1 Files and the Filesystem

1.1 File Paths

```
/Users/christina/Documents/my_project/data.csv
/home/christina/my_project/data.csv
C:\Users\christina\Documents\my_project\data.csv
```

are examples of **Absolute File Paths**

```
../my_project/data/data.csv
data.csv
data/data.csv
```

are examples of **Relative File Paths**

The working directory is **the directory associated with a running process or program; it's where the computer will start looking for files when a relative path is given.**

1.1.1 Special Patterns

In file paths,

`..` means **go up one level in the file tree**

`~` means **home directory**

1.2 Working with Files

To retrieve information from a file, **read** from it. To put information into a file, **write** to it.

CSV (comma separated values) files are one type of plain text file. Plain text files have no **formatting** such as bold text, colors, or fonts.

R and Python expect that when data is stored in CSV (or tab-delimited) files, each observation is a **row** and each variable is a **column**. Rows and columns may or may not have names. Data is stored in a rectangle: each row has the same number of columns, and each column has the same number of rows.

2 Data Types

Common data types include:

- Boolean
- Integer
- Numeric
- Character

Boolean variables can be either **true** or **false**. When converted to an integer, **False** becomes 0 and **true** becomes 1.

Most languages also have special types such as `NULL` or `None` that indicate no value. These special types are different from missing value indicators (e.g. `NA`).

Character data is also known as **string** or **text** data.

3 Strings

Tabs, spaces, and new line characters are examples of **whitespace** characters.

`\n` is a **newline (return)**

`\t` is a **tab**

A string without any characters in it (length 0) is called an **empty** string.

Strings are sorted in alphabetical order. Lower case letters are different from upper case letters. The order of upper and lower case letters depends on the program.

Strings must be surrounded by **quotation marks**. In R and Python, single or double **quotes** can be used, but they must match. Pick one style and be consistent where possible!

`"north"` is a **substring** of `"northwestern"`.

Concatenating strings means to **join** the strings together.

4 Variables

Variables let us refer to a value with a name. We can use the same name, but change the value.

`<-` in R, and `=` in Python, are **assignment** operators. The name of the variable goes on the **left hand** side, and the value goes on the **right hand**. Everything on the right hand side is evaluated first before the value is assigned to the variable.

In R and Python, a variable with name `age_list` is the [**not the same**] is a variable with name `Age_list`.

If you run this code:

```
x = 3
x + 2
x
```

the value of x at the end will be **3**.

If you run this code:

```
x = 3
y = x
x = x + 1
y = y + 2
y
```

the value of y at the end will be **5**.

5 Lists, Vectors, Arrays

Vectors or **arrays** hold multiple values (usually) of the same type. **Lists** hold multiple values, possibly of different types.

Elements are stored in order, and elements can be referenced by their **index**. The first element has **an index** 0 or 1 depending on the language. The **length** of a list, vector, or array is the number of elements in it. An empty list has a **length** of 0.

You can **prepend** an item to the beginning of a list or vector or **append** an item to the end.

Sometimes, lists can be **nested** inside other lists.

In R and Python, you can take a slice of a list (or R vector) using the list indices:

```
my_list[a:b]
```

Example:

```
my_list[3:6]
```

In Python, `a` is the index of the **start** value, `b` is the index of the **end** value EXCLUSIVE (meaning it's not included).

In R, the first number is the index of the **start** value, and the second number is the index of the **end** value INCLUSIVE.

6 Assigning Values

Below, assume Python lists, which start with an index of 0.

To change the value of an element in a list, assign a new value to it:

```
my_list = [7, 6, 5, 4]
my_list[2] = 3
my_list
```

`my_list` now contains [**7, 6, 3, 4**].

If instead you assign a new value as:

```
my_list = [7, 6, 5, 4]
my_list = [1, 2, 3]
my_list
```

`my_list` now contains [**1, 2, 3**].

7 Conditions

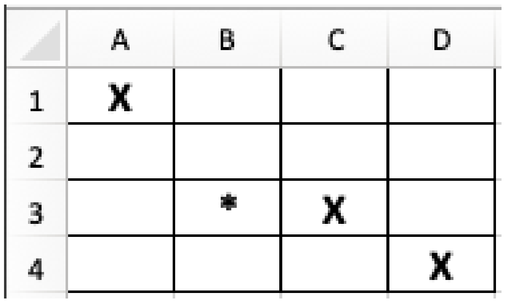
[**False**] When using variables with boolean values in a conditional statement, you should explicitly compare them to True or False to determine their value.

The operator to test for equality is `==`.

Is the following [**True**]; `(TRUE and FALSE) or (not FALSE and TRUE)`

8 Flow Control

If statements determine what to do based on a condition that evaluates to [**a single**] True or False value(s).



	A	B	C	D
1	X			
2				
3		*	X	
4				X

Figure 1:

Where will the * in cell B3 of Figure 1 above end up if you execute the following statements? **C1**

```
if the space to your right is occupied
  move one space up
  move one space right
else
  move one space right
if you are in column B
  move one space to your left
else if you are in column C
  move one space up
else
  move one space down
```

9 For Loops

Loops are used to **execute** the same code for **multiple** values.

The following code will print **4** numbers.

```
x = [1, 4, 3, 6, 7, 2]
for i in x
    if i < 5
        print i
```

10 Functions

The values you send to a function are called **arguments**, while the variables that are defined in a function definition are called **parameters**.

The **order** of non-keyword arguments must match the **order** of parameters in the function definition.

In Python, **keyword** arguments cannot come before **non-keyword** arguments. In R, it's more complicated!

You can [**always**] use the name with all of the parameters when calling a function.

It [**is**] good practice to specify arguments in a function call in the order they appear in the function definition, regardless of whether you're using named/keyword parameters or not.

Parameters without default values in a function definition are [**required**].

It's [**OK**] to have variables in your script with the same name as function parameters.

The output of a function is called the **return** value.

11 Packages/Libraries

Packages/libraries/modules need to be **loaded** before using them in every script or session. Some are built-in, while others need to be **installed** first.

It's a [**good**] idea to use packages written by other people.

12 Ways to Execute Code

When working interactively in the **console**, each input line starts with a **prompt**, which may look like `>`, `>>>`, or `$` (or something else entirely) with a space after it. After typing input, hit return to **execute** the code. When the code is done executing, any output will be printed, and the command prompt will appear again at the start of a new line.

You can also write a **script**: a file with many lines of code in it to be executed together. They can be run from within your Integrated Development Environment (IDE) such as RStudio or Spyder, or from the command line.