Notes on Probability, Simulation, and Sampling

Christopher Brown

November 9, 2018

Introduction

Probability

Terminology

An *event space* or *sample space* S is a nonempty set. An *event* is a subset of the given sample space. Visually, we can represent events in sample spaces as .

Probability is a complicated term. In these notes we give two rough definitions of probability, each of which is useful in some scenarios. For a more precise mathematical definition of probability see [].

Frequentist definition: Roughly, we determine the *probability of an event* A by repeatedly performing an experiment in which the event A may or may not occur, and the probability is then the ratio of the number of times A occurs to the number of times the experiment is performed. That is, the probability of A is the relative frequency at which A occurs in an experiment repeated many times. For example, if we flip a fair coin 1000 times, we expect that the relative frequency with which we see heads will be about $1/2$.

Bayesian definition: Roughly, we determine the *probability of an event* A by evaluating the credibility we have in the occurrence of A given our observations of the system. For example, if we have not flipped a coin, we might give equal credibility to Heads and Tails for the next flip. We base this on the fact that most coins in our experience seem to be fiar. On the other hand, if we have already flipped that coin 10 times and seen 10 heads, we might give more credibility to Heads than Tails for the next flip, because we suspect the coin may be biased.

Notation

Notation in probability theory is notoriously complex, informal, and nonstandardized. In these notes we'll try to be consistent and explain what we mean by our notation. There will still be times when informality is helpful.

We usually denote events by capital letters like A, B, and so on. We denote the **probability of event** A by $P(A)$.

When A and B are both events, the event "A and B" is the set $A \cap B$. We write $P(A \text{ and } B) = P(A \cap B)$. Many textbooks also write $P(A, B)$ for the probability of A and B; we won't use this notation but we mention it for the sake of connecting with other works.

Two events are *mutually exclusive* when $A \cap B = \emptyset$.

The event "A or B" is the set $A \cup B$ and we write $P(A \text{ or } B) = P(A \cup B)$.

The event "not A" is called the *complement of* A, and is written A^C. If we need to specify the sample space, we may use the set difference notation $S \setminus A$ instead of A^C.

Properties of Probability

Chapters 2 and 3 of [1] contain a standard presentation of the properties of probability. Chapter 1 of [2] contains a more advanced look at this material. Here, we recount only the axioms and the basic propositions

at a basic level.

Suppose that S is a given sample space. Then a probability measure on S is a function P mapping events to real numbers, with P satisfying the following three axioms:

Axiom 1: For every event A, $0 \leq P(A) \leq 1$.

Axiom 2: $P(S) = 1$.

Axiom 3: For any sequence of mutually exclusive events A_1, A_2, ... A_n,...,

$$P\left(\bigcup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} P(A_i)$$

From these three axioms, we can then prove the following propositions:

Proposition (Complementation Rule): $P(A^C) = 1 - P(A)$

Proposition (P is Increasing): If $A \subseteq B$ then $P(A) \leq P(B)$.

Proposition (Sum Rule): For all events A and B,

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

Conditional Probability

The notion of conditional probability is probably the most important concept in probability theory. In essence, we compute probabilities of events given different collections of information.

If A and B are events, then the *probability of A given that B occurred* is denoted $P(A|B)$. When we know that B occurred, this reduces our sample space to B. A few sketches with Venn diagrams should convince you that

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

This expression can be rearranged to read, equivalently, $P(A \cap B) = P(A|B)P(B)$. Section 3.5 of [1] shows that the function $P(\cdot|B)$ mapping events to real numbers is also a probability measure, in that it satisfies the three axioms above.

Because $P(A \cap B) = P(B \cap A)$, we can now write

Proposition (Bayes' Rule):

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Proposition (Conditioning on a Partition): If B_1, B_2,...,B_n is a collection of mutually exclusive events whose union is S (a *partition* of S), then

$$P(A) = \sum_{i=1}^{n} P(A|B_i)P(B_i)$$

Two events A and B are said to be *independent* if $P(A|B) = P(A)$. That is, A and B are independent if the computation of the probability of A does not change whether we know B happened or not.

Proposition: A and B are independent events if and only if $P(A \cap B) = P(A)P(B)$.

Random Variables

Exercises

Simulation

Random Number Generators in R

Aside: The Differences Among For Loops, While Loops, and Apply

When deciding which programming structure to use to repeat an action ("loop"), we need to ask three questions: 1. Before beginning the loop, do we know how many times we need to run the loop? 2. Will any of the repetitions of the loop influence the computation in the *next* repetition of the loop? 3. Will any of the repetitions of the loop influence the computation in the *previous* repetition of the loop?

For us, an answer of *yes* to 3 takes us a little too deeply into a more advanced computer programming topic (recursively defined functions). Also, there are almost no situations we will run into that will give an answer of *yes* to 3. So, let's just assume we don't have to worry about 3! Now on to the more practical concerns...

If the answer to 1 is *no*, we will almost surely need a while loop.

If the answer to 1 is *yes* and the answer to 2 is *yes*, we will almost surely need a for loop.

If the answer to 1 is *yes* and the answer to 2 is *no*, we will be able to use an apply loop. This will happen very frequently for us; we'll see why in the examples!

Modularity and Simulation

In computer programming, *modularity* is the programming practice in which programs are broken down into small chunks by using functions and comments, in such a way that the chunks can be individually assessed for performance and the code can be more easily understood.

Example: Dice Throw Probability

Let's first simulate a probability we can compute symbolically, as a check: what is the probability of rolling a sum of 7 when rolling a pair of fair six-sided dice?

Generally, we can fit probability simulations into a common framework:

- Define constants for the experiment.
- Create one or more functions to perform one repetition of an experiment (in this case, one roll of the two dice).
- Create a function or code to perform many experiments and record the results of each experiment.
- Analyze the results of the collection of experiments.

Let's begin by defining constants for the experiment.

```
number_of_experiments <- 100000
one_die <- 1:6
```

Next, let's define a function to roll two fair six-sided dice and sum the faces showing.

```
one_roll <- function() {
  return(sum(sample(one_die,2,replace=TRUE)))
}
```

Before we go on to write the rest of our code, let's test this function.

```
print(one_roll())
```

```
## [1] 8
```

And again:

```
print(one_roll())
```

```
## [1] 5
```

And you can execute this a few more times to see how it behaves.

```
sapply(1:10,function(z) one_roll())
```

```
##  [1] 11  8  6 12  7  9  9  4  8  6
```

This gives us a good idea of how we should proceed for our next phase: repeat the experiment many times and record the data.

```
data <- sapply(1:number_of_experiments,
               function(z) one_roll())
```

At this point we need to compute the number of 7's we see in the data.

```
number_of_7 <- length(which(data==7))
```

And now we can compute the probability that we roll a 7.

```
number_of_7/number_of_experiments
```

```
## [1] 0.16663
```

Symbolically, we have six 7's out of 36 outcomes, i.e.

```
6/36
```

```
## [1] 0.1666667
```

So our estimate is certainly good enough for government work. If we need to how can we improve our estimate?

Longer Example: Professor Z's Office Hours

Professor Z tries to show up at his office for appointments on time. He really does! But students and faculty frequently stop him to ask questions or just chat, and he is a typical absent-minded professor, so sometimes he is late.

Estrella is trying to meet Professor Z at his office for an appointment. She really is! But between traffic and campus parking, Estrella might be late.

Professor Z and Estrella are both busy, so they can't wait long for the other person.

(a) Let's assume that Professor Z and Estrella arrive at the appointment at some time uniformly distributed between noon (time 0) and 1:00PM (time 1), and let's assume each will wait for the other for 15 minutes but not longer. Write a simulation to estimate the probability that Professor Z and Estrella actually meet for their appointment.

(b) Now let's assume that Professor Z and Estrella each arrive at the appointment at a time uniformly distributed between noon and 1PM. Let's also assume that each will wait for a uniformly distributed random amount of time between 0 and 20 minutes. What do you expect to happen to the probability the two meet in this scenario, as compared to that in part (a)? Write a simulation to estimate the probability that Professor Z and Estrella actually meet for their appointment.

Part (a)

Define our constants.

```
number_of_experiments <- 100000
wait_time <- 15
```

Define functions to perform one repetition of this experiment.

```
one_rep <- function() {
  arrival_time <- runif(2,min=0,max=60)
  abs(arrival_time[1]-arrival_time[2])<=15
}
```

How can we test the one_rep function?

Now repeat the experiment many times and record the data.

```
data <- sapply(1:number_of_experiments,
       function(z) one_rep())
```

Finally we can analyze the data and compute the probability of interest.

```
length(which(data))/number_of_experiments
```

```
## [1] 0.43723
```

Part (b)

Define our constants.

```
number_of_experiments <- 100000
```

Define functions to perform one repetition of this experiment.

```
one_rep <- function() {
  arrival_time <- runif(2,min=0,max=60)
  wait_time <- runif(1,min=0,max=20)
  abs(arrival_time[1]-arrival_time[2])<=wait_time
}
```

How can we test the one_rep function?

Now repeat the experiment many times and record the data.

```
data <- sapply(1:number_of_experiments,
       function(z) one_rep())
```

Finally we can analyze the data and compute the probability of interest.

```
length(which(data))/number_of_experiments
```

```
## [1] 0.29712
```

Does this probability agree with your expectation?

Exercises

Sampling

Distributions in R

The Monte Carlo Method

Inversion Sampling

Acceptance-Rejection Sampling

Exercises

References

[1] Ross. **A First Course in Probability: Fifth Edition**. Prentice-Hall, 1998.

[2] Grimmett, Stirzaker. **Probability and Random Processes: Third Edition**. Oxford University Press, 2001.

[] Shonkwiler. **Explorations in the Monte Carlo Method**.

[] Rubinstein and Kroese. **Simulation and the Monte Carlo Method: Third Edition**. Wiley, 2017.