Statistical Rethinking

Chapter 1: The Golem of Prague
Chapter 2: Small Worlds and Large Worlds
Chapter 3: Sampling the Imaginary

The Golems of Prague

Golem of Science

Golem

- Made of clay
- Animated by truth
- Powerful
- Blind to creator's intent
- Easy to misuse
- Fictional

Model

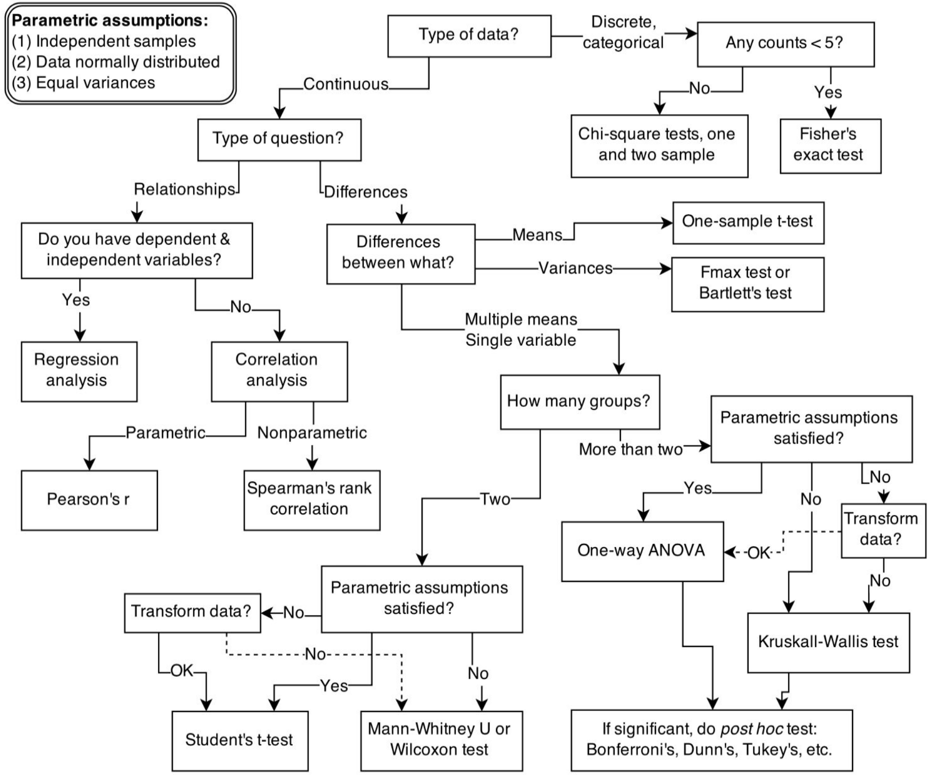
- Made of "silicon"
- Animated by "truth"
- Hopefully powerful
- Blind to creator's intent
- Easy to misuse
- Not even false

The statistical procedure is just an algorithm

Model is not about true or false

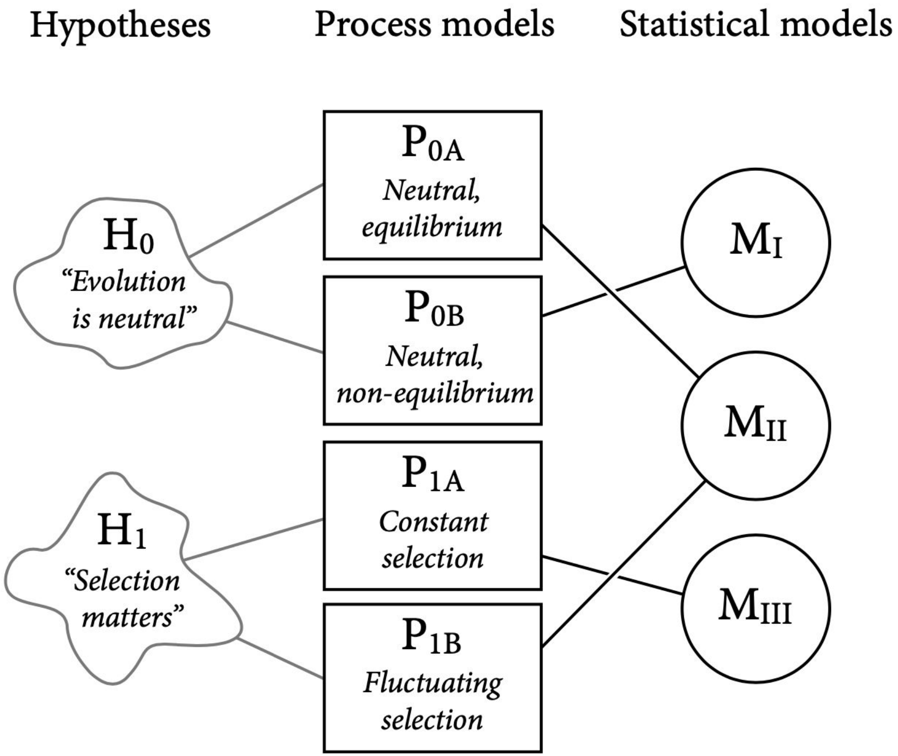
Model is a tool → useful or not

Frequentist Golem



- Pre-made golem
- Complicated procedure, difficult to deploy
- We need more freedom
- We need to **build our own golem**
- Falsifying *null* model is not sufficient
- They just produce “inference”, not decision

Hypotheses ≠ Models



- Many models correspond to the same hypothesis, and many hypotheses correspond to a single model
- Measurement matters
- All swans are white? The Ivory-billed Woodpecker is extinct? Faster-than-light neutrino?

Tools for Golem Engineering

Bayesian Data Analysis

Count all the ways data can happen, according to assumptions

- Use probability to describe uncertainty
- Computationally difficult
- Used to be controversial

Multilevel Modelling

Models with multiple levels of uncertainty

- Repeat & imbalanced sampling, study variation, avoid averaging
- Phylogenetics, factor and path analysis, networks, spatial models
- Natural Bayesian strategy

Model Comparison

Models with multiple levels of uncertainty

- Overfitting
- Causal inference
- Must distinguish prediction from inference

Small Worlds and Large Worlds

Small and Large Worlds

Small World

- All possibilities are nominated
- There are no pure surprises
- Bayesian models are optimal in small world

Large World

- There may be events that were not imagined in the small world
- No guarantee of optimality for Bayesian models

Garden of Forking Data

Contains 4 marbles

Possible contents:

(1)

(2)

(3)

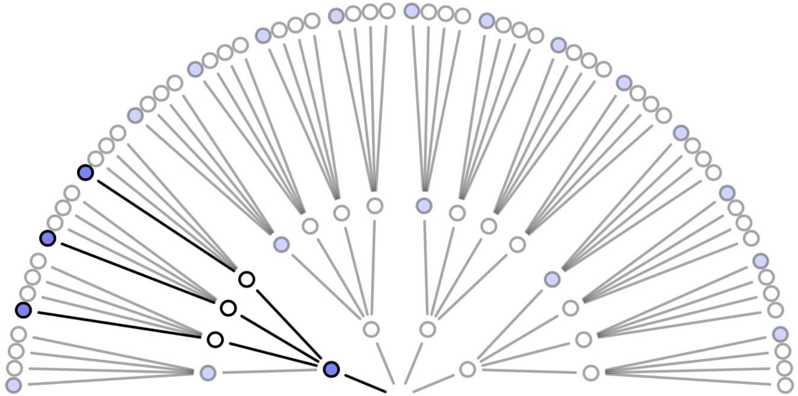
(4)

(5)

Observe:

Conjecture:

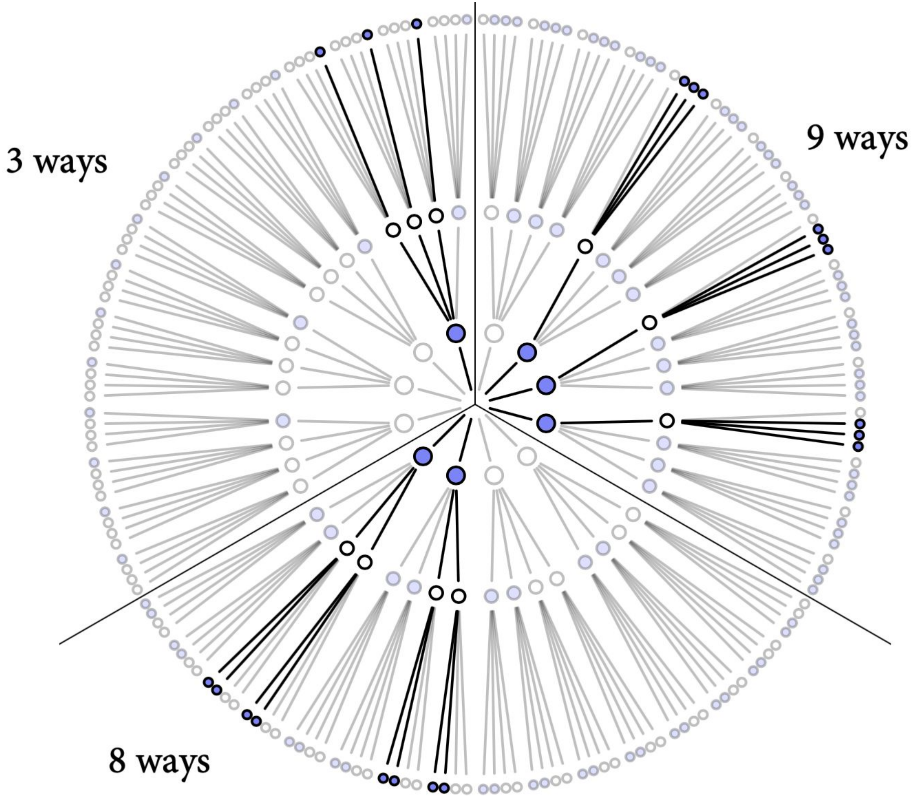
Data:



3 paths consistent with data

Garden of Forking Data

Conjecture	Ways to produce ●○●
[○○○○]	$0 \times 4 \times 0 = 0$
[●○○○]	$1 \times 3 \times 1 = 3$
[●●○○]	$2 \times 2 \times 2 = 8$
[●●●○]	$3 \times 1 \times 3 = 9$
[●●●●]	$4 \times 0 \times 4 = 0$



Updating

Another draw from the bag: ●

Conjecture	Ways to produce ●	Previous counts	New count
[○○○○]	0	0	$0 \times 0 = 0$
[●○○○]	1	3	$3 \times 1 = 3$
[●●○○]	2	8	$8 \times 2 = 16$
[●●●○]	3	9	$9 \times 3 = 27$
[●●●●]	4	0	$0 \times 4 = 0$

Using other information

Factory says: ● marbles rare, but every bag contains at least one.

Conjecture	Prior ways	Factory count	New count
[○○○○]	0	0	$0 \times 0 = 0$
[●○○○]	3	3	$3 \times 3 = 9$
[●●○○]	16	2	$16 \times 2 = 32$
[●●●○]	27	1	$27 \times 1 = 27$
[●●●●]	0	0	$0 \times 0 = 0$

Counts to plausibility

Things that can happen more ways are more plausible

Possible composition	p	ways to produce data	plausibility
[○○○○]	0	0	0
[●○○○]	0.25	3	0.15
[●●○○]	0.5	8	0.40
[●●●○]	0.75	9	0.45
[●●●●]	1	0	0

In:

```
import numpy as np

ways = np.array([0, 3, 8, 9, 0])
ways / ways.sum()
```

Out:

```
array([0.  , 0.15, 0.4 , 0.45, 0.  ])
```

Building a model

Design the model (data story) → Condition on the data (update) → Evaluate the model (critique)

Nine tosses of the globe:

W L W W W L W L W

Building a model

Design the model (data story) → Condition on the data (update) → Evaluate the model (critique)

- Data story motivates the model
  - How do the data arise?
- For W L W W W L W L W:
  - Some true proportion of water, *p*
  - Toss globe, probability *p* of observing W, *1-p* of L
  - Each toss therefore independent of other tosses
- Translate data story into probability statements

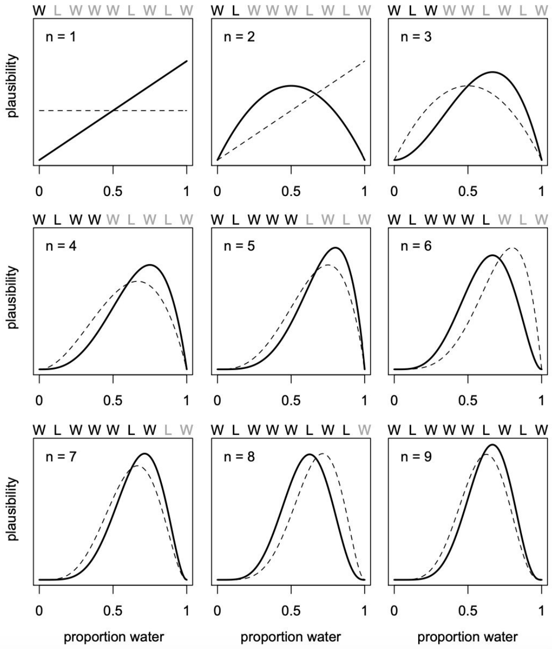
Building a model

Design the model (data story)

Condition on the data (update)

Evaluate the model (critique)

- Data order irrelevant
- Every posterior is a prior for next observation
- Every prior is posterior of some other inference
- Sample size automatically embodied in posterior



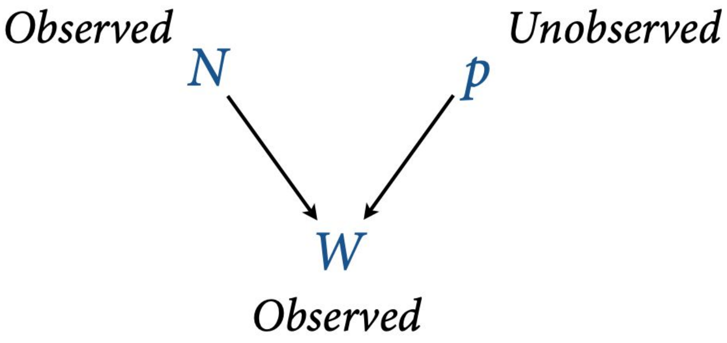
Building a model

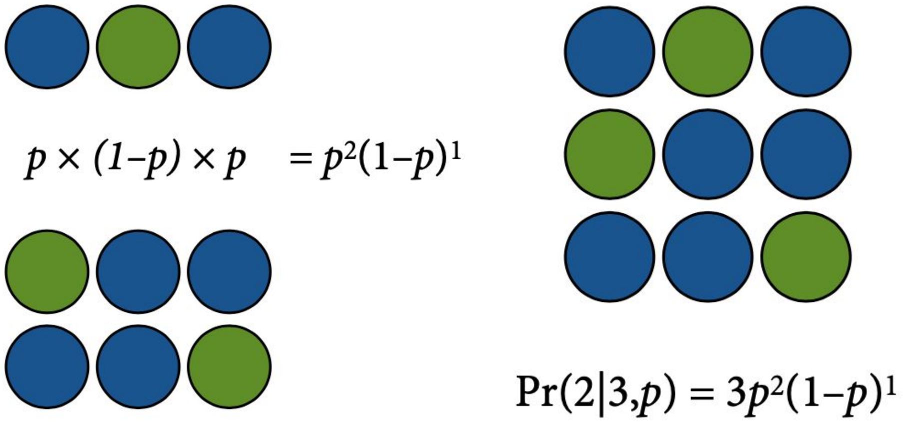
Design the model (data story) → Condition on the data (update) → **Evaluate the model (critique)**

- Bayesian inference: *How plausible is each proportion of water, given these data?*
- Golem must be supervised
  - Did the golem malfunction?
  - Does the golem's answer make sense?
  - Does the question make sense?
  - Check sensitivity of answer to changes in assumptions

Definition of *W*

- Relative number of ways to see *W*, given *N* and *p*?
- Probability distribution





Definition of *W*

- Relative number of ways to see *W*, given *N* and *p*?
- Probability distribution

$$\Pr(\underbrace{W}_{\text{count W}}|\overbrace{N}^{\text{number tosses}},\underbrace{p}_{\text{probability W}}) = \frac{N!}{W!(N-W)!}p^W(1-p)^{N-W}$$

The count of *W*'s is of binomial distribution, with probability *p* of a *W* on each toss and *N* tosses total.

In:

```
import scipy.stats as stats

stats.binom.pmf(6, n=9, p=0.5)
```

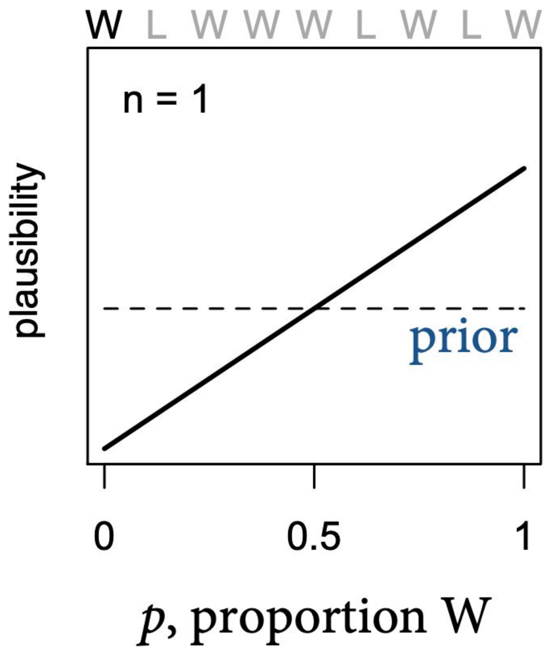
Out:

```
0.16406250000000006
```

Prior probability *p*

- What the golem believes before data arrive
- Pr(*W*) and Pr(*p*) define prior predictive function
- Huge literature on choice of prior
- Flat prior conventional & bad

$$W \sim \text{Binomial}(N, p)$$
$$p \sim \text{Uniform}(0, 1)$$

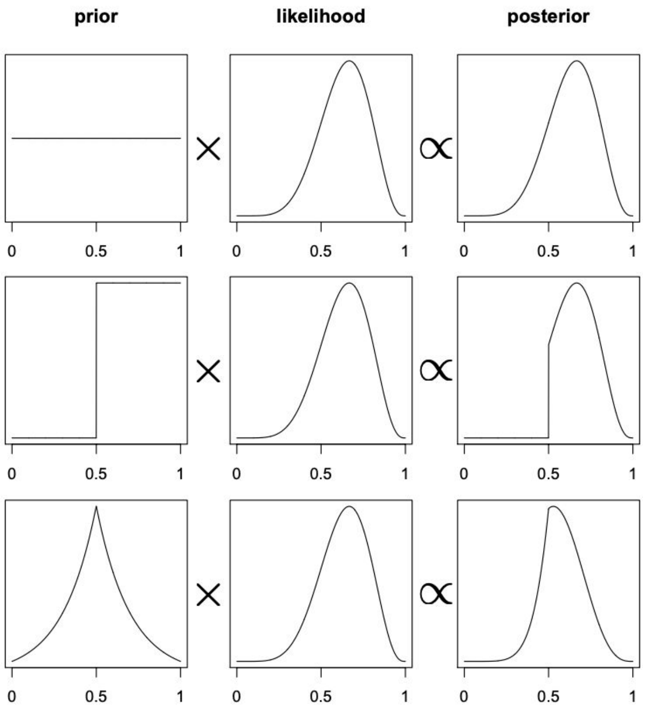


Posterior probability

- Bayesian estimate is always posterior distribution over parameters, Pr(parameters|data)
- Here: Pr(p|W, N)
- Bayes' theorem

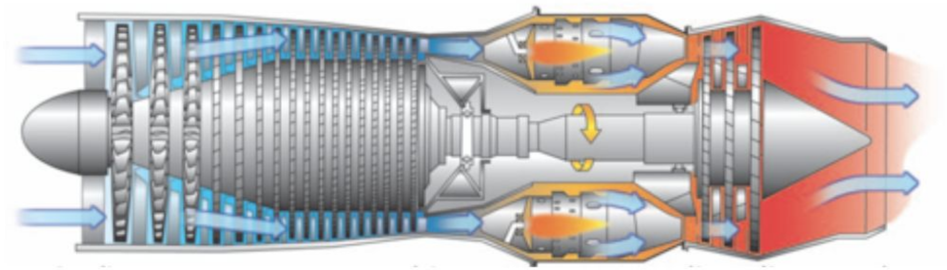
$$\Pr(p|W,N) = \frac{\Pr(W|N,p)\Pr(p)}{\sum \Pr(W|N,p)\Pr(p) \text{ for all } p}$$

$$\text{Posterior} = \frac{(\text{Prob observed variables}) \times (\text{Prior})}{\text{Normalizing constant}}$$



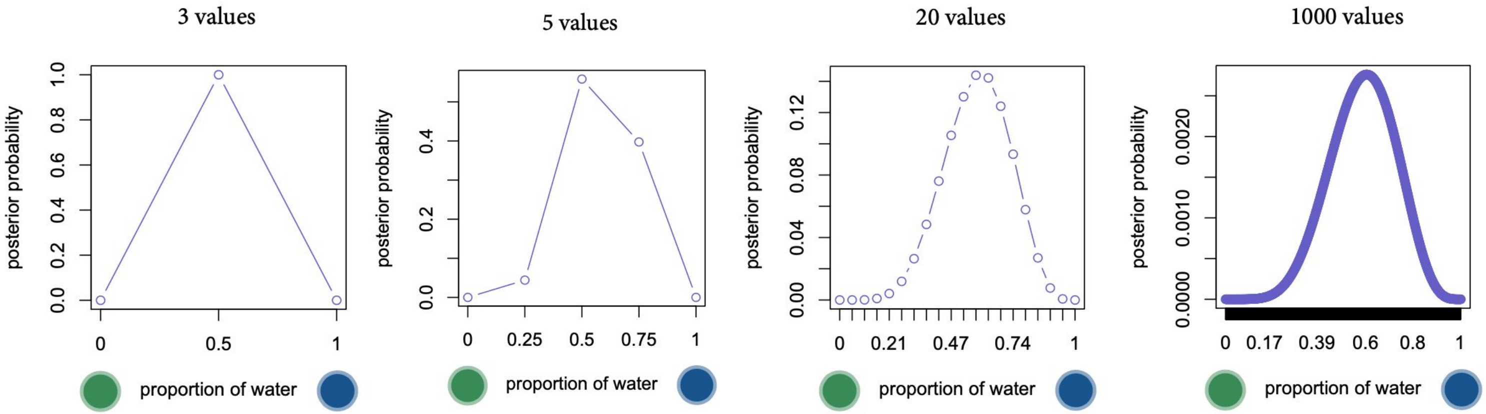
Computing the posterior

- Analytical approach (often impossible)
- Grid approximation (very intensive)
- Quadratic approximation (limited)
- Markov chain Monte Carlo (intensive)



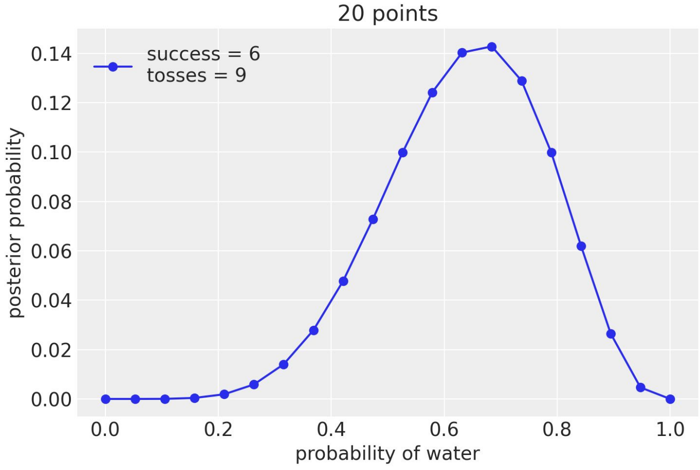
Grid approximation

- The posterior probability *standardized* product of probability of the data and prior probability
- *Standardized* means: Add up all the products and divide each by this sum
- Grid approximation uses finite grid of parameter values instead of continuous space
- Too expensive with more than a few parameters



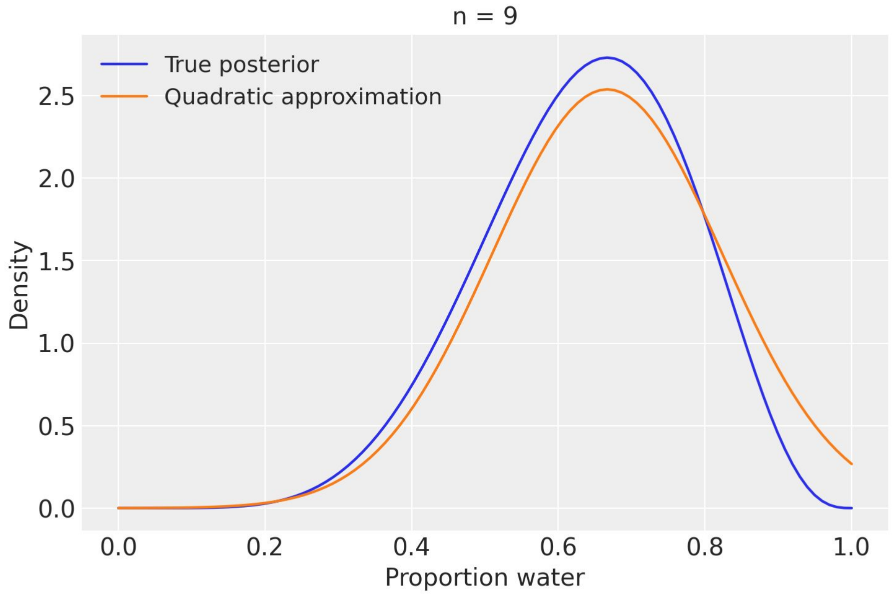
Grid approximation

```
import matplotlib.pyplot as plt

def posterior_grid_approx(grid_points=5, success=6, tosses=9):
    """
    """
    # define grid
    p_grid = np.linspace(0, 1, grid_points)

    # define prior
    prior = np.repeat(5, grid_points)  # uniform

    # compute likelihood at each point in the grid
    likelihood = stats.binom.pmf(success, tosses, p_grid)

    # compute product of likelihood and prior
    unstd_posterior = likelihood * prior

    # standardize the posterior, so it sums to 1
    posterior = unstd_posterior / unstd_posterior.sum()
    return p_grid, posterior

points = 20
w, n = 6, 9
p_grid, posterior = posterior_grid_approx(points, w, n)
plt.plot(p_grid, posterior, 'o-', label='success = {}\ntosses = {}'.format(w, n))
plt.xlabel('probability of water', fontsize=14)
plt.ylabel('posterior probability', fontsize=14)
plt.title('{} points'.format(points))
plt.legend(loc=0)
```



Quadratic approximation

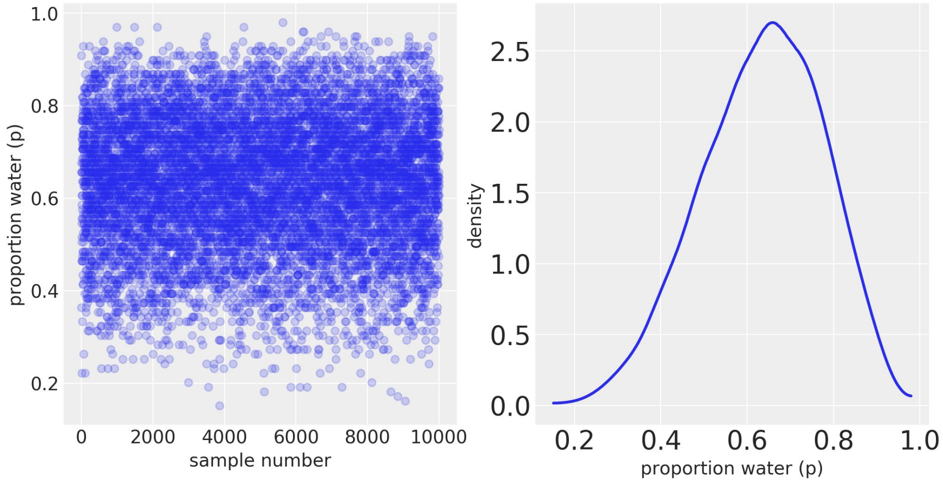
```
import pymc3 as pm

data = np.repeat((0, 1), (3, 6))
with pm.Model() as normal_aproximation:
    p = pm.Uniform('p', 0, 1)
    w = pm.Binomial('w', n=len(data), p=p, observed=data.sum())
    mean_q = pm.find_MAP()
    std_q = ((1/pm.find_hessian(mean_q, vars=[p]))**0.5)[0]
mean_q['p'], std_q

# analytical calculation
w, n = 6, 9
x = np.linspace(0, 1, 100)
plt.plot(x, stats.beta.pdf(x , w+1, n-w+1),
         label='True posterior')

# quadratic approximation
plt.plot(x, stats.norm.pdf(x, mean_q['p'], std_q),
         label='Quadratic approximation')
plt.legend(loc=0, fontsize=13)

plt.title('n = {}'.format(n), fontsize=14)
plt.xlabel('Proportion water', fontsize=14)
plt.ylabel('Density', fontsize=14)

plt.show()
```



Sampling the Imaginary

Sample from posterior

```
import arviz as az
%config InlineBackend.figure_format = 'retina'
az.style.use('arviz-darkgrid')

p_grid, posterior = posterior_grid_approx(grid_points=100,
                                          success=6,
                                          tosses=9)
samples = np.random.choice(p_grid,
                           p=posterior,
                           size=int(1e4),
                           replace=True)

_, (ax0, ax1) = plt.subplots(1,2, figsize=(12,6))
ax0.plot(samples, 'o', alpha=0.2)
ax0.set_xlabel('sample number', fontsize=14)
ax0.set_ylabel('proportion water (p)', fontsize=14)
az.plot_kde(samples, ax=ax1)
ax1.set_xlabel('proportion water (p)', fontsize=14)
ax1.set_ylabel('density', fontsize=14);
```



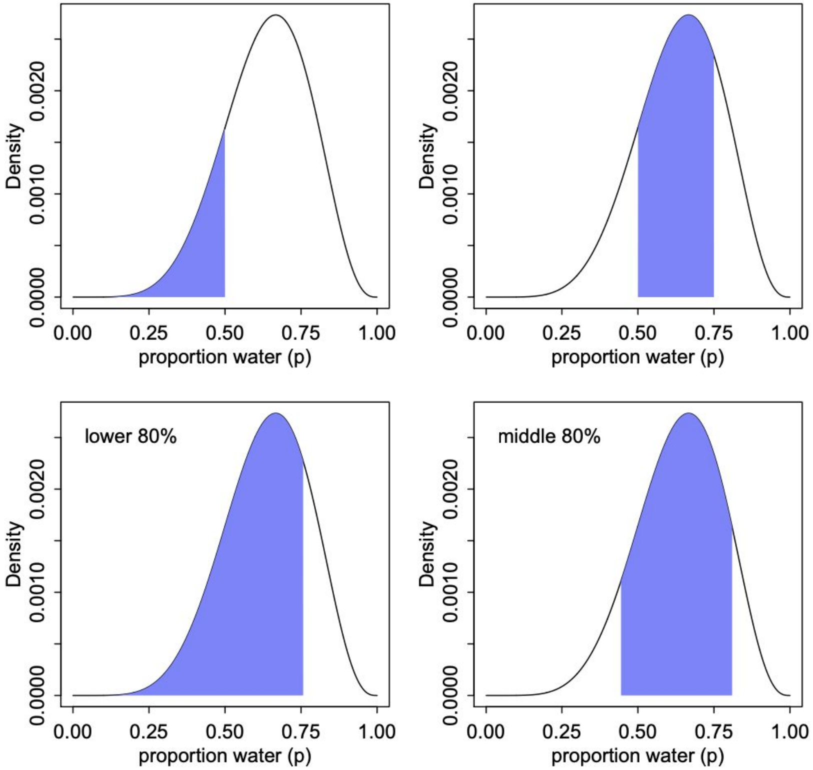
Compute stuff

- How much posterior probability below/above/between specified parameter values?
- Which parameter values contain 50%/80%/95% of posterior probability? “Confidence” intervals
- Which parameter value maximizes posterior probability? Minimizes posterior loss? Point estimates

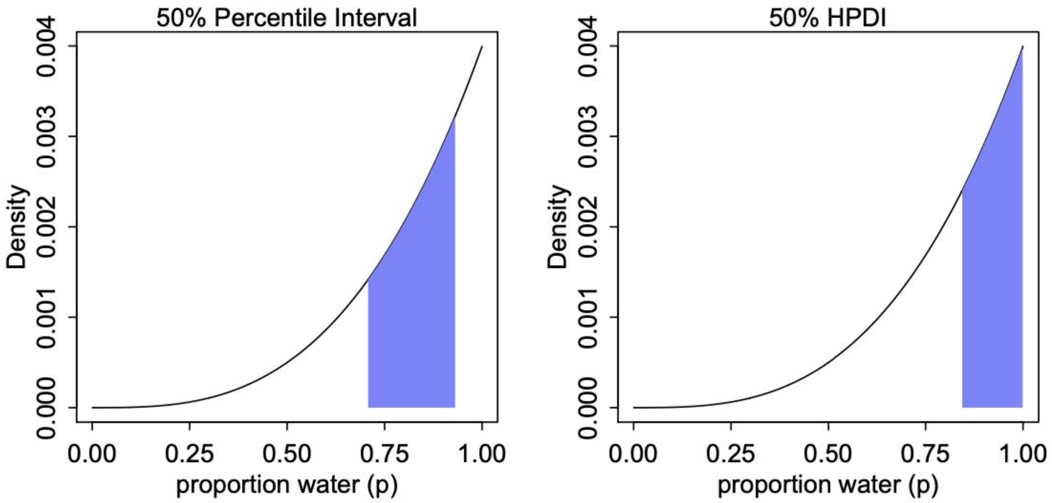
Compute stuff

Intervals of defined boundary ask how much mass?

Intervals of defined mass ask which values?



PI and HPDI



Percentile intervals (PI): equal area in each tail

Highest posterior density intervals (HPDI): narrowest interval containing mass

Point estimates not the point

- Don’t usually want point estimates
    - Entire posterior contains more information
    - “Best” point depends upon purpose
    - Mean nearly always more sensible than mode

E(posterior) arg max(posterior)

In:

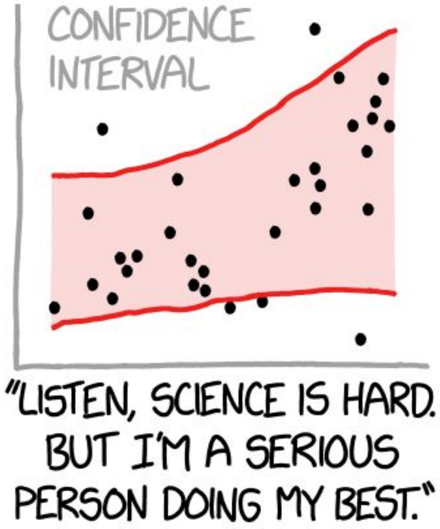
```
print(stats.mode(samples))
print(np.mean(samples))
```

Out:

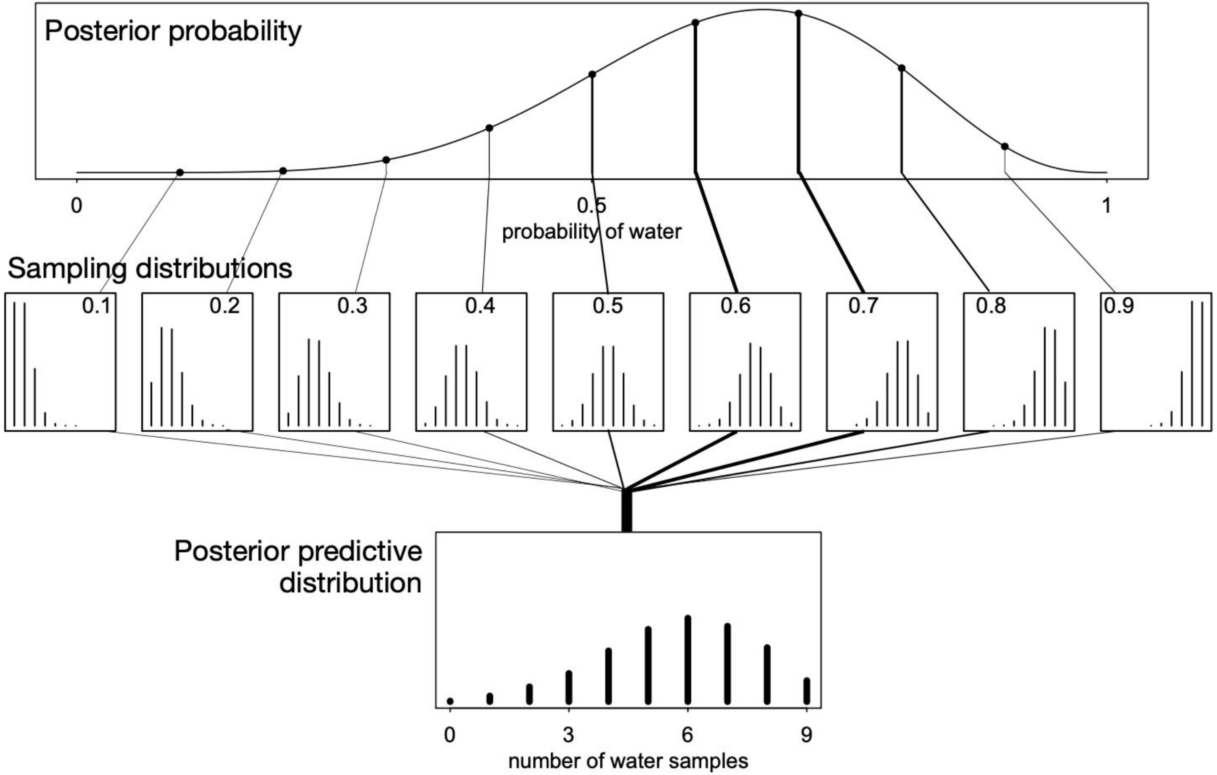
```
ModeResult(mode=array([0.66666667]), count=array([292]))
0.6348787878787879
```

Talking about intervals

- Confidence interval?
  - A non-Bayesian term that doesn’t even mean what it says
- Credible interval?
  - The values are not “credible” unless you trust the model & data
- How about: Compatibility interval?
  - Interval contains values compatible with model and data as provided
  - Small World interval



Posterior predictive distribution



Posterior predictive distribution

```
dummy_w = stats.binom.rvs(n=9, p=samples)
plt.hist(dummy_w, bins=50)
plt.xlabel('number of water samples', fontsize=14)
plt.ylabel('Frequency', fontsize=14)
```