

Statistics with Recitation: TA Session

Danny Po-Hsien Kang (康柏賢)

September 23, 2025

Today's agenda

1 Random Number Generator

- `set.seed()`
- `runif()`

2 Sequence

- `as.numeric()`
- `cumsum()`
- `seq_along()`

Set Random Seed: set.seed()

- set.seed() sets R's random-number generator (RNG) state
 - Random sampling functions: runif(), rnorm(), sample()
 - Key to **reproducibility**: It guarantees that the code produce the same sequence every time you rerun.
- **Syntax:**

```
set.seed(seed)
```

- **Example:**

```
set.seed(20250923)
```

Generate Uniform Random Samples: `runif()`

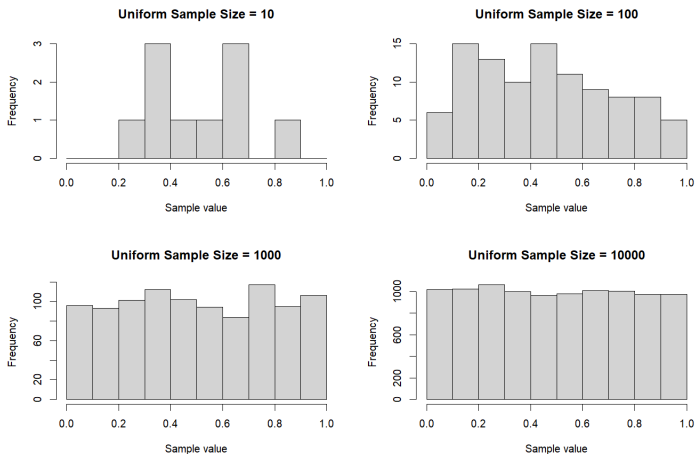


Figure: Random samples drawn from $\text{Uniform}[0,1]$ in different sizes

Generate Uniform Random Samples: runif()

- **Syntax:**

```
num <- runif(n = ..., min = ..., max = ...)
```

- **Example:**

```
rdnum <- runif(10) # min = 0, max = 1  
rdnum_2 <- runif(n = 10, min = 0, max = 100)
```

- **Output**

```
print(rdnum)  
[1] 0.5246533 0.3351281 0.2918250 0.0136793 0.0681535  
[6] 0.3955869 0.0036232 0.0449897 0.2758995 0.6786644  
  
print(rdnum_2)  
[1] 39.070 28.228 64.084 44.266 34.400  
[6] 62.392 53.085 36.450 62.883 80.622
```

Create Numeric Type Object: as.numeric()

- **Syntax:**

```
data_numeric <- as.numeric(vector)
```

- **Example:**

```
head_logical <- rdnum > 0.5  
head_numeric <- as.numeric(rdnum > 0.5)
```

- **Output**

```
print(head_logical)  
[1] TRUE FALSE FALSE FALSE FALSE  
[6] FALSE FALSE FALSE FALSE TRUE  
  
print(head_numeric)  
[1] 1 0 0 0 0 0 0 0 0 1
```

Create Numeric Type Object: `as.numeric()`

- Lists and data.frame can not be convert by `as.numeric()` directly.
- Similar (and useful) functions: `as.logical()`, `as.integer()`, `as.character()`
- `as.numeric()` is a strong tool when doing data cleaning/analysis.

```
data_example <- data.frame(name = c("Alice", "Bob", "Cindy"),
                           score = c("80", "60", "100")
                           )
mean(data_example$score) # NA !
mean(as.numeric(data_example$score)) # 80
```

Supplement: Operator \$ vs [] vs [[]]

- A data.frame is actually a special kind of list
 - Each element of the list is a column (vector having asame length)
- [[]] can extract a single element of a list.

```
data_example[["name"]] # "Alice", "Bob", "Cindy"
data_example[[1]] # "Alice", "Bob", "Cindy"
```

- \$ is the shorter syntax for [[]]

```
data_example$name # "Alice", "Bob", "Cindy"
```

- While using [] returns a smaller dataframe

```
data_example["name"]
  name
1 Alice
2  Bob
3 Cindy
```


Cumulative Sum of Sequence: cumsum()

- **Syntax:**

```
cumulative_data <- cumsum(data)
```

- **Example:**

```
cum_head <- cumsum(head_numeric)
```

- **Output:**

```
print(cum_head)  
[1] 1 1 1 1 1 1 1 1 1 2
```

Generate Serial Number Sequence: seq_along()

- **Syntax:**

```
data_serial_number <- seq_along(data)
```

- **Example:**

```
cum_head_serial <- seq_along(cum_head)
```

- **Output:**

```
print(cum_head_serial)
[1]  1  2  3  4  5  6  7  8  9 10
```