Chapter 2

Ian Flores Siaca

2019-09-03

2.2.2 Exercises

1) Explain the relationship between a, b, c and d in the following code:

```
a <- 1:10
b <- a
c <- b
d <- 1:10
```

a is creating a name binding for a vector from 1 to 10. b is pointing to the same vector as a but it is not copying it. c is pointing to the same vector as b & a. d is creating a new vector in memory. We can confirm this using `lobstr`:

```
lobstr::obj_addr(a)
```

```
## [1] "0x5591c7d05bc0"
```

```
lobstr::obj_addr(b)
```

```
## [1] "0x5591c7d05bc0"
```

```
lobstr::obj_addr(c)
```

```
## [1] "0x5591c7d05bc0"
```

```
lobstr::obj_addr(d)
```

```
## [1] "0x5591c926baa0"
```

2) The following code accesses the mean function in multiple ways. Do they all point to the same underlying function object?

They all point to the same underlying function.

```
lobstr::obj_addr(mean)
```

```
## [1] "0x5591c9aa4008"
```

```
lobstr::obj_addr(base::mean)
```

```
## [1] "0x5591c9aa4008"
```

```
lobstr::obj_addr(get("mean"))
```

```
## [1] "0x5591c9aa4008"
```

```
lobstr::obj_addr(evalq(mean))
```

```
## [1] "0x5591c9aa4008"
```

```
lobstr::obj_addr(match.fun("mean"))
```

```
## [1] "0x5591c9aa4008"
```

3) By default, base R data import functions, like read.csv(), will automatically convert non-syntactic names to syntactic ones. Why might this be problematic? What option allows you to suppress this behaviour?

   This can be problematic as we can expect data corruption to occur if the the values of certain variables are not translated correctly once the data is opened in R. If we change the `check.names` default value to FALSE we can supress this behavior at least for the column names.

4) What rules does make.names() use to convert non-syntactic names into syntactic ones?

   First it will add the X character. Then it will translate all invalid characters to ".". Missing values are translated to "NA". Names thar match R keywords will have a "." appended.

5) I slightly simplified the rules that govern syntactic names. Why is .123e1 not a syntactic name? Read ?make.names for the full details.

   .123e1 is not a syntactic names because it's a name that starts with a dot followed by a number. If the syntaxis was .e123 it would have been a valid syntactic name.

2.3.6 Exercises

1) Why is tracemem(1:10) not useful?
2) Explain why tracemem() shows two copies when you run this code. Hint: carefully look at the difference between this code and the code shown earlier in the section.

```
x <- c(1L, 2L, 3L)
tracemem(x)
```

```
## [1] "<0x5591c8fa7b28>"
```

```
x[[3]] <- 4
```

```
## tracemem[0x5591c8fa7b28 -> 0x5591c9ffbb98]: eval eval withVisible withCallingHandlers handle timing_
## tracemem[0x5591c9ffbb98 -> 0x5591cadfd498]: eval eval withVisible withCallingHandlers handle timing_
```

3) Sketch out the relationship between the following objects:

```
a <- 1:10
b <- list(a, a)
c <- list(b, a, 1:10)
```

4) What happens when you run this code?

```
x <- list(1:10)
x[[2]] <- x
```