concordance=TRUE Ec 5023 Prof. Le Wang
Statistics for Decision Making

Homework #6

November 28, 2018

Instruction: Do all the following empirical exercises using R. Turn in your answer with tables and graphs, if any, (along with your program and output files appended at the end of document). Refer to the output file whenever appropriate when discussing your results. It does not matter which method you use to plot a time series variable.

Note that for all simulation exercises, set the seed number to 123456 to ensure the reproducibility of your results.

1. Question 1.[Monte Carlo Simulation]

In class we discussed how to use Monte Carlo simulation to approximate integration. That example is based on the standard uniform distribution because the support is $[0, 1]$. You can extend this to obtain integration over a suppose different from $[0, 1]$. For example,

$$\begin{aligned} \theta &= \int_1^3 (x^2 + x)dx \\ &= 2 \cdot \int_1^3 (x^2 + x)\frac{1}{2}dx \\ &= 2\mathbb{E}[X^2 + X] \end{aligned}$$

Note that X is distributed from $U(1, 3)$ (Uniform distribution with support $[1, 3]$) whose density is $\frac{1}{3-2} = \frac{1}{2}$. The key is to simulate X that is distributed from such distribution. The idea is to stretch and shift the standard uniform distribution. Let me guide you through this process step by step.

1. Generate 10000 observations drawn from the standard uniform distribution. Denote this by U.
2. Generate a new variable based on the random variable in (1). **Hint:** Stretch is done by multiplication, and shift is done by addition. Convince yourself these operations actually give you a uniform variable with support $[1, 3]$. Please try to figure out how to do this manually instead of using the built-in R command.

   More Hints: Basically, you'd like to come up with a new variable, x, as follows

   $$x = a + b \cdot U$$

   Your goal is to come up with numbers a and b that would make x follow the uniform distribution $[1, 3]$. To convince yourself of your answer, note that both 0 and 1 are the end points for the standard uniform distribution; after transformation, these two points should become 1 and 3, respectively.
3. Generate a function, $g(x)$ of these observations: $gx = x^2 + x$.
4. calculate the mean of $g(x)$
5. Calculate 2 multiplies the mean of $g(x)$. For comparison, note that the exact answer to the original integration problem is 12.67.

I know it is probably a straightforward question that you'd like to have. And it may be a bit frustrating and time-consuming to figure out such a small problem... Then, I was watching Kongfu Panda III..

2. Question 2.[Understand the Use of the (standard) Uniform Distribution]

In class we showed how we can construct a set of random values drawn from a normal distribution using an approach called **inverse transform method**.

Prior to continuing, lets use the typical way that we discussed in class. It is easy. We simply use the command prefixing a "p" to the root function. Suppose that we would like to generate a random sample of 10000 observations drawn from the standard normal distribution. We will do the following

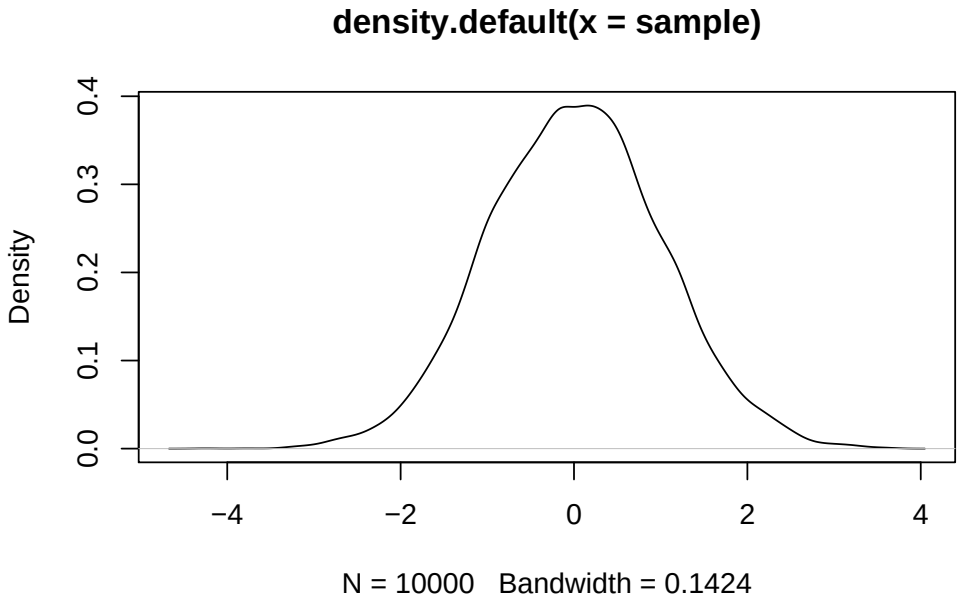
```
set.seed(123456)
sample<-rnorm(10000)
```

Let's make sure that we indeed have the "right" sample.

```
mean(sample)

## [1] 0.01683967

sd(sample)

## [1] 0.9984414
```

The mean and the standard deviation of the random sample are indeed close to zero and one as intended. We can also examine the density plot

```
plot(density(sample))
```



The density plot looks like a bell-shape, symmetric distribution that resembles a normal distribution. Now, let's use the inverse transform method, which proceeds with two steps.

Note: Even if you don't know understand the theory behind this exercise, you should not have any difficulty fininishing it.

1. Generate a sample of 10000 random values drawn from the **standard uniform distribution**.
2. Use the inverse CDF function (i.e., quantile function) for the standard normal distribution to obtain 10000 corresponding values.
3. Examine the data you produce from (2). Are they normally distributed? **Hint:** Please see my example above for how to visually determine whether a random variable follows from a normal distribution. It is only a rough approach, not a statistical one; the latter will be discussed later.
4. Let's work on another example.
   (a) Generate a sample of 10000 random values drawn from the **standard uniform distribution**.
   (b) Use the inverse CDF function (i.e., quantile function) for the normal distribution (with mean 5 and standard deviation 3) to obtain 10000 corresponding values.
   (c) Examine the data you produce from (2). Are they normally distributed with mean 5 and standad deviation?

3. Question 3.[Visulize the Central Limit Theorem]

In class we visualized the Central Limit theorem when the underlying distribution is a "fair-coin" distribution. We conducted the experiments for 10000 samples of size 1000 and found their means do follow the standard normal distribution.

Now, repeat this exercise instead for a standard uniform distribution for 10000 samples of sizes 50, 500, 5000, respectively. And report the results. Are the results consistent with the Central Limit Theorem? **Hint:** To be more effcient, you can code double loops in this case to save time.

Note: This exercise is not different from the example that we covered in class (I actually went through this homework question (intentionally) quickly, too). You just need to modify my code to reflect a different sample size, and a different underlying distribution from which the data are drawn.

4. Optional Questions

5. Question 4. [Learn how to code Loop in R (I)]

Loops are very useful to perform repetitive tasks with efficient code. As we saw in today's class, we can think of Monte Carlo simulation exercises as repeating the drawing process for many many times. To help you better understand how to use loops in R, let's work on the following simple exercise.

Suppose that we want to ask R to print out numbers from 1 to 5 for us. The key is to first identify the common things that we want to do in each repetition. The command that we will use is `print()`. This command will return in R whatever input is. For example,

```
print(1)

## [1] 1
```

So, if we want to accompolish our task, we can simply do the following

```
print(1)

## [1] 1

print(2)

## [1] 2

print(3)

## [1] 3

print(4)

## [1] 4

print(5)

## [1] 5
```

Notice, now, that the only difference between every line of the code is the argument in the `print()` function. If we can ask R to feed this function with numbers from 1 to 5, we are done. Now, write down R code that can accompolish this using **For Loop** in R, as we did in class. Using your code, you should produce the following output.

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
```

6. Question 5. [Learn how to code Loop in R (II)]

Question 4. is relatively straightforward. But suppose now that I want to produce the following results.

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
```

```
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 8
## [1] 9
## [1] 10
## [1] 9
## [1] 10
## [1] 10
```

The pattern is very clear: you first print out $1:10$, then $2:10$, then $3:10$, until $10:10$. This can be done with five lines of code if you use double loops. This question may be difficult for some of your who have never been exposed to programming before, but this exercise would be very useful for you to understand and learn how to program in R.