Programming: Intro to Conditionals

Stat 133 with Gaston Sanchez

Creative Commons Attribution Share-Alike 4.0 International CC BY-SA

Introduction to if-else

Conditionals

If-else or if-then-else

Use to decide what to do based on a logical condition

Motivation example

Is it positive or negative?



```
x <- rnorm(1)
```

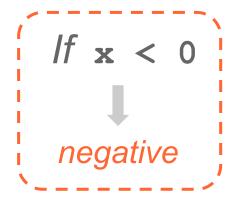
Is it positive or negative?

Is it positive or negative?

x < - rnorm(1)

```
If x > 0
positive
```

```
if (x > 0) {
   print("positive")
} else {
   print("negative")
}
```



Another option

```
x <- rnorm(1)
```

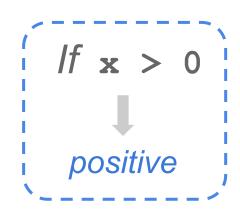
Is it positive or negative?

Is it positive or negative?

```
If x < 0
negative
```

```
x <- rnorm(1)

if (x < 0) {
   print("negative")
} else {
   print("positive")
}</pre>
```



Anatomy of an if-then-else statement

```
x <- rnorm(1)
if (x > 0) {
 print("positive")
} else {
 print("negative")
```

```
x < - rnorm(1)
if-else statement
if (x > 0) {
 print("positive")
} else {
 print("negative")
```

```
x <- rnorm(1)
     Logical condition single TRUE single FALSE
if (x > 0) {
  print("positive")
} else {
  print("negative")
```

```
x < - rnorm(1)
if (x > 0) {
                        What to do if
  print("positive")
                        condition is TRUF
} else {
  print("negative")
```

```
x <- rnorm(1)
if (x > 0) {
  print("positive")
} else {
  print("negative") What to do if
                        condition is FALSE
```

When you don't care about the condition being FALSE

```
x <- rnorm(1)
if (x > 0) {
  print("positive")
  else {
 print("negative")
                 What if you don't
                 care about this?
```

```
x <- rnorm(1)

if (x > 0) {
  print("positive")
}
```

If you don't care about the else clause, then don't use it

```
x <- rnorm(1)

if (x > 0) {
  print("positive")
} else NULL
```

Equivalent: when you don't care about the **else** clause

Multiple chained if's

```
x <- rnorm(1)
if (x > 0) {
  print("positive")
\} else if (x < 0) {
  print("negative")
} else if (x == 0) {
  print("zero")
```

```
x <- rnorm(1)
if (x > 0) {
  print("positive")
else if (x < 0) 
  print("negative")
} else if (x == 0) {
  print("zero")
```

```
x <- rnorm(1)
if (x > 0) {
  print("positive")
\} else if (x < 0) {
  print("negative")
} else if (x == 0) {
  print("zero")
```

```
x <- rnorm(1)
if (x > 0) {
  print("positive")
} else if (x < 0) {
  print("negative")
} else if (x == 0) {
  print("zero")
```

```
x <- rnorm(1)
if (x > 0) {
  print("positive")
\} else if (x < 0) {
  print("negative")
} else {
  print("zero")
```

Errors and Warnings

Error and Warning messages

There are 2 main functions for generating errors and warnings:

```
stop()
warning()
```

*There's also the function stopifnot()

Error and Warning messages

Use stop () to stop the execution of a function, raising an error message.

Use warning () to show a warning message, without stopping execution.

A warning is useful when we don't want to stop execution, but we still want to show potential issues/errors.

29

Example

Future Value (in its simplest version)

$$FV = p (1 + r)^n$$

where:

p = principal

r = interest rate (annual)

n = time (years)

```
# future value function
future value <- function(p=1, r=0.01, n=1) {
  fv = p * (1 + r)^n
  fv
# how much would you get in 5 years if you
# invest 1000 at a 4% annual rate of return?
future value (p=1000, r=0.04, n=5)
```

```
# negative returns?
future_value(p=1000, r=-0.04, n=5)

# negative principal (e.g. paying debt)?
future_value(p=-1000, r=0.04, n=5)

# negative time?
future_value(p=1000, r=0.04, n=-5)
```

Negative time doesn't make much sense

Error Messages

```
# future value function
future_value <- function(p=1, r=0.01, n=1) {
   if (n < 0) {
      stop('n cannot be positive')
   } else {
      fv = p * (1 + r)^n
      return(fv)
   }
}</pre>
```

stop() will stop execution with an error message

```
# future value function
future_value <- function(p=1, r=0.01, n=1) {
   if (n < 0) {
      stop('n cannot be positive')
   } else {
   fv = p * (1 + r)^n
   return(fv)
   }
}</pre>
```

stop() will stop execution with an error message

```
# future value function
future_value <- function(p=1, r=0.01, n=1) {
   if (n < 0) {
      stop('n cannot be positive')
   }
   fv = p * (1 + r)^n
   fv
}</pre>
```

stop() will stop execution with an error message

Warning Messages

```
# future value function
future value <- function(p=1, r=0.01, n=1) {
  if (n < 0) {
     warning('n cannot be negative')
     n = -n
  } else {
     fv = p * (1 + r)^n
     return(fv)
```

warning() does not stop execution but gives a warning message

```
# future value function
future value <- function(p=1, r=0.01, n=1) {
  if (n < 0) {
     warning('n cannot be negative')
     n = -n
  } else {
  fv = p * (1 + r)^n
  return(fv)
```

warning() does not stop
 execution but gives a
 warning message

```
# future value function
future_value <- function(p=1, r=0.01, n=1) {
   if (n < 0) {
      warning('n cannot be negative')
      n = -n
   }
   fv = p * (1 + r)^n
   fv
}</pre>
```

warning() does not stop execution but gives a warning message