Common warnings and errors

Steve Paterson

14/8/20

Errors vs warnings

Both errors and warnings result in red text output to the console. If you see the word 'Error: …' it means that R has tried to run a command and failed. If you see 'Warning: …', R has run a command but is telling you it might not have worked quite as you expect and you should check the output. For example, it might give some missing values or similar in the output, but it's not completely failed.

An example of each

```
log("sheep")
```

```
## Error in log("sheep"): non-numeric argument to mathematical function
```

Here we've tried to pass a string object to the `log()` function, which is a nonsense and R throws an error.

In the example below, R warns us with red text that something may not have worked as you expected but it's still completed the function. It's an indication to check the output for missing values or similar. Here a log of a negative number is reported as NaN (not a number) and a log of 0 as negative infinity.

```
log(c(-10,0,10,100))
```

```
## Warning in log(c(-10, 0, 10, 100)): NaNs produced
```

```
## [1]      NaN     -Inf 2.302585 4.605170
```

(Remember that R uses natural logs by default not log10)

Loading data

This will definitely go wrong for you at some point. The most common problem is R giving an error that it can't find the file (or 'open the connection' in R error-speak).

```
wont.work <- read.table("no_file.txt")
```

```
## Warning in file(file, "rt"): cannot open file 'no_file.txt': No such file or
## directory
```

```
## Error in file(file, "rt"): cannot open the connection
```

I've given a video on how to solve this. Basically first find the file in the files pane of RStudio. Then use *Session* | *Set working directory* | *To files pane location* to tell R where to look (this calls the `setwd()` command). If it still gives the same error, check that you've spelt the file name correctly. Type `dir()` and either check the spelling (which may be case sensitive). Cut and paste the filename from the terminal into your command if needed. If you still get the same error ('can't open the connection'), some paths on the University system may not be available for read/write by RStudio.

The second problem that comes up is that files are not read in correctly. This may because the header (the first line of the file) is/isn't set to be the column names. The delimiter character, which separates entries in a row, may be a comma when R expects a space or a tab.

There may be unequal numbers of entries for some rows. This is often difficult to spot, but can arise from more than one delimiter between entries, spaces within entries, or having some entries contain quotes or apostrophes. Specifying a delimiter explicitly might help, eg a tab ("\t") to stop any whitespace being read as a delimiter.

R gets stuck

You might enter a command in the console, hit return but nothing happens. Some commands take a while to finish if they need a lot of calculation. Most commands in this module don't need that much calculation. A more common reason is that R thinks you're still writing a command in. You can tell this becaiuse instead of a '>' you see a '+' at the start of a line in the console. The usual reason is that an open bracket ("(") isn't followed by a closed bracket (")"), or a quoted string is missing the end quote.

Try this (see help page for `cat()` to see what it does):

```
cat("Hello world")
```

```
## Hello world
```

```
cat("Hello world
")
```

```
## Hello world
```

R allows you to split commands over lines and it won't think you've finished until the brackets and quotes have been finished. In RStudio, entering the commands first into the text editor pane will flag up unmatched brackets and quotes. If you get stuck in the console you can hit escape and try again.

R can't find a function

A common complaint by R is that a particular function (command) isn't found. Check the spelling carefully, remembering that it's case sensitive. It may also be that you haven't told R that you want to use the package that function is in. You do first this using `library("packagename")`. R should then be able to find the function.