

Statistical Rethinking: Lecture 02

with Python Codes

Tossing the globe example

The problem: to estimate the proportion of water covering the globe by “tossing it”, i.e. sampling water and land observations. Imagine being away from the planet and being able only to send probes to get the information of whether it landed on land “L” or water “W”. Alternatively, imagine a globe replica being tossed at a crowd and taking note of where in the globe, water or land, it first touched the hand of a person in the crowd.

In frequentist statistics, we could use the relative frequencies of “W” or “L” as the estimate for water and land, respectively. Here we deal with the Bayesian approach: given a sample consisting on W and L observations of water and land, what value of p , the proportion of water covering the globe, is more likely to reproduce the sample?

Since p can range from 0 to 1, let us consider a 4-sided “globe”, a tetrahedron or 4-sided die. This allows discretizing the problem. At the end we can let the limit of sides reach infinity, recovering a proper globe.

Recall the fundamental principle in Bayesian analysis

For each possible explanation of the sample, count the ways each explanation can reproduce the sample. Explanations with the largest number of ways to reproduce the data are more likely.

And the basic workflow

- 1) Define a generative model of the sample (generate synthetic data?)
- 2) Define a specific estimand
- 3) Design a way to produce the estimate (estimator?)
- 4) Test 3) using 1) (use the estimator to get an estimate in controlled situation, where you know the answer)
- 5) Analyze sample (real data)

1) The generative model

Let us define a function to generate the sampling of water “W” or land “L” observations. If p is the proportion of water covering the globe, then the probability of observing “W” is p and “L” is $1 - p$.

```
def toss_globe(proportion, N):  
    return np.random.choice(["W", "L"], size=N, p=[proportion, 1 - proportion])
```

We can toss a globe covered 70% by water ten times by calling `toss_globe(p=0.7, N=10)`, which gives

```
array(['W', 'L', 'L', 'W', 'L', 'W', 'W', 'W', 'W', 'W'], dtype='<U1')
```

2) The estimand

Our estimand will be the number of ways each explanation can reproduce the sample. In this case, the number of ways each proportion of water p covering the globe can reproduce a series of observations. That value of p with the largest number of ways to reproduce the data is more plausible to be the true value. Equivalently, instead of counting, we can normalize by the total number of ways of explaining the data and work with a probability. In this manner, our estimand will be the probability distribution for the proportion of water p .

3) The estimator

The probability of observing an array of W water and L land observations is

$$\Pr(W, L) = p^W (1 - p)^L,$$

since each observation is independent. For a total of $N = W + L$ tosses, the expected number of water and land observations are given by the above equation multiplied by 4^N , or $4^L \times 4^W$:

$$W \quad \text{or} \quad L = (4p)^W (4 - 4p)^L$$

which count the ways each explanation – the proportion of water p – can reproduce the sample. This is our estimator.

Using the sample observed in the previous toss, we can calculate the number of ways each proportion (explanation) can reproduce the sample as follows

```

W, L = np.sum(sample=='W'), np.sum(sample=='L')
proportions = [0, 0.25, 0.5, 0.75, 1]
ways = [(4 * p)**W * (4 - 4 * p)**L for p in proportions]
ways

```

```
[0, 27.0, 1024.0, 2187.0, 0]
```

Normalizing by the total number of ways we can calculate the probabilities each proportion p has to reproduce the data. The plot is shown in [Figure 1](#).

```

probabilities = np.array(ways)/np.sum(ways)
probabilities

```

```
array([0.          , 0.00833848, 0.3162446 , 0.67541692, 0.          ])
```

```

import matplotlib.pyplot as plt
fig, ax = plt.subplots()
ax.bar([str(prop) for prop in proportions], probabilities)
ax.set_xlabel('water proportions')
ax.set_ylabel('proportion probability')

plt.show()

```

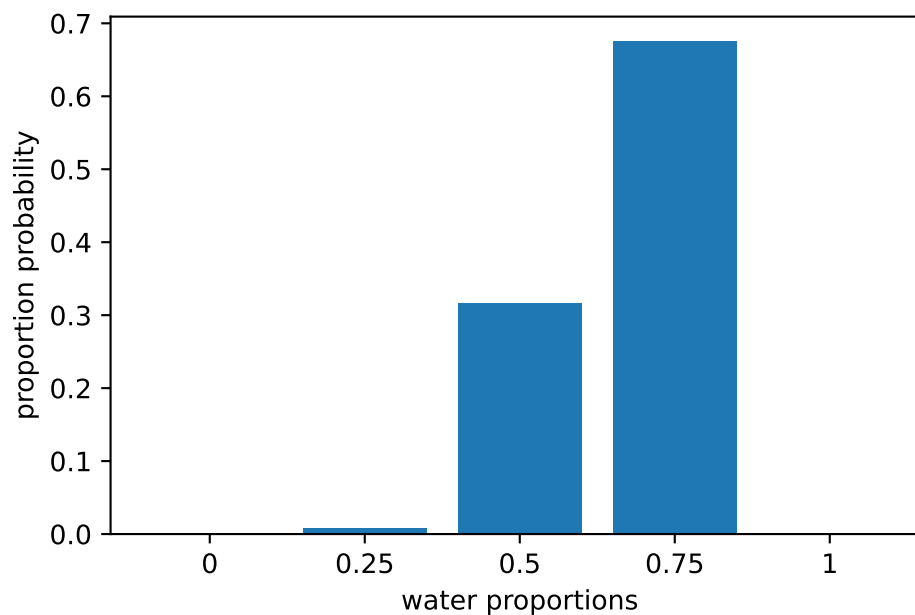


Figure 1: Bar plot for the plausability of each proportion of

4) Testing

We have thus far succesfully designed the generative model, defined the estimand and designed the statistical estimator. We have also anticipated one estimate by applying the estimator to the first sample we got from the generative model. Now we dive deeper in testing if the estimand and estimator are valid.

Let us simulate extreme cases.

```
sample = toss_globe(proportion=1, N=10)
sample
```

```
array(['W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W'], dtype='<U1')
```