BEE 4750 Lab 2: Uncertainty and Monte Carlo

2024-08-05

> Due Date
>
> Wednesday, 9/25/24, 9:00pm

Setup

The following code should go at the top of most Julia scripts; it will load the local package environment and install any needed packages. You will see this often and shouldn't need to touch it.

```
import Pkg
Pkg.activate(".")
Pkg.instantiate()
```

```
using Random # random number generation
using Distributions # probability distributions and interface
using Statistics # basic statistical functions, including mean
using Plots # plotting
```

Overview

In this lab, we will use Monte Carlo analysis to analyze a version of the culmination of every episode of the long-running game show The Price Is Right: the Showcase.

Monte Carlo methods involve the simulation of random numbers from probability distributions. In an environmental context, we often propagate these random numbers through some more complicated model and then compute a resulting statistic which is relevant for assessing performance or risk, such as an average outcome or a particular quantile.

You should always start any computing with random numbers by setting a "seed," which controls the sequence of numbers which are generated (since these are not *really* random, just "pseudorandom"). In Julia, we do this with the `Random.seed!()` function.

```
Random.seed!(1)
```

```
TaskLocalRNG()
```

It doesn't matter what seed you set, though different seeds might result in slightly different values. But setting a seed means every time your notebook is run, the answer will be the same.

> ⚠ Seeds and Reproducing Solutions
>
> If you don't re-run your code in the same order or if you re-run the same cell repeatedly, you will not get the same solution. If you're working on a specific problem, you might want to re-use `Random.seed()` near any block of code you want to re-evaluate repeatedly.

Probability Distributions and Julia

Julia provides a common interface for probability distributions with the `Distributions.jl` package. The basic workflow for sampling from a distribution is:

1. Set up the distribution. The specific syntax depends on the distribution and what parameters are required, but the general call is the similar. For a normal distribution or a uniform distribution, the syntax is

   ```
   # you don't have to name this "normal_distribution"
   #   is the mean and   is the standard deviation
   normal_distribution = Normal( ,  )
   # a is the upper bound and b is the lower bound; these can be set to
    ↪  +Inf or -Inf for an unbounded distribution in one or both
    ↪  directions.
   uniform_distribution = Uniform(a, b)
   ```

   There are lots of both univariate and multivariate distributions, as well as the ability to create your own, but we won't do anything too exotic here.

2. Draw samples. This uses the `rand()` command (which, when used without a distribution, just samples uniformly from the interval $[0, 1]$.) For example, to sample from our normal distribution above:

```
# draw n samples
rand(normal_distribution, n)
```

Putting this together, let's say that we wanted to simulate 100 six-sided dice rolls. We could use a Discrete Uniform distribution.

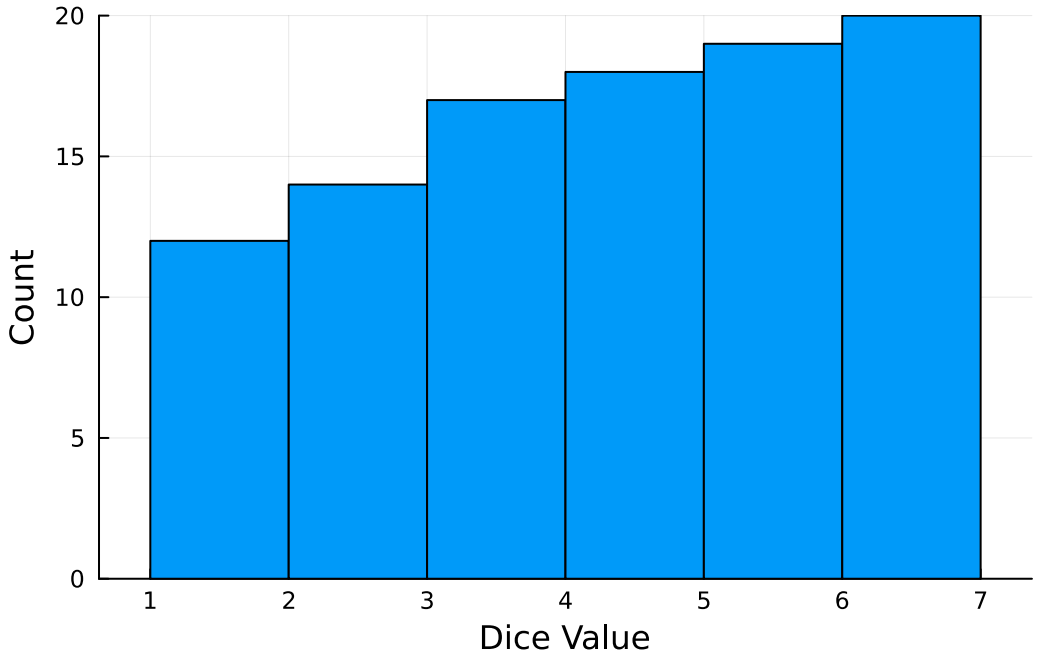
```
dice_dist = DiscreteUniform(1, 6) # can generate any integer between 1 and 6
dice_rolls = rand(dice_dist, 100) # simulate rolls
```

```
100-element Vector{Int64}:
 3
 5
 4
 5
 5
 2
 3
 4
 1
 6

 4
 2
 2
 5
 4
 2
 1
 2
 6
```

And then we can plot a histogram of these rolls:

```
histogram(dice_rolls, legend=:false, bins=6)
ylabel!("Count")
xlabel!("Dice Value")
```



Instructions

Remember to:

- Evaluate all of your code cells, in order (using a `Run All` command). This will make sure all output is visible and that the code cells were evaluated in the correct order.
- Tag each of the problems when you submit to Gradescope; a 10% penalty will be deducted if this is not done.

Exercise (3 Points)

The Showcase is the final round of every episode of The Price is Right, matching the two big winners from the episode. Each contestant is shown a "showcase" of prizes, which are usually some combination of a trip, a motor vehicle, some furniture, and maybe some other stuff. They then each have to make a bid on the retail price of the showcase. The rules are:

- an overbid is an automatic loss;
- the contest who gets closest to the retail price wins their showcase;
- if a contestant gets within $250 of the retail price and is closer than their opponent, they win both showcases.

Your goal is to find a wager which maximizes your expected winnings, which we may as well call utility, based on your assessment of the probability of your showcase retail price. Make the following assumptions about your expected winnings if you don't overbid:

- If you win both showcases, the value is the double of the single showcase value.
- If you did not win both showcases but bid under the showcase value, the probability of being outbid increases linearly as the distance between your bid and the value increases (in other words, if you bid the exact value, you win with probability 1, and if you bid \$0, you win with probability 0).

We'll assume that the distribution of all showcases offered by the show is given as truncated normal distribution, which means a normal distribution which has an upper and/or lower bound. `Distributions.jl` makes it easy to specify truncations on any distribution, not just normal distributions. For example, we'll use the distribution shown in Figure 1 for the showcase values (you can just directly use the `showcase_dist` distribution in your lab code).

```
showcase_dist = truncated(Normal(31000, 4500), lower=5000, upper=42000)
```

```
Truncated(Distributions.Normal{Float64}( =31000.0,  =4500.0); lower=5000.0, upper=42000.0)
```

Figure 1: Distribution of Showcase values for this lab, given as a truncated normal distribution.

Find your expected winnings if you bid \$35,000. Plot the outcomes of your Monte Carlo experiment (iterations vs. running estimate). How did you decide how many samples to use?

References

Put any consulted sources here, including classmates you worked with/who helped you.