R Programming - Part 2

Essential Functions

Table of contents

1 unique()

- The `unique()` function removes duplicated elements from a vector or data frame.
- Example: `unique()`

```r
x = c(1, 2, 2, 3, 4, 4, 5)
unique(x)
```

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

2 any() and all()

- `any()` returns `TRUE` if any of the values are `TRUE`.
- `all()` returns `TRUE` if all of the values are `TRUE`.
- Example: `any()`

```r
v = c(FALSE, FALSE, TRUE)
any(v)
```

```
[1] TRUE
```

- Example: `all()`

```r
all(v)
```

```
[1] FALSE
```

3 Set Functions

- Set is a collection of distinct elements.
- Set functions perform operations on sets of elements.

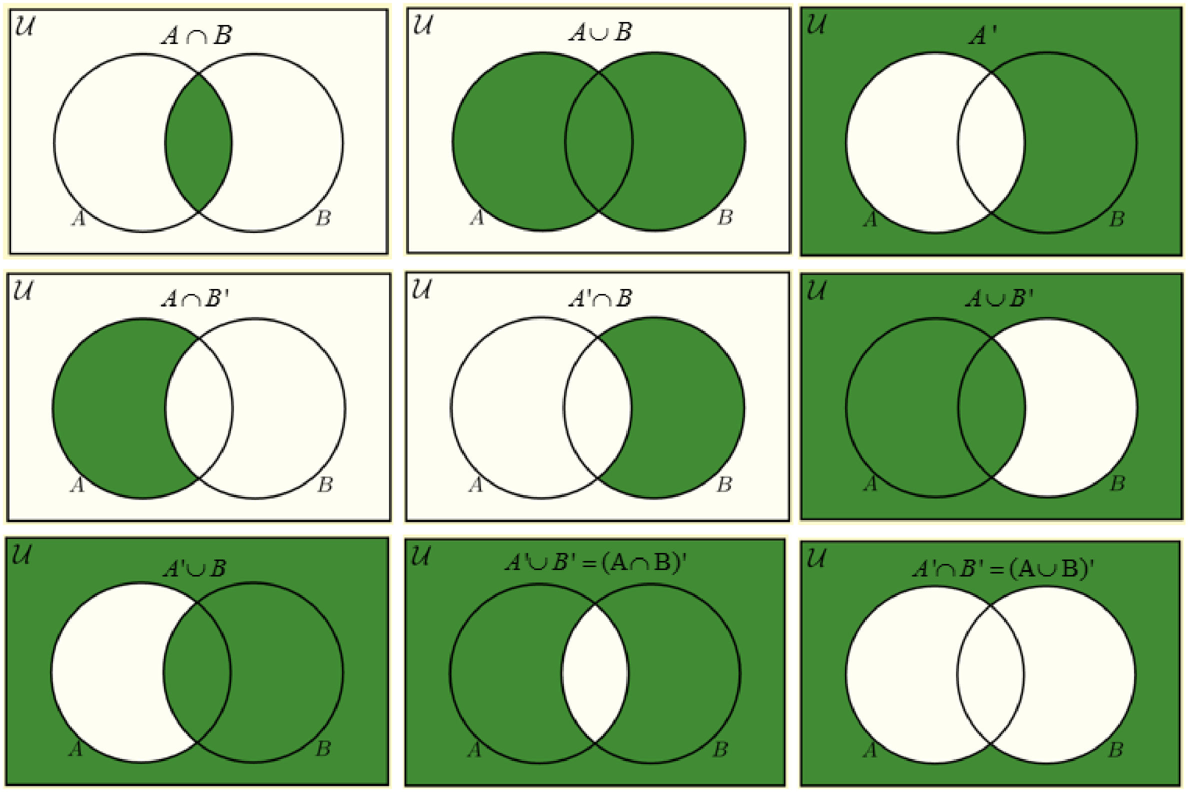

Figure 1: Set Theory

3.1 Union ($A \cup B$)

- The set of all elements in A, or in B, or in both.
- $A \cup B = \{x | x \in A \text{ or } x \in B\}$
- `union()`
- Example: ::: {.cell}

```r
A = c(1, 2, 3, 4)
B = c(3, 4, 5, 6)
union(A, B)
```

```
[1] 1 2 3 4 5 6
```

:::

3.2 Intersection ($A \cap B$)

- The set of all elements that are both in A and B.
- $A \cap B = \{x | x \in A \text{ and } x \in B\}$
- `intersect()`
- Example:

```r
A = c(1, 2, 3, 4)
B = c(3, 4, 5, 6)
intersect(A, B)
```

```
[1] 3 4
```

3.3 Set Difference ($A - B$)

- The set of all elements that are in A but not in B.
- $A - B = \{x | x \in A \text{ and } x \notin B\}$
- `setdiff()`
- Example:

```r
A = c(1, 2, 3, 4)
B = c(3, 4, 5, 6)
setdiff(A, B)
```

```
[1] 1 2
```

3.4 Subset ($A \subseteq B$)

- A is a subset of B if every element of A is also an element of B.
- $A \subseteq B \iff (\forall x)(x \in A \implies x \in B)$
- Example:

```r
A = c(1, 2, 3, 4)
all(A %in% c(1, 2, 3, 4, 5))
```

```
[1] TRUE
```

```r
all(A %in% c(1, 2, 3))
```

```
[1] FALSE
```

3.5 Set Equality ($A = B$)

- Two sets are equal if they have exactly the same elements.
- $A = B \iff (A \subseteq B)$ and $(B \subseteq A)$
- `setequal()`
- Example:

```r
A = c(1, 2, 3, 4)
B = c(3, 4, 5, 6)
setequal(A, B)
```

```
[1] FALSE
```

4 Random Sampling

Simple Random Sample (SRS): is a subset of a population, chosen in such a way that every possible sample of a given size has an equal chance of being selected. This method ensures that each individual or item within the population has an equal probability of being included in the sample, and the selection process is entirely by chance, without any bias.

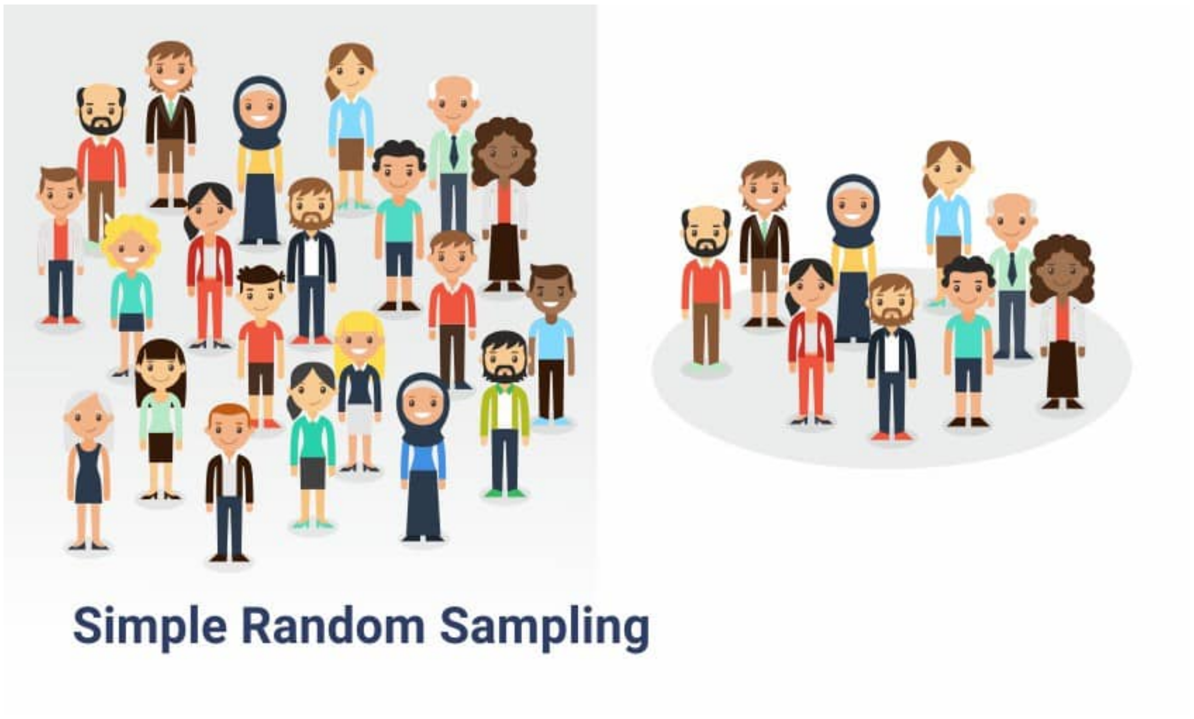

Figure 2: Random Sampling

4.1 With *vs.* Without Replacement

Sampling With Replacement (SWR): In this method, after an individual or item is selected for the sample, it is placed back into the population before the next selection is made, allowing for the possibility of being chosen more than once. This method is particularly useful when dealing with small population sizes or when it's important to maintain the same population size for each draw.

Sampling Without Replacement (SWOR): Contrary to SWR, in Sampling Without Replacement, once an individual or item is selected, it is not placed back into the population, and hence, cannot be selected again. This method is often utilized when the population size is large, or when maintaining the same population size for each draw is not crucial.

4.2 sample()

- The `sample()` function draws random samples from a vector.
- Syntax:

```r
sample(x, size, replace = FALSE, prob = NULL)
```

- Example:

```r
sample(1:10, 5)
```

```
[1] 10  6  8  9  2
```

5 cbind() and rbind()

- `cbind()` combines vectors, matrices, or data frames by columns.
- `rbind()` combines vectors, matrices, or data frames by rows.

5.1 Example: cbind

```r
A = matrix(1:4, ncol=2)
A
```

1	3
2	4

```r
B = matrix(5:8, ncol=2)
B
```

5	7
6	8

```r
cbind(A, B)
```

1	3	5	7
2	4	6	8

5.2 Example: `rbind`

```r
rbind(A, B)
```

1	3
2	4
5	7
6	8

6 `ifelse()`

- The `ifelse` function applies a function to elements of a vector depending on a condition.
- Example:

```r
numbers = 1:10
numbers
```

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

```r
ifelse(numbers %% 2 == 0, "Even", "Odd")
```

```
 [1] "Odd"  "Even" "Odd"  "Even" "Odd"  "Even" "Odd"  "Even" "Odd"  "Even"
```

7 Testing Data Types

- R provides functions to test the data type of a variable.
- Examples:

```r
is.character("Hello")
```

```
[1] TRUE
```

```r
is.numeric(10)
```

```
[1] TRUE
```

```r
is.na(NA)
```

```
[1] TRUE
```

```r
is.vector(c(1, 2, 3))
```

```
[1] TRUE
```

```r
is.matrix(matrix(1:4, ncol=2))
```

```
[1] TRUE
```

```r
is.data.frame(data.frame(x=1:4, y=4:1))
```

```
[1] TRUE
```

```r
is.factor(factor(c("a", "b", "a")))
```

```
[1] TRUE
```

8 Exercise: Coin Flip Simulation

8.1 Part 1: Single Simulation

Write an R function `coin_flip()` that simulates flipping a coin. The function should return `H` (for head) or `T` (for tail).

```r
coin_flip = function() {
  flip = sample(c("H", "T"), size = 1)
  return (flip)
}

coin_flip()
```

```
[1] "T"
```

8.2 Part 2: Multiple Simulations

Now, extend your function to perform multiple simulations of coin flips and return the number of heads and tails.

```r
coin_flip = function(n) {
  flips = sample(c("H", "T"), size = n, replace = TRUE)
  return(table(flips))
}

coin_flip(5)
```

H	T
4	1

8.3 Part 3: Analysis

Analyze the results of your multiple simulations. What do you observe as the number of flips increases?

```
coin_flip(10)/10
```

H	T
0.6	0.4

```
coin_flip(100)/100
```

H	T
0.53	0.47

```
coin_flip(1000)/1000
```

H	T
0.489	0.511

```
coin_flip(10000)/10000
```

H	T
0.5115	0.4885