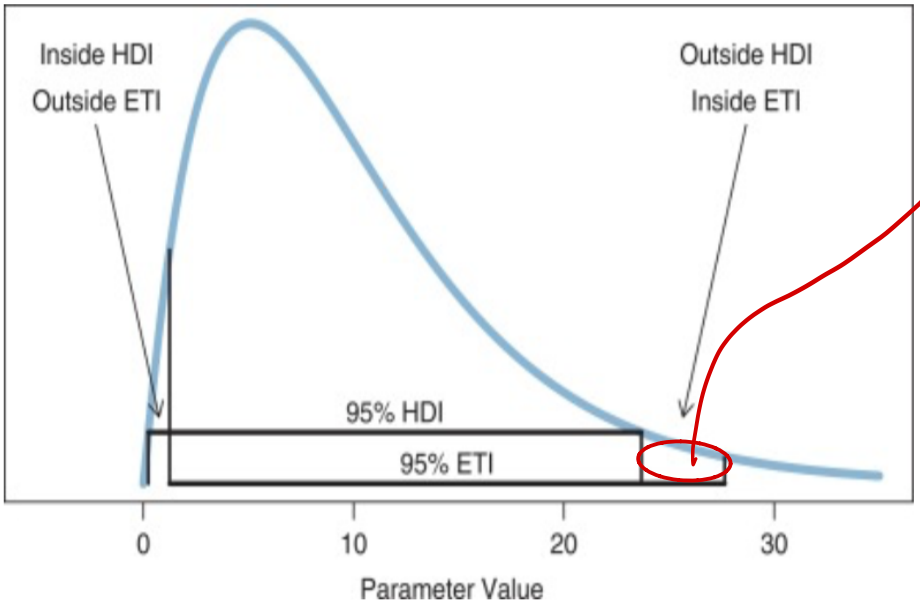
- Whilst these methods never accept null hypothesis (get it right), the precision rule get the sample distribution correct (equal to the true distribution).

Clearly precision is the better option of the 4. It is true that it takes quite a bit more data that HDI with ROPE, but hey, that's the inconvenient truth about noisy data → it can take a lot of data to cancel out noise.

Calculate Bayes with HDI, ROPE and Precision

What is HDI and why not equal-tailed intervals?

- HDI → high density interval
- The reason for using HDI is very intuitive: all values inside the HDI have higher probability density (or credibility) than any outside the HDI.
- Using equally tailed intervals (percentiles) is very easy to compute.
  - ↳ in symmetric distributions, HDI and ETI are the same.
  - ↳ but not when the distributions are skewed.



parameters in this region are not rejected even though they have a lower credibility.

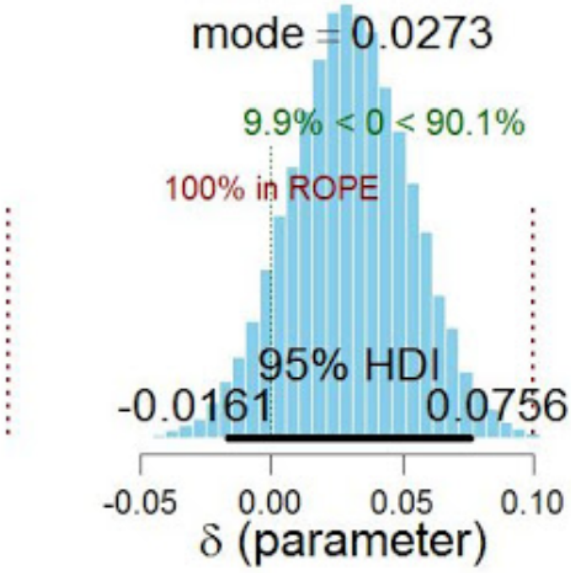
Calculate using PyMC3 → pymc3.stats.hpd([distribution to input]).

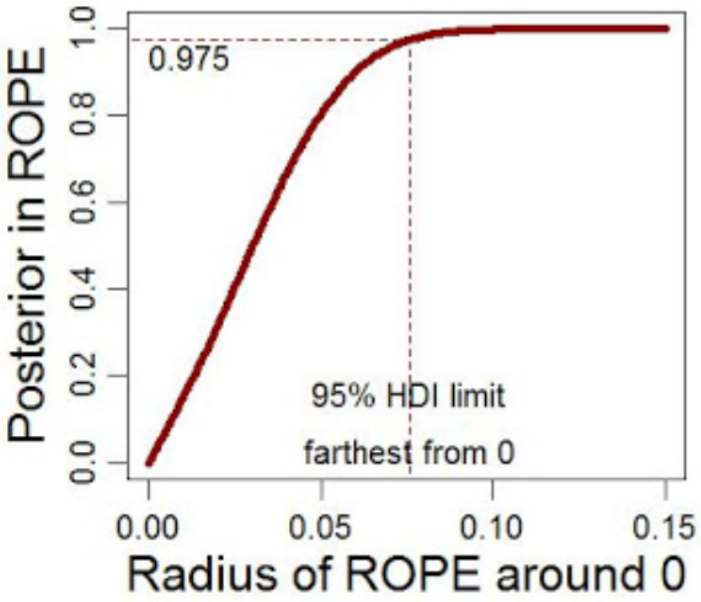
What is ROPE?

- ROPE → region of practical equivalence.
- This is region where we could conclude that the probability of being outside of a specific range is negligible.
  - ↳ using Bayesian, it doesn't make much sense to state that the probability of a distribution is different from 0 (P(distribution = single point) = 0).
  - ↳ The underlying idea of ROPE is to let the user define an area around the null value enclosing values that are equivalent to the null value.

Example of HDI in/out of ROPE

- We have seen that the posterior distribution of a parameter shows explicitly the relative credibility of the parameter values given the data
  - ↳ but generally, people prefer a Y/N answer rather than analysing a distribution.
  - ↳ for example the "null" values of 0.0 difference or 50% chance.





- ↳ suppose that we are running an experiment and we don't care if the differences seen move ±0.1.
- ↳ we can see that the HDI is completely contained within this region of practical equivalence.
- ↳ Therefore, we could make the decision to accept the null hypothesis (no difference is comparing results).

- ↳ The only problem is defining a sensible ROPE.
  - ↳ something that the user can define (sometimes an interval [-10%, 10%] is OK and sometimes it isn't)

Notes on HDI, ROPE and Precision

* Maybe difficult or abstract to define ROPE or the minimum detectable effect.

* It would be great if we could transform a recommendation closer to business goals

  ↳ For example, if we could calculate based on the volume of incoming data per day, how long would it take to reach a target, this could be easier for stakeholders and product owners to define.

  ↳ so far TIME isn't taken into account, while this is probably one of the most important factors in business.

    ↳ how long is it going to take to prove significant difference or practical equivalence ?

  ↳ In addition, specialists are much better at setting a target than defining a practical equivalence without any sense of time.

Idea:

- For control and variant, calculate testing time to reach a goal.
- Calculate whether variant is expected to be significant within that horizon.
- check an expectation if the test is going to be feasible.
  - ↳ If 50%+ of scenarios have an expected signifi it result, we should continue the test.
  - ↳ otherwise, stop the test.