Exercise 1

Formally, nothing is wrong with the function call. It is valid R code. However, as described in the presentation using absolute file names causes many problems as soon as you want to move your project to another computer or even to another place on the same computer. Moreover, the project folder is called `patency index`. It contains a space. This is not optimal, even though it would work in this example. A better way to write the code (assuming that the project folder has been renamed accordingly) is:

```
list.files("./data/")
```

Here, the period `.` refers to the current working directoy, which should always be the root folder of your project.

Exercise 2

An interpreted language is a language that is executed by an interpreter. The interpreter is a program that reads and analyzes the source code and then executes the commands bit by bit. A compiled language in contrast is first translated by a compiler into machine code (resulting in e.g. an `.exe` file on windows) which can be directly executed by the CPU. This comes with some pros and cons for both system:

- Interpreters usually exist for all computer systems. This means that you could share your script with someone working on a completely different system, i.e. interpreted languages are usually more portable. Compiled languages are less portable. Even though compilers exist for almost all systems as well you usually have to make many changes to your code in order to make it executable on a different system.
- Compilers produce stand-alone files (e.g. `.exe` on Windows), which can then be shared with other people. These people do not need the compiler or anything else. They could just execute the file (given they are using the same computer system). If you share the script of an interpreted language like R or Python the other person would have to install the interpreter in order to be able to execute the script.
- Since compilers produce machine code that is directly executed by the CPU the resulting program is considerably faster than interpreted languages, which always have the interpreter as an intermediate. C or Fortran, which are considered as the fastes languages, are magnitudes faster than R or Python.
- Developing a program in a compiled language is usually a pain in the ass. The code is less readable by humans and usually much longer for doing the same thing. Also,

compilers do not forgive any formal mistakes in your code. Also, after each change you would have to re-compile your code, which takes time, before you could see any results. Interpreter usually only throw errors when a mistake becomes relevant. Additionally, you can simply write something on the command line, send it to the interpreter, and directly see the result—unimaginable for compiled languages. Therefore, it is easier for beginners to learn programming by using an interpreted language.

- Compilers produce machine code, which is unreadable for humans. Thus, if you have an outstanding idea for a program and you do not want others to see how your program is operating, you can write it in a compiled language.
- A more philosophical problem is that interpreter are not self-sustaining. Somewhere at the bottom you need machine code and that was produced by a compiler. For instance, the standard interpreter of R is written in C.