Conditions and Loops

Estimated time needed: **15** minutes

Objectives

After completing this lab you will be able to:

- Understand and write R conditional statements
- Understand and write R loop statements

You will be working on this lab in cloud-based RStudio hosted by IBM Skills Network Labs.

Note that your lab will be reset after about 1 hour inactive window. It's recommended to backup the files you created.

About the Dataset

Imagine you got many movie recommendations from your friends and compiled all of the recommendations in a table, with specific info about each movie.

The table has one row for each movie and several columns

- `name` - The name of the movie
- `year` - The year the movie was released
- `length_min` - The lenght of the movie in minutes
- `genre` - The genre of the movie
- `average_rating` - Average rating on Imdb
- `cost_millions` - The movie's production cost in millions
- `sequences` - The amount of sequences
- `foreign` - Indicative of whether the movie is foreign (1) or domestic (0)
- `age_restriction` - The age restriction for the movie

You can see part of the dataset below

name	year	length_min	genre	average_rat
Toy Story	1995	81	Animation	
Akira	1998	125	Animation	
The Breakfast Club	1985	97	Drama	
The Artist	2011	100	Romance	
Modern Times	1936	87	Comedy	
Fight Club	1999	139	Drama	
City of God	2002	130	Crime	
The Untouchables	1987	119	Drama	
Star Wars	1977	121	Action	
American Beauty	1999	122	Drama	
Room	2015	118	Drama	
Dr. Strangelove	1964	94	Comedy	
The Ring	1998	95	Horror	
Monty Python and the Holy Grail	1975	91	Comedy	
High School Musical	2006	98	Comedy	
Shaun of the Dead	2004	99	Horror	
Taxi Driver	1976	113	Crime	
The Shawshank Redemption	1994	142	Crime	
Interstellar	2014	169	Adventure	
Casino	1995	178	Biography	
The Goodfellas	1990	145	Biography	
Blue is the Warmest Colour	2013	179	Romance	
Black Swan	2010	108	Thriller	
Back to the Future	1985	116	Sci-fi	
The Wave	2008	107	Thriller	
Whiplash	2014	106	Drama	
The Grand Hotel Budapest	2014	100	Crime	
Jumanji	1995	104	Fantasy	
The Eternal Sunshine of the Spotless Mind	2004	108	Drama	
Chicago	2002	113	Comedy	

First, let's identify your current working directory

- In the RStudio Console, run the following code snippet:

```
getwd()
```

then you should see the following result in console:

```
[1] "/resources/rstudio"
```

In the Files panel on the right, if you are not in `/resources/rstudio`, click `resources` folder and you should find a `rstudio` folder. This will be your current working directory in RStudio.

- Now run the following code in console to download the dataset:

```
# code to download the dataset
download.file("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-RP0101EN-Coursera/v2/dat
```

and you should see the dataset gets downloaded into the working directory as `movies-db.csv`

To begin, we can start by associating our data to a data frame. Let's call it `movies_Data`.
You will learn more about read csv file in other videos and labs.

- Copy/paste and run the following line into console

```
# code to download the dataset
movies_data <- read.csv("movies-db.csv", header=TRUE, sep=",")
```

Control statements

Control statements are ways for a programmer to control what pieces of the program are to be executed at certain times.

The syntax of control statements is very similar to regular english,
and they are very similar to logical decisions that we make all the time.

Conditional statements and loops are the control statements that are able to change the execution flow. The expected execution flow is that each line and command should be executed in the order they are written.

Control statements are able to change this,
allowing you to skip parts of the code or to repeat blocks of code.

Next, let's see how to write R conditional and loop statements.

Conditional Statements

We often want to check a conditional statement and then do something in response to that condition being true or false.

If statements

If statements are composed of a conditional check and a block of code that is executed if the check results is TRUE.

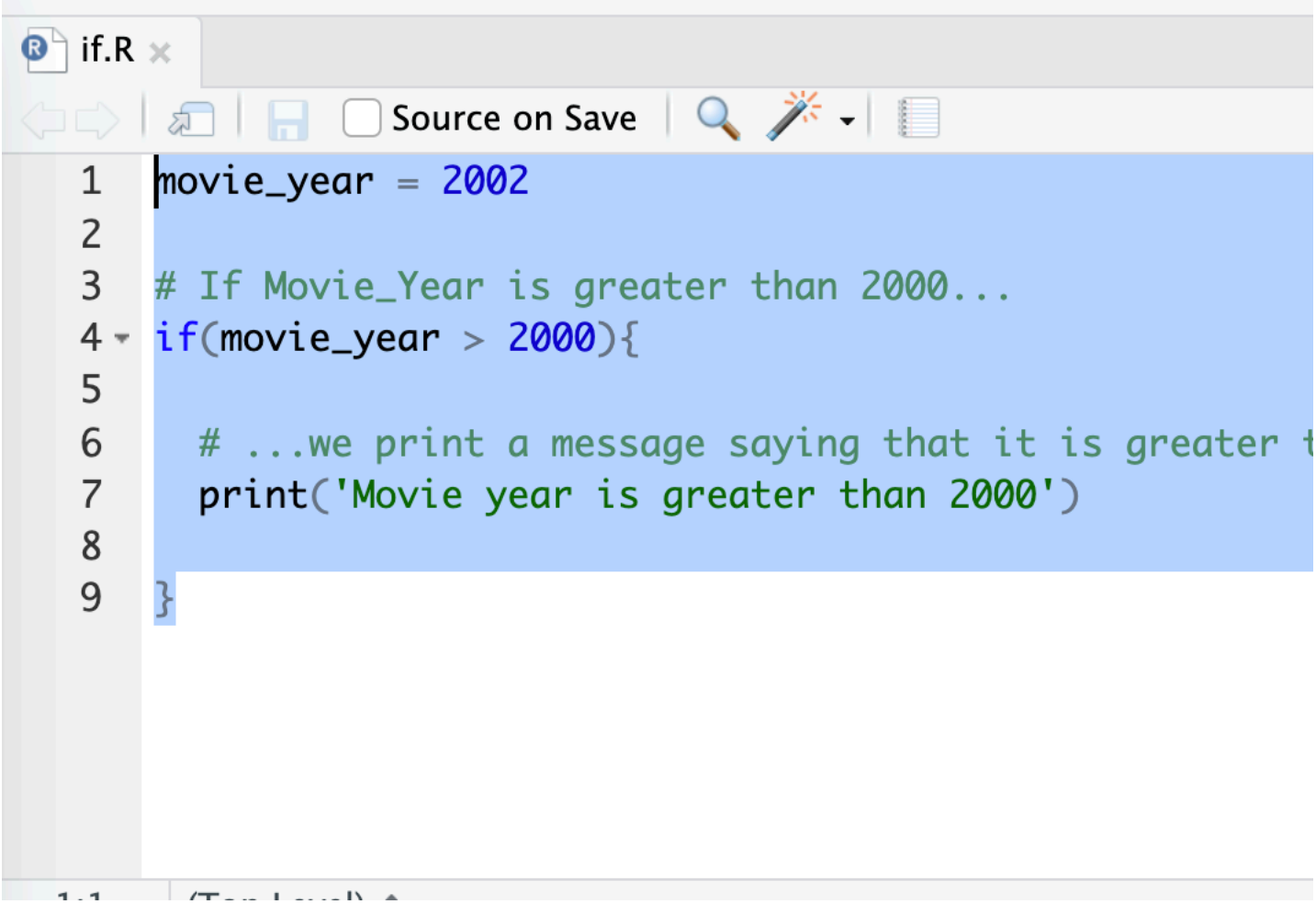
OK, let's start with `IF` statement

- From the RStudio top menu bar, select `File->New File->R Script` and save the file as `if.R`. You should see your saved file in the working directory (`resources/rstudio/`).
- In the `if.R` file, copy/paste the following code

```
movie_year = 2002
# If Movie_Year is greater than 2000...
if(movie_year > 2000){
    # ...we print a message saying that it is greater than 2000.
    print('Movie year is greater than 2000')
}
```

You need to make sure the last line is a new empty line so after copying the above code,
press the `Enter` key to start a new line in the script file.

Then use your mouse to select all the statements and click `Run the current line or selection` to run the selected lines.



Or click the `Source` icon to run the all lines in the `if.R` file

Above code assume we want to check a movie's year, and print something if it greater than 2000. So that you should see the following result in console.

```
[1] "Movie year is greater than 2000"
```

You can also add an `else` block to `if` block, the code in `else` block will only be executed if the check results in **false**.

Syntax:

```
if (condition) {
    # do something
} else {
    # do something else
}
```

TIPS: This syntax can be spread over multiple lines for ease of creation and legibility.

Let's create a variable called `movie_Year` and with a value 1997.
Additionally, let's add an `if` statement to check if the value stored in `movie_Year`
is greater than 2000 or not:

- If it is, then we want to output a message saying that `movie_Year` is greater than 2000,
- If not, then we output a message saying that it is not greater than 2000.
- Copy/paste the following code into `if.R`, select the copied lines and run them:

```
movie_year = 1997
# If Movie_Year is greater than 2000...
if(movie_year > 2000){
    # ...we print a message saying that it is greater than 2000.
    print('Movie year is greater than 2000')
}else{ # If the above conditions were not met (Movie_Year is not greater than 2000)...

    # ...then we print a message saying that it is not greater than 2000.
    print('Movie year is not greater than 2000')
}
```

If you didn't see the result, better to manually press `Enter` key to start a new line in the script file after copy/paste.

This time, you can see

```
[1] "Movie year is not greater than 2000"
```

Feel free to change `movie_year`'s value to other values – you'll see that the result changes based on it!

To create our conditional statements to be used with **`if`** and **`else`**, we
have a few operators:

Comparison operators

When comparing two values you can use this operators:

- equal: ==
- not equal: `!=`
- greater/less than: > <
- greater/less than or equal: >= <=

Logical operators

Sometimes you want to check more than one condition at once. For example you might want to check if one condition and other condition are true.

Logical operators allow you to combine or modify condition:

- and: &
- or: |
- not: `!`
- Let's try using these operators by copy/paste the following code snippet into `if.R` and run them

```
movie_year = 1997
# If Movie_Year is BOTH less than 2000 AND greater than 1990 -- both conditions have to be true! -- ...
if(movie_year < 2000 & movie_year > 1990 ) {
    # ...then we print this message.
    print('Movie year between 1990 and 2000')
}
# If Movie_Year is EITHER greater than 2010 OR less than 2000 -- any of the conditions have to be true! -- ...
if(movie_year > 2010 | movie_year < 2000 ) {
    # ...then we print this message.
    print('Movie year is not between 2000 and 2010')
}
```

and you should see `[1] "Movie year is not between 2000 and 2010"` shown in console.

Coding Exercise:: Check if the movie_year value is year 1998.

If not year 1998, print "Movie year is not 1998"

▶ Click here to see solution

TIPS: All the expressions will return the value in Boolean format – this format can only house two values: TRUE or FALSE!

Subset

Sometimes, we don't want an entire dataset,
maybe in a dataset of people we want only people with age 18 and over, or in the movies dataset,
maybe we want only movies that were created after a certain year.
This means we want a subset of the dataset.

In R, we can do this by utilizing the `subset` function.

Suppose we want a subset of the `movies_data` data frame composed of movies from a given year forward (e.g. year 2000)
if a selected variable is recent, or from that given year back if we select old.

We can quite simply do that in R using following code snippet:

```
decade = 'recent'
# If the decade given is recent...
if(decade == 'recent' ){
    # Subset the dataset to include only movies after year 2000.
    subset(movies_data, year >= 2000)
} else { # If not...
    # Subset the dataset to include only movies before 2000.
    subset(movies_data, year < 2000)
}
```

and you should see only movies after year 2000, i.e., the recent ones got printed in the console.

That's it for the control statement.

Before we learn loops statements, you can save and close `if.R` file and
use `Ctrl` or `Control` + L to clear the console,

Done? let's move on to R loops statements.

Loops

Sometimes, you might want to repeat a given function many times.
Maybe you don't even know how many times you want it to execute, but have an idea
like `once for every row in my dataset`.

Repeated execution like this is supplemented by loops.
In R, there are two main loop structures, `for` and `while`.

The `for` loop

Before we start, let's create another script file called `loop.R` to separate the code from `if.R`.

The `for` loop structure enables you to execute a code block once for every element in a given structure.
For example, it would be like saying execute this once for every row in my dataset,
or "execute this once for every element in this column bigger than 10".

`for` loops are a very useful structure that make the processing of a large amount of data very simple.

Let's try to use a `for` loop to print all the years present in the `year` column in the `movies_data` data frame.

- We can do that like this. Copy/paste the following lines into `loop.R` and run them:

```
# Get the data for the "year" column in the data frame.
years <- movies_data['year']
# For each value in the "years" variable...
# Note that "val" here is a variable -- it assumes the value of one of the data points in "years"!
for (val in years) {
    # ...print the year stored in "val".
    print(val)
}
```

you should see all the years in the `movies_data`

```
[1] 1995 1998 1985 2011 1936 1999 2002 1987 1977 1999 2015 1964 1998 1975 2006 2004 1976
[18] 1994 2014 1995 1990 2013 2010 1985 2008 2014 2014 1995 2004 2002
```

Coding Exercise:: Print every value in `length_min` column

▶ Click here to see solution

The `while` loop

As you can see, the `for` loop is useful for a controlled flow of repetition.
However, what if we don't know when we want to stop the loop? What if we want to keep executing a code block until a certain threshold has been reached, or maybe when a logical expression finally results in an expected fashion?

The `while` loop exists as a tool for repeated execution based on a condition.
The code block will keep being executed until the given logical condition returns a `False` boolean value.

Let's try using `while` to print the first five movie names of our dataset.

- It can be done like this:

```
# Creating a start point.
iteration = 1
# We want to repeat until we reach the sixth operation -- but not execute the sixth time.
# While iteration is less or equal to five...
while (iteration <= 5) {

    print(c("This is iteration number:",as.character(iteration)))

    # ...print the "name" column of the iteration-th row.
    print(movies_data[iteration,]$name)

    # And then, we increase the "iteration" value -- so that we actually reach our stopping condition
    # Be careful of infinite while loops!
    iteration = iteration + 1
}
```

Coding Exercise:: Print the "name" column of between 5th and 9th rows

▶ Click here to see solution

Applying Functions to Vectors

One of the most common uses of loops is to apply a given function to every element in a vector of elements.
Any of the loop structures can do that, however,
R conveniently provides us with a very simple way to do that: By inferring the operation.

R is a very smart language when it comes to element-wise operations on complex data structures.
For example, you can perform an operation on a whole list by utilizing that function directly on it.
Let's try that out in console:

```
# First, we create a vector...
my_list <- c(10,12,15,19,25,33)
# ...we can try adding two to all the values in that vector.
my_list + 2
# Or maybe even exponentiating them by two.
my_list ** 2
# We can also sum two vectors element-wise!
my_list + my_list
```

R makes it very simple to operate over vectors – anything you think should work will probably work.
Try to mess around with vectors and see what you find out!

Hopefully, now you know how to manipulate the flow of your code to your needs.

Thank you for finishing the lab, and good luck on your studies.

Author(s)

Hi! We are Helly Patel
and Walter Gomes,
the author of this lab. We hope you found R easy to learn!
There's lots more to learn about R but you're well on your way.
Feel free to connect with us if you have any questions.

Other Contributor(s)

Yan Luo