

VOLUME 1| ISSUE 1| JULY 2026

THE DATA DIALOGUE

Your Weekly Deep Dive into
Insights, Analysis & Trends

The Data Dialogue.

Simplified. Not Simplistic

July. 15th. 2026 | Vol.1 | Issue 1 | Probability Theory

In-Focus

Probability Theory: Their Gambling Hobby became Your Headache

Back in 1654, a French nobleman and serious gambler known as the Chevalier de Méré found himself in a sticky situation. A betting game had to be cut short before anyone could win the pot, leaving them wondering how to split the money fairly.

Chevalier reached out to his buddy, the mathematician Blaise Pascal, who then confided in his friend Pierre de Fermat to solve the problem. Instead of just dividing the pot based on how many rounds each player had *already* won, the duo had a brilliant (and revolutionary) idea: figure out a fair split based on future expectations.

They realized that the past didn't matter; what mattered was counting exactly how many rounds were still needed for each player to win. By mapping out all the possible future scenarios, they could calculate everyone's true chance of winning.

Since then, a chain reaction of mathematical developments: Bernoulli took things a step further, proving that repeating an experiment reveals hidden patterns in chaos, Bayes introduced conditional probability to figure out how things change when new information is added to the mix, Gauss brought us the bell curve (normal distribution), and Pearson and Fisher built the statistical tests we still use constantly.

It's an incredible story of how brilliant theories are born out of everyday problems, passed along from a friend to a friend of a friend. It's exactly why statistics is the beating heart of data science and analysis today—while computer software is just there to crunch the numbers.

Isn't it ironic how one man's gambling hobby became some students' nightmare. Anyway, next time a friend asks for your advice, take it seriously it might be your ticket to a great theory, who knows!

The Data Dialogue.

Simplified. Not Simplistic

July. 15th. 2026 | Vol.1 | Issue 1 | Probability Theory

Tools

You Reap what you Sow? Set.seed() in R

- **When will you need it?** when writing a code that requires generating variable with random values.
- **Why would an analysis require generating variables with random values?** Simply when the idea of uncertainty is part of the analysis, and more specifically in: Monte Carlo Simulations, Resampling methods (bootstrapping & permutation tests), Power analysis, Bayesian analysis, and finally Machine Learning using Synthetic data generation.
- **How set.seed() works?** It tells the computer to pick the same random values every time you (or any one else) run the code. This secures reproducibility of results.

What difference will it make?(See for yourself)

A screenshot of the RStudio interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu is a toolbar with icons for file operations and running code. The main editor window shows a script with the following R code:

```
1 # Generating 5 random decimals between 20 and 60 using uniform distribution
2 random_uniform1 <- runif(n=5, min=20, max=60)
3 random_uniform1
4 random_uniform2 <-runif(n=5, min=20, max=60)
5 random_uniform2
6
7 #repeat it again with same exact command and save it as a random_uniform3 and
8 #how the exact same command generate different results each trial.
9
10
```

The console at the bottom shows the output of the code:

```
> rm(random_uniform1, random_uniform2, Trial1, Trial2)
> # Generating 5 random decimals between 20 and 60 using uniform distribution
> random_uniform1 <- runif(n=5, min=20, max=60)
> random_uniform1
[1] 33.11869 34.54050 58.34180 43.55497 47.81712
> random_uniform2 <-runif(n=5, min=20, max=60)
> random_uniform2
[1] 25.90022 41.58748 49.12800 39.07671 49.17584
>
> #repeat it again with same exact command and save it as a random_uniform3 and
> #how the exact same command generate different results each trial.
>
```

This is the results of generating values for a variable without using the command set.seed()

You can notice that with each trial, you get a totally different result despite the fact that you use the same command.

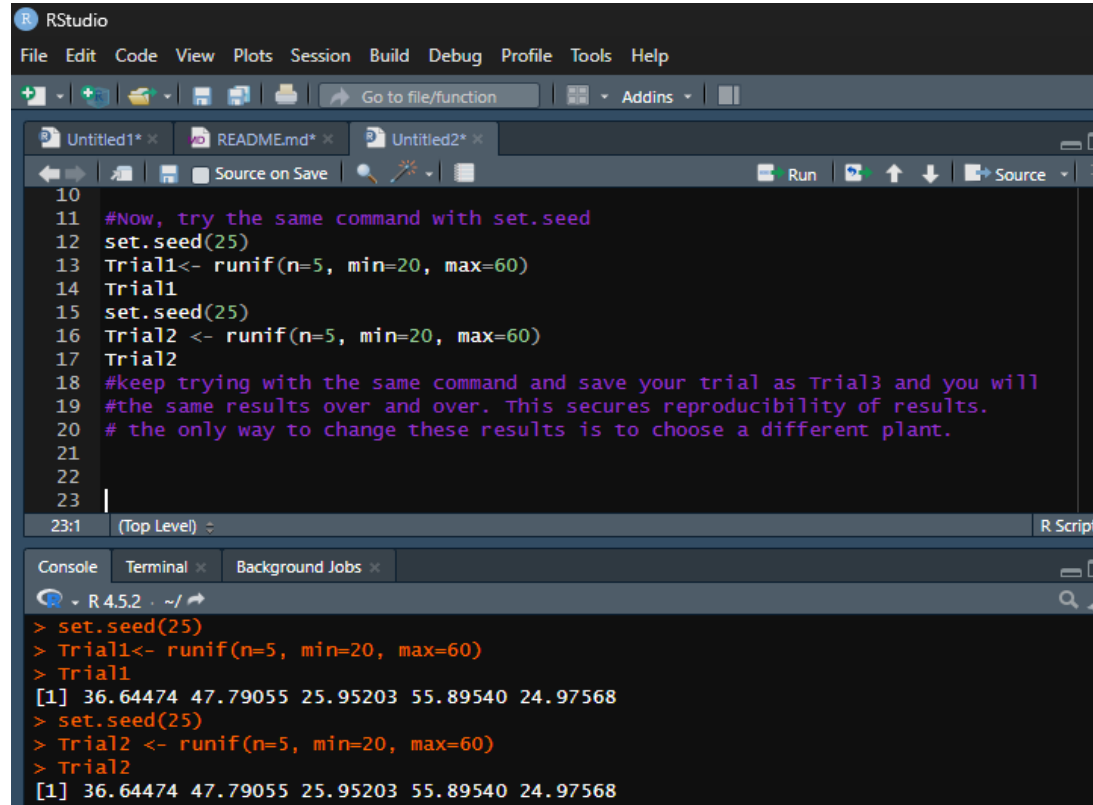
The Data Dialogue.

Simplified. Not Simplistic

July. 15th. 2026 | Vol.1 | Issue 1 | Probability Theory

Here, when using `set.seed()` before the command, it gives back the same exact results.

Notice that we have to repeat the command in line 12 and in line 15 of the script. Each time you attempt reproducing the results, you should use it.



```
10
11 #Now, try the same command with set.seed
12 set.seed(25)
13 Trial1<- runif(n=5, min=20, max=60)
14 Trial1
15 set.seed(25)
16 Trial2 <- runif(n=5, min=20, max=60)
17 Trial2
18 #keep trying with the same command and save your trial as Trial3 and you will
19 #the same results over and over. This secures reproducibility of results.
20 # the only way to change these results is to choose a different plant.
21
22
23
```

```
> set.seed(25)
> Trial1<- runif(n=5, min=20, max=60)
> Trial1
[1] 36.64474 47.79055 25.95203 55.89540 24.97568
> set.seed(25)
> Trial2 <- runif(n=5, min=20, max=60)
> Trial2
[1] 36.64474 47.79055 25.95203 55.89540 24.97568
```

Pitfall Patrol

- Using loop when modifying data in R using command: `for ()`
- Using vectorization instead, which either available as built-in functions or using packages such as `mutate`.

Why?

Assume that you want a variable's values to be multiplied the entry $\times 3$.

Loop command is like a security guard that has to check on every person one by one and follow your instructions (any function. In this case it is multiplication) on every single value one-by-one.

Vectorization is like a facial recognition camera that can check on all the individuals simultaneously. Loops take way more time than vectorization and slow your software and laptop down.

The Data Dialogue.

Simplified. Not Simplistic

July. 15th. 2026 | Vol.1 | Issue 1 | Probability Theory

Pandora's Box *Is synthetic data the same* *as data fabrication?*

Synthetic data are usually used in machine learning applications. It is when you ask the software (usually R) to create data using specific attributes. This might be completely synthetic or partially synthetic when you give it some data collected from human participants.

Skeptic voices argue against synthetic data and call it fabricated data. For them, fake data is not different from fake news and it will lead to fake conclusions. Training AI on old data to get new data and claim new insights is similar to your mother recycling bread into a cake and claiming baking you a new dessert. Allowing synthetic data, they argue, lowers the bar for researchers who would love to escape academic audit. Finally, opponents argue that using synthetic data breaks down the scientific wheel where theoretical expectations should be checked against real world empirics not unrealistic data.

On the other hand, proponents of synthetic data admire it for canceling out concerns about human privacy and confidentiality compared to having your name or ID masked or anonymized and stored somewhere on a student's laptop. For them, it is noise-free, highly precise data, and this it is not fake.

Before you Go

Did it happen that you made a mistake while conducting data analysis? If so, send us a punchy 250 words describing this mistake and how you fixed it.

If it's humanized, we'll publish it next week with your name, title and link to your profile.

Disclaimer: *"We reserve the right to edit all submissions for conversational tone, length, and formatting to ensure it matches the Data Dialogue voice"*

You will find the submission button on my website.