Essentials of Data Science With R Software - 1

Probability and Statistical Inference

Introduction to R Software

:::

Lecture 7

Logical Operators and Selection of Sample

Shalabh

Department of Mathematics and Statistics

Indian Institute of Technology Kanpur

Logical Operators and Comparisons:

- **The cities and households are categorized in the data as 1, 2,…**

- **We want to find the mean of income those households which are in the cities coded as 1.**

- **We want to find the mean of income of the households in cities coded as 1 having household size more than 3.**

- **Less than, more than and not equal to are the logical operations, not mathematical operations.**

Logical Operators and Comparisons

The following table shows the operations and functions for logical comparisons (True or False).

TRUE and FALSE are reserved words denoting logical constants.

Operator	Executions
>	Greater than
>=	Greater than or equal
<	Less than
<=	Less than or equal
==	Exactly equal to
!=	Not equal to
!	Negation (not)

Logical Operators and Comparisons

TRUE and FALSE are reserved words denoting logical constants

Operator	Executions
`xor()`	either... or (exclusive)
`isTRUE(x)`	test if `x` is TRUE
`TRUE`	true
`FALSE`	false

Examples:

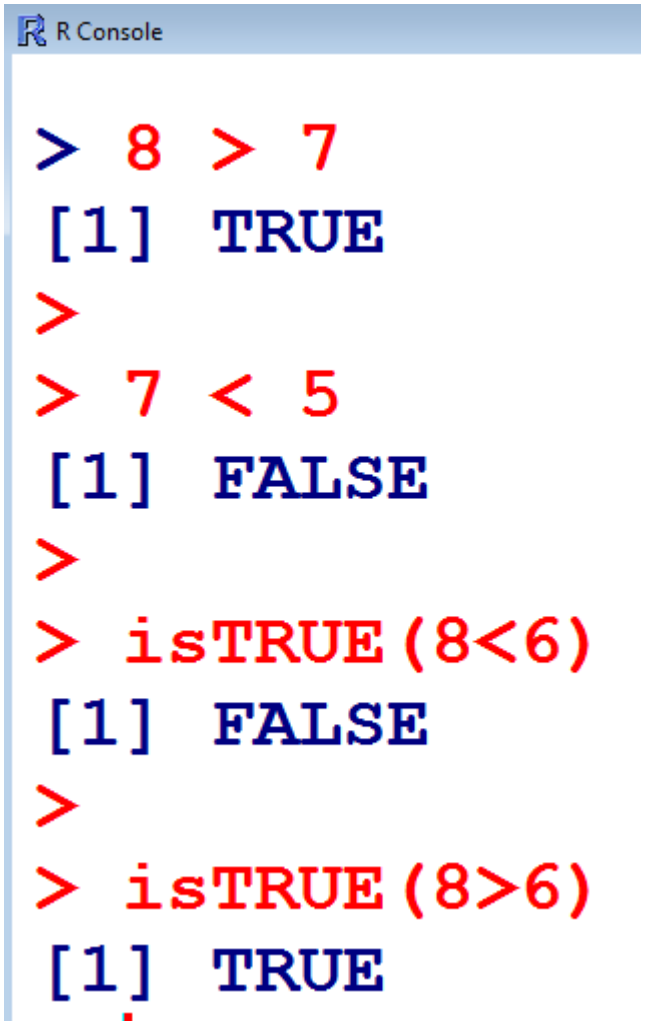
```
> 8 > 7
[1] TRUE

> 7 < 5
[1] FALSE
```

Is 8 less than 6?

```
> isTRUE(8<6)
[1] FALSE
```

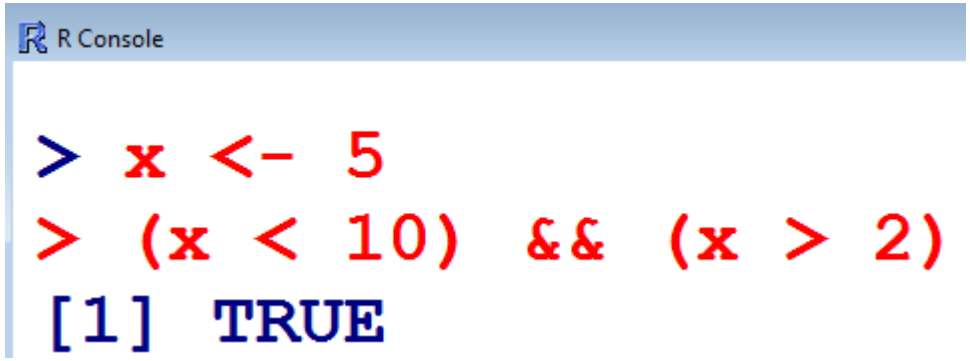
Is 8 greater than 6?

```
> isTRUE(8>6)
[1] TRUE
```



Examples:

```
> x <- 5
> (x < 10) && (x > 2)   # && means AND
[1] TRUE
```



Examples:

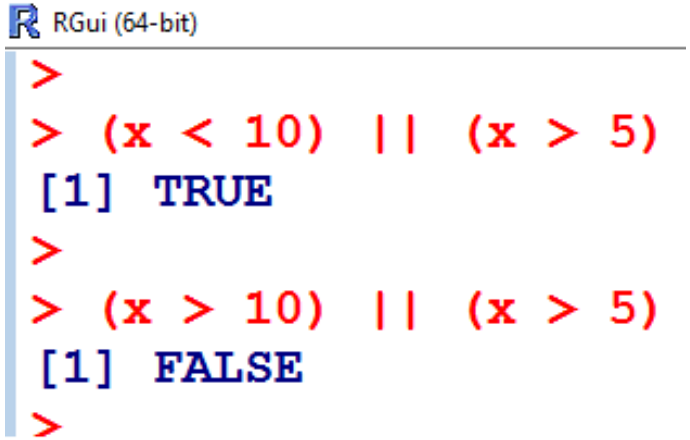
```
> x <- 5
```

Is `x` less than 10 or `x` is greater than 5 ?

```
> (x < 10) || (x > 5)     # || means OR
[1] TRUE
```

Is `x` greater than 10 or `x` is greater than 5 ?

```
> (x > 10) || (x > 5)
[1] FALSE
```



Examples:

```
> x = 10
> y = 20
```

Is x equal to 10 and is y equal to 20?

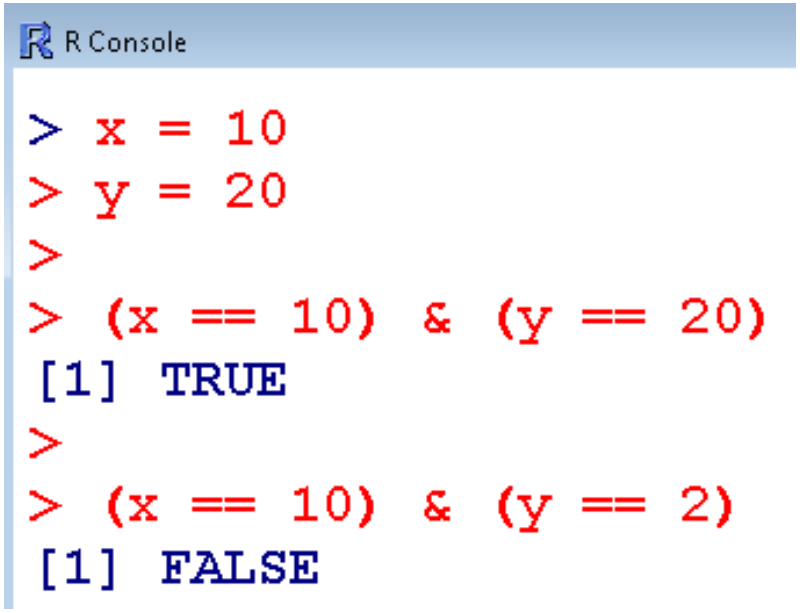
```
> (x == 10) & (y == 20)
```

```
[1] TRUE
```

```
# == means exactly
  equal to
```

Is x equal to 10 and is y equal to 2?

```
> (x == 10) & (y == 2)
[1] FALSE
```



Simple Random Sampling:

Simple random sampling (SRS) is a method of selection of a sample comprising of *n* number of sampling units from the population having *N* number of units such that every sampling unit has an equal chance of being chosen.

Simple Random Sampling Without and With Replacement:

SRSWOR

The sampling units are chosen without replacement in the sense that the units once chosen are not placed back in the population .

SRSWR

The sampling units are chosen with replacement in the sense that the chosen units are placed back in the population.

Drawing a Simple Random Sample Without Replacement (SRSWOR)

`sample` takes a sample of the specified size from the elements of `x` using either with or without replacement.

Usage

```
sample(x, size, replace = FALSE)
```

Arguments

`x` Either a vector of one or more elements from which to choose, or a positive integer.

`size` a non-negative integer giving the number of items to choose.

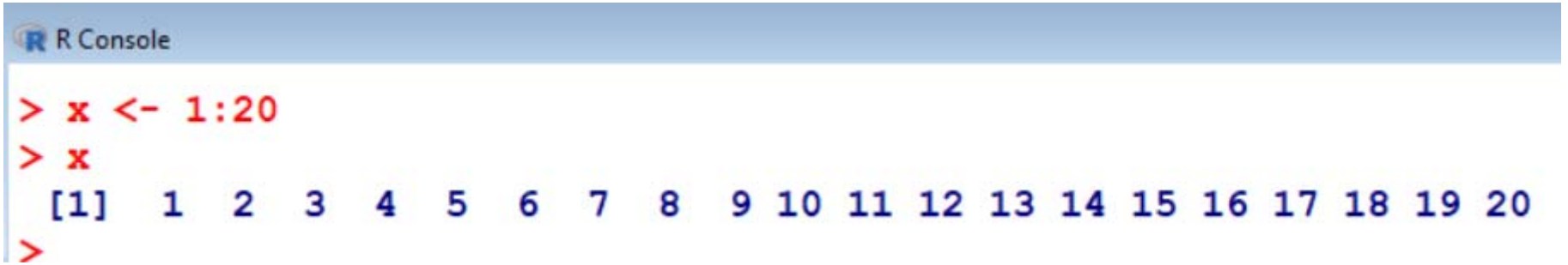
`replace` Should sampling be with replacement?

Drawing a Simple Random Sample Without Replacement (SRSWOR)

First we define a population units containing the numbers 1 to 20.

This can be defined by a sequence as x.

```
> x <- 1:20
> x
[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
18 19 20
```



Drawing a Simple Random Sample Without Replacement (SRSWOR)

Let us draw the sample of size 5 from population `x` by SRSWOR .

This is controlled by the statement `replace = FALSE` inside the argument.

```
> x
[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
18 19 20
```

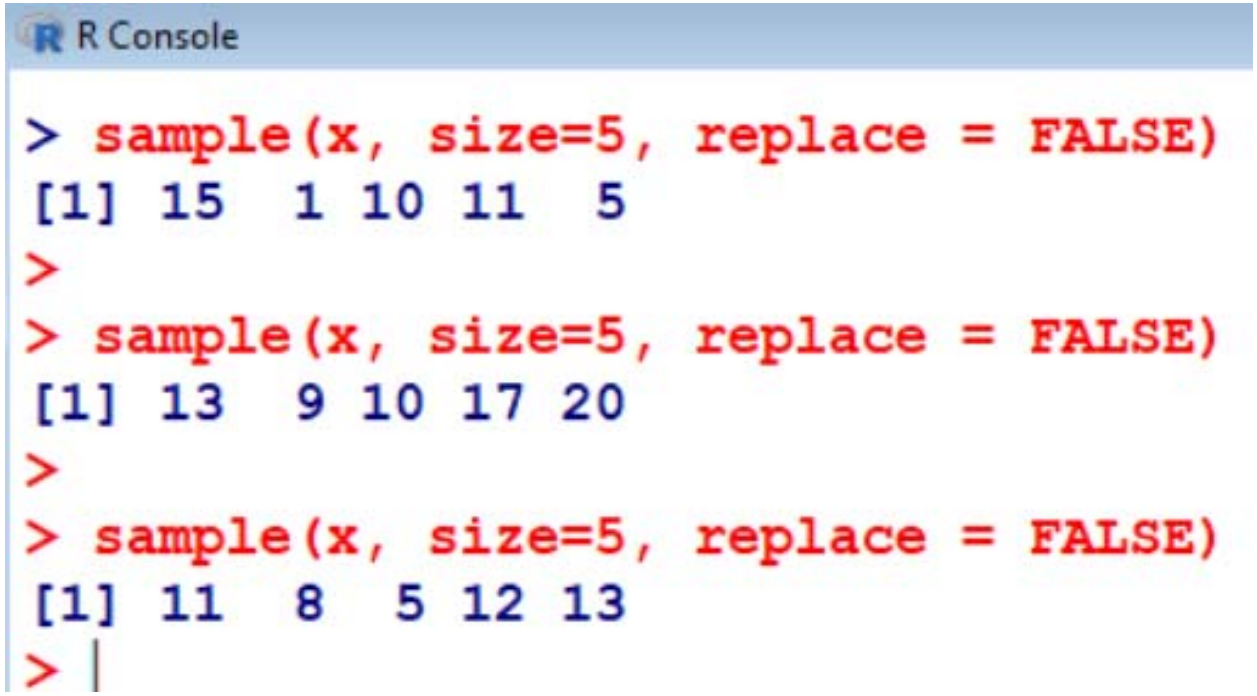
SRSWOR command

```
sample(x, size=5, replace = FALSE)
```

Drawing a Simple Random Sample Without Replacement (SRSWOR)

```
> sample(x, size=5, replace = FALSE)
[1] 15  1 10 11  5

> sample(x, size=5, replace = FALSE)
[1] 13  9 10 17 20

> sample(x, size=5, replace = FALSE)
[1] 11  8  5 12 13
```

Drawing a Simple Random Sample Without Replacement (SRSWOR)



Drawing a Simple Random Sample With Replacement (SRSWR)

Let us draw the sample of size 10 from population `x` by SRSWR .

This is controlled by the statement `replace = TRUE` inside the argument.

```
> x
[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
18 19 20
```

SRSWR Command

```
sample(x, size=10, replace = TRUE)
```

Drawing a Simple Random Sample With Replacement (SRSWR)

SRSWR

```
> sample(x, size=10, replace = TRUE)
 [1]  4 17  6  3 20 14 13  2 15  2
```

Value 2 is repeated.

```
> sample(x, size=10, replace = TRUE)
[1]  5 12 7  4 18  2 12  1  3 7
```

Values 12 and 7 are repeated.

```
> sample(x, size=10, replace = TRUE)
 [1] 15 11 19 10  4  3 11 17  9  3
```

Value 11 is repeated.

Drawing a Simple Random Sample With Replacement (SRSWR)

R Console

```
> x
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
> sample(x, size=10, replace = TRUE)
 [1]  4 17  6  3 20 14 13  2 15  2
> sample(x, size=10, replace = TRUE)
 [1]  5 12  7  4 18  2 12  1  3  7
>
> sample(x, size=10, replace = TRUE)
 [1] 15 11 19 10  4  3 11 17  9  3
>
```

Drawing a Simple Random Sample With Replacement (SRSWR)

```
> x
[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
18 19 20
```

For `sample` the default for size is the number of items inferred from the first argument, so that `sample(x)` generates a random permutation of the elements of `x` (or `1:x`).

```
> sample(x)
 [1] 19  2  1  7 12 15  4 14 13  5 10 17  6 16
18  9 20  3  8 11

> sample(x)
 [1]  6 15  8  2 14  9 18 12  4 17  7  5 20 13
1 16 11  3 10 19
```

Drawing a Simple Random Sample With Replacement (SRSWR)