**Reading User Input**

readinteger <- function()

{

n <- readline(prompt="Enter an integer: ")

return(as.integer(n))

}

print(readinteger())

Enter an integer: 88

[1] 88

**The readline function**

readline() lets the user enter a one-line string at the terminal.   
The prompt argument is printed in front of the user input. It usually ends on ": ".

**The as.integer function**

as.integer makes an integer out of the string.

**Preventing failure if no number is entered**

Right now if the user doesn't enter an integer, as.integer will return NA (Not Available). We can avoid this by using is.na to check the user input and asking again if the value is NA:

readinteger <- function()

{

n <- readline(prompt="Enter an integer: ")

n <- as.integer(n)

if (is.na(n)){

n <- readinteger()

}

return(n)

}

print(readinteger())

Enter an integer:

Enter an integer: boo

Enter an integer: 44

[1] 44

Warning message:

In readinteger() : NAs introduced by coercion

However, a warning message is still shown. This is how to avoid it:

readinteger <- function()

{

n <- readline(prompt="Enter an integer: ")

if(!grepl("^[0-9]+$",n))

{

return(readinteger())

}

return(as.integer(n))

}

print(readinteger())

Enter an integer:

Enter an integer: 31r132weq

Enter an integer: effasdf

Enter an integer: 222

[1] 222

# getwd, setwd

R is always pointed at a directory on your computer. You can find out which directory by running the getwd (get working directory) function; this function has no arguments. To change your working directory, use setwd and specify the path to the desired folder.

getwd()

setwd(dir)

* dir – Specify a working directory. This may be from the root directory (starting with / on a Mac), it may include a double-dot (..) to move locally up a folder from the current directory, and it may include a path from the current directory.

Example. In the first block of code, I verify there is nothing in the work space, I check the current directory, and I specify a directory I want to go to (getting to my Videosfolder), navigate further to the subfolder packageFiles. Finally, I source in a file from this folder, which creates the lmPlot function in my workspace. The last two commands show how easy it is to move up one folder.

> ls()

character(0)

> getwd()

[1] "/Users/ddiez"

> setwd("/Users/ddiez/Dropbox/rFunction/Videos")

> setwd("packageFiles")

> getwd()

[1] "/Users/ddiez/Dropbox/rFunction/Videos/packageFiles"

>

> source("lmPlot.R")

> ls()

[1] "lmPlot"

>

> setwd("..")

> getwd()

[1] "/Users/ddiez/Dropbox/rFunction/Videos"