R Essentials

Operators and Basic Vector Types

Tony Yao-Jen Kuo

Agenda

- An overview
- Assignment operators
- Arithmetic operators and numerics
- `character`
- Logical operators and `logical`
- Flow of control: decision making

An overview

3 types of operator

- Assignment operators
- Arithmetic operators
- Logical operators

5 basic vector types in R

- numerics
    - `numeric`
    - (optional)`integer`
    - (optional)`complex`
- `character`
- `logical`

Using `class()` to check vector types

```
In [1]: ?class
```

```
In [2]: class(87)
        class("Luke Skywalker")
        class(TRUE)
```

```
'numeric'

'character'

'logical'
```

Assignment operators

Using <- or = for assignment

- = is OKAY, but <- is more common in R
- Using `Alt + -` in **RStudio** to get <-

In [3]:

```
lucky_number <- 24
lucky_number
```

```
24
```

In [4]:

```
lucky_number = 24
lucky_number
```

```
24
```

Arithmetic operators and numerics

There are 7 basic arithmetic operators

Operators	Usage
+	add
–	minus
*	multiply
/	divide
** or ^	power
%%	modulo
%/%	quotient

Using `()` for **priority**.

```
In [5]: 8 + 7
```

```
15
```

```
In [6]: 8 - 7
```

```
1
```

```
In [7]: 8 * 7
```

```
56
```

```
In [8]: 8 / 7
```

```
1.142857142857143
```

```
In [9]: 8**2 # 8^2
```

```
64
```

```
In [10]: 8 %% 5
```

```
3
```

```
In [11]: 8 %/% 5
```

```
1
```

```
In [12]: 8 + 7 * 2
         (8 + 7) * 2
```

```
22

30
```

Using `numeric` as the primary digit type

In [13]:

```
my_lucky_number <- 24
class(my_lucky_number) # numeric
class(2.4) # numeric
class(-8.7) # numeric
class(0) # numeric
```

```
'numeric'

'numeric'

'numeric'

'numeric'
```

Using L to denote an integer vector

In [14]:
```
my_lucky_integer <- 87L
class(my_lucky_integer) # integer
my_lucky_integer
```

'integer'

87

Using i to denote imaginary part and get a complex vector

In [15]:

```
my_lucky_complex <- 8 + 7i
class(my_lucky_complex) # complex
my_lucky_complex
```

```
'complex'

8+7i
```

Practices: Kilometer to Mile

$$Miles = Kilometers \times 0.62137$$

In [16]:

```
marathon_km <- 42.195
# marathon_mile
```

Practices: Celsius to Fahrenheit

$$Fahrenheit_{(^{\circ}F)} = Celsius_{(^{\circ}C)} \times 1.8 + 32$$

In [17]:

```
current_temp_celsius <- 20
# current_temp_fahrenheit
```

Practices: Jeremy Lin’s BMI

$$BMI = \frac{weight_{kg}}{height_m^2}$$

In [18]:

```
jeremy_lin_height <- 191
jeremy_lin_weight <- 91
# jeremy_lin_bmi
```

character

Using ” or ”” for characters

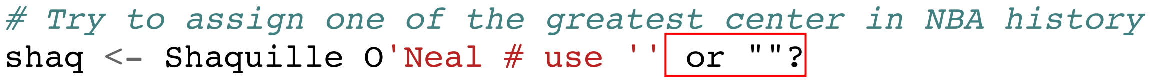
```
In [19]: mj <- "Michael Jordan"
         class(mj)
```

'character'

```
In [20]: mj <- 'Michael Jordan'
         class(mj)
```

'character'

When to use ” or ””?



```
# Try to assign one of the greatest center in NBA history
shaq <- Shaquille O'Neal # use '' or ""?
```

In [21]:

```
#shaq <- 'Shaquille O'Neal' # error
shaq <- 'Shaquille O\'Neal' # \ is the escape symbol
shaq <- "'Shaquille O'Neal'"
```

Practices: What did Ross Geller say?

```
Let's put aside the fact that you "accidentally" pick up my grand mother's ring.
```

Using `sprintf()` for string print with format

```
In [22]: jeremy_lin_height <- 1.91
         jeremy_lin_weight <- 91
         jeremy_lin_bmi <- jeremy_lin_weight / jeremy_lin_height**2
         sprintf("Jeremy Lin's BMI is %s", jeremy_lin_bmi)
```

```
'Jeremy Lin\'s BMI is 24.9444916531893'
```

Common formats using `sprintf()`

- `%s`: pure text
- `%f`: float
- `%e`: scientific notation

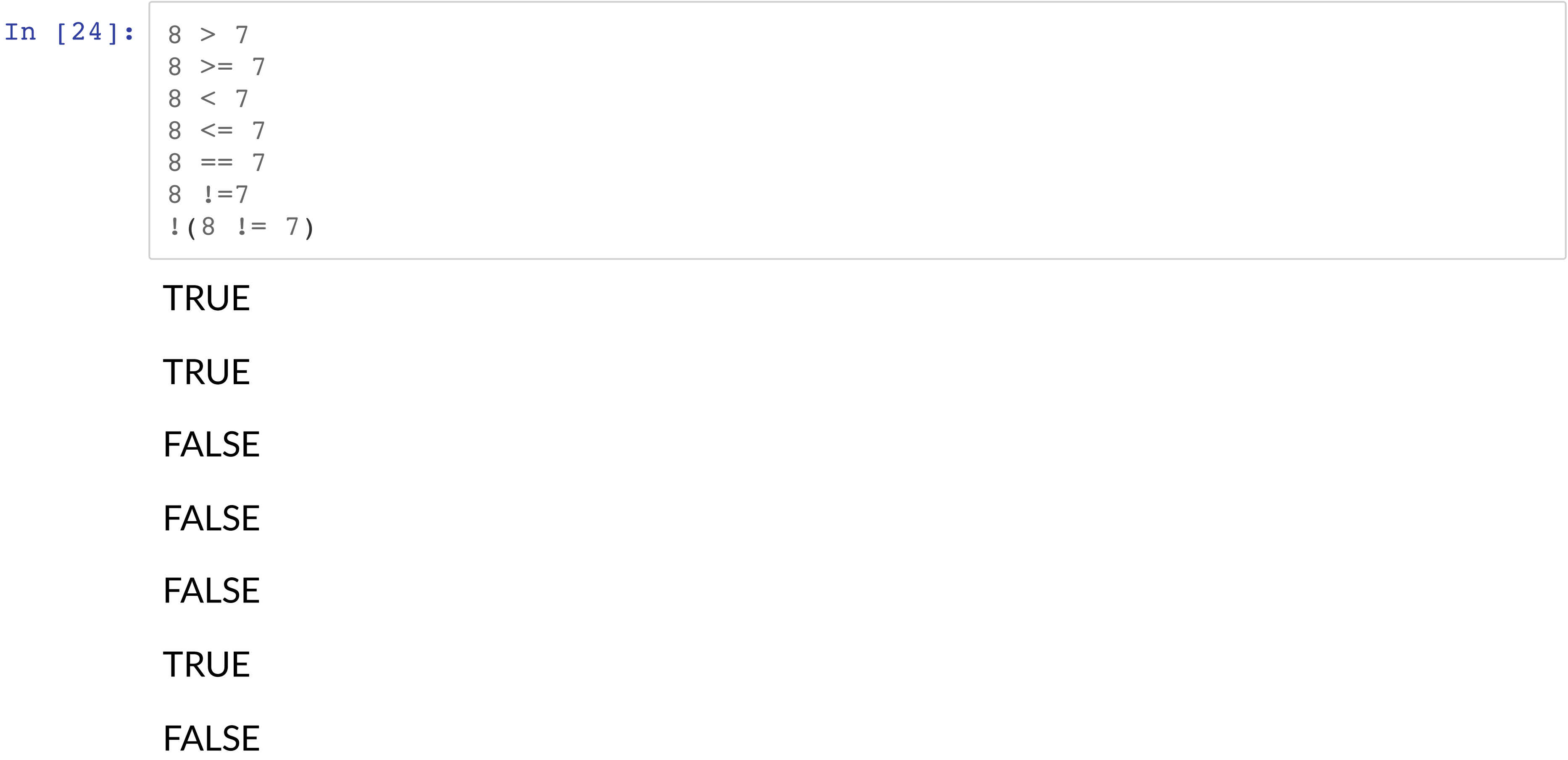
In [23]:

```
?sprintf # this function is well-documented!
```

Logical operators and `logical`

Logical operators

Operators	Usage
>	larger than
>=	larger than or equal to
<	smaller than
<=	smaller than or equal to
==	equal to
!=	not equal to
%in%	belong to
!	not
&	logical AND
|	logical OR



In [24]:

```
8 > 7
8 >= 7
8 < 7
8 <= 7
8 == 7
8 !=7
!(8 != 7)
```

```
TRUE

TRUE

FALSE

FALSE

FALSE

TRUE

FALSE
```

Be case-sensitive!

```
In [25]: class(TRUE)
         class(FALSE)
         #class(True)  # error
         #class(False) # error
         #class(true)  # error
         #class(false) # error
```

```
'logical'

'logical'
```

Comparing characters

```
"Z" > "z" > "Y" > "y" > "X" > "x" ... > "B" > "b" > "A" > "a"
```

Is Shaq overweight?

Well, that depends on the definition of "overweight".

In [26]:

```
# Overweight: BMI > 30 AND body fat > 25%
shaq_bmi <- 31
shaq_body_fat <- 0.15
```

In [27]:

```
# Overweight: BMI > 30 OR body fat > 25%
shaq_bmi <- 31
shaq_body_fat <- 0.15
```

Flow of control: decision making

Using logical vector in control statements

```
# single branch
if (EXPR) {
  # do something when EXPR is evaluated as TRUE
}
```

Using `if-else` for 2 branches

```
# 2 branches
if (EXPR) {
  # do something when EXPR is evaluated as TRUE
} else {
  # do something when EXPR is evaluated as FALSE
}
```

Using `if-else if-else` for more than 3 branches

```
# 3 branches
if (EXPR_1) {
  # do something when EXPR_1 is evaluated as TRUE
} else if (EXPR_2) {
  # do something when EXPR_2 is evaluated as TRUE
} else {
  # do something when both EXPR_1 and EXPR_2 are evaluated as FALSE
}
```

Getting BMI Labels

https://en.wikipedia.org/wiki/Body_mass_index (https://en.wikipedia.org/wiki/Body_mass_index)

In [28]:

```r
player_bmi <- 24

if (player_bmi > 30) {
  print("Obese")
} else if (player_bmi < 18.5) {
  print("Underweight")
} else if ((player_bmi >= 18.5) & (player_bmi < 25)) {
  print("Normal weight")
} else {
  print("Overweight")
}
```

```
[1] "Normal weight"
```

Practices: fizz buzz

- if `int` can be divided by 3, return "fizz"
- if `int` can be divided by 5, return "buzz"
- if `int` can be divided by 15, return "fizz buzz"
- otherwise, return `int` itself

In [29]:

```
int <- 3
# ...
# "fizz"
```

In [30]:

```
int <- 5
# ...
# "buzz"
```

In [31]:

```
int <- 15
# ...
# "fizz buzz"
```

In [32]:

```
int <- 16
# ...
# 16
```