Data Science with R
Environments

Graham.Williams@togaware.com

19th July 2014

Visit http://onepager.togaware.com/ for more OnePageR's.

The advanced topic of R environments are covered in this chapter.

The required packages for this chapter include:

As we work through this chapter, new R commands will be introduced. Be sure to review the command's documentation and understand what the command does. You can ask for help using the `?` command as in:

```
?read.csv
```

We can obtain documentation on a particular package using the *help=* option of `library()`:

```
library(help=rattle)
```

This chapter is intended to be hands on. To learn effectively, you are encouraged to have R running (e.g., RStudio) and to run all the commands as they appear here. Check that you get the same output, and you understand the output. Try some variations. Explore.

1 Environment Basics

R references all objects within environments. An environment itself is an object in R. It is a special object in that it consists of a frame and a link to its enclosing environment. The frame is a collection of named objects, and it is within the frame where the object lives.

We can easily create one using `new.env()`:

```
my.env = new.env()
my.env
```

```
## <environment: 0x10c28a8>
```

The hexadecimal code listed here is the address of the environment in the computers memory. It is where the data is actually stored in memory and is used to access the data.

Environments are contained within other environments, called the enclosure, and identified as the parent environment. The R manual page suggests avoiding confusion by not calling the enclosing environment the parent environment—it is easily confused with the concept of a parent frame. We do use `parent.env()` to identify the enclosing environment though!

```
parent.env(my.env)
```

```
## <environment: R_GlobalEnv>
```

Every environment has an enclosure except for one environment, which is the empty environment `R_EmptyEnv`.

```
emptyenv()
```

```
## <environment: R_EmptyEnv>
```

```
parent.env(emptyenv())
```

```
## Error: the empty environment has no parent
```

We note that `my.env`'s parent is `R_GlobalEnv`. This is the global environment and is commonly referred to as the user's workspace.

```
.GlobalEnv
```

```
## <environment: R_GlobalEnv>
```

```
globalenv()
```

```
## <environment: R_GlobalEnv>
```

By default, the enclosure of any environment created using `new.env()` is the current environment. We can override this.

```
your.env <- new.env()
parent.env(your.env)
```

```
## <environment: R_GlobalEnv>
```

```
parent.env(your.env) <- my.env
parent.env(your.env)
```

```
## <environment: 0x10c28a8>
```

2 Enclosures

The enclosure of the global environment is often the stats package, named `package:stats`.

```
parent.env(parent.env(my.env))
```

```
## <environment: package:stats>
## attr(,"name")
## [1] "package:stats"
## attr(,"path")
## [1] "/usr/lib/R/library/stats"
```

It is not always the case. Within the production of this document that you are currently reading, whilst the code is being run by the knitr package, the enclosure is the xtable package!

```
parent.env(parent.env(my.env))
```

```
## <environment: package:xtable>
## attr(,"name")
## [1] "package:xtable"
## attr(,"path")
## [1] "/usr/lib/R/site-library/xtable"
```

When running R within the ESS package for Emacs we see the following:

```
parent.env(parent.env(my.env))
```

```
## <environment: 0x383ef98>
## attr(,"name")
## [1] "ESSR"
```

And within RStudio we see:

```
parent.env(parent.env(my.env))
```

```
## <environment: 0x3745410>
## attr(,"name")
## [1] "tools:rstudio"
```

Generally the next level of enclosure is the stats package.

```
parent.env(parent.env(globalenv()))
```

```
## <environment: package:knitr>
## attr(,"name")
## [1] "package:knitr"
## attr(,"path")
## [1] "/usr/lib/R/site-library/knitr"
```

3 Naming Environments

From the various environments we have just seen, we might notice that environments can have a name. The name is stored as an attribute of the objects. We access the name of the environment using `attr()` and can set the name with assignment to this function:

```
attr(my.env, "name")
```

```
## NULL
```

```
attr(my.env, "name") <- "MyEnv"
attr(my.env, "name")
```

```
## [1] "MyEnv"
```

```
my.env
```

```
## <environment: 0x10c28a8>
## attr(,"name")
## [1] "MyEnv"
```

We can also retrieve the environment's name using `environmentName()`:

```
environmentName(my.env)
```

```
## [1] "MyEnv"
```

```
environmentName(globalenv())
```

```
## [1] "R_GlobalEnv"
```

```
environmentName(emptyenv())
```

```
## [1] "R_EmptyEnv"
```

4 Objects within Environments

In R we create objects through assignment. Here we create a new named object, called `m`, and store as the value of the object the numeric value 42.

```
m <- 42
```

This object is automatically placed into the current environment. By default the current environment is the global environment, which we noted above is also referred to as the user's workspace. We can check which environment is current using `environment()`:

```
environment()
```

```
## <environment: R_GlobalEnv>
```

To identify the contents of the frame of an environment we use `ls()`:

```
ls()
```

```
## [1] "hook_output" "hook_source" "m"           "Module"      "my.env"
## [6] "start.time"  "your.env"
```

By default `ls()` lists the contents of the current environment. We can specify a particular environment using the optional `name=` argument:

```
ls(name=my.env)
```

```
## character(0)
```

```
ls(my.env)
```

```
## character(0)
```

```
ls(globalenv())
```

```
## [1] "hook_output" "hook_source" "m"           "Module"      "my.env"
## [6] "start.time"  "your.env"
```

```
ls("package:stats")
```

```
##   [1] "acf"                   "acf2AR"                "add1"
##   [4] "addmargins"            "add.scope"             "aggregate"
##   [7] "aggregate.data.frame" "aggregate.ts"          "AIC"
##  [10] "alias"                 "anova"                 "ansari.test"
....
```

```
ls(parent.env(parent.env(environment())))
```

```
##  [1] "all_labels"         "all_patterns"       "asis_output"
##  [4] "current_input"      "dep_auto"           "dep_prev"
##  [7] "eclipse_theme"      "fig_path"           "hook_ffmpeg_html"
## [10] "hook_movecode"      "hook_optipng"       "hook_pdfcrop"
....
```

5 Adding Objects to Environments

To add an object to a specific environment we use `assign()`:

```
ls(my.env)
```

```
## character(0)
```

```
assign("n", c("a", "b"), envir=my.env)
ls(my.env)
```

```
## [1] "n"
```

Notice that the object is not in the current, global environment:

```
ls()
```

```
## [1] "hook_output" "hook_source" "m"           "Module"      "my.env"
## [6] "start.time"  "your.env"
```

```
n
```

```
## Error:  object 'n' not found
```

To retrieve the value of this object from the correct environment we use `get()`:

```
get("n", envir=my.env)
```

```
## [1] "a" "b"
```

6 Environments within Functions

On invoking a function, a new environment is created, enclosed by the current environment. That environment becomes the current environment whilst the function is executing.

```
myFun <- function(x)
{
  y <- 1
  print(environment())
  print(ls())
  print(parent.env(environment()))
}

environment()
```

```
## <environment: R_GlobalEnv>
```

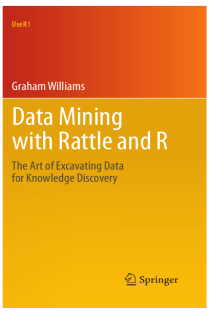
```
myFun()
```

```
## <environment: 0x1603c50>
## [1] "x" "y"
## <environment: R_GlobalEnv>
```

This is useful to ensure that what we do within a function stays within the function and objects created within the function are removed at the termination of the function, and do not overwrite objects of the same name outside of the function.

7 Further Reading and Acknowledgements

The Rattle Book, published by Springer, provides a comprehensive introduction to data mining and analytics using Rattle and R. It is available from Amazon. Other documentation on a broader selection of R topics of relevance to the data scientist is freely available from `http://datamining.togaware.com`, including the Datamining Desktop Survival Guide.



This module is one of many OnePageR modules available from `http://onepager.togaware.com`. In particular follow the links on the website with a * which indicates the generally more developed OnePageR modules.

8 References

R Core Team (2014). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

Williams GJ (2009). "Rattle: A Data Mining GUI for R." *The R Journal*, **1**(2), 45–55. URL http://journal.r-project.org/archive/2009-2/RJournal_2009-2_Williams.pdf.

Williams GJ (2011). *Data Mining with Rattle and R: The art of excavating data for knowledge discovery*. Use R! Springer, New York. URL http://www.amazon.com/gp/product/1441998896/ref=as_li_qf_sp_asin_tl?ie=UTF8&tag=togaware-20&linkCode=as2&camp=217145&creative=399373&creativeASIN=1441998896.

Xie Y (2014). *knitr: A general-purpose package for dynamic report generation in R*. R package version 1.6, URL http://CRAN.R-project.org/package=knitr.

This document, sourced from EnvironmentsO.Rnw revision 460, was processed by KnitR version 1.6 of 2014-05-24 and took 1.5 seconds to process. It was generated by gjw on nyx running Ubuntu 14.04 LTS with Intel(R) Xeon(R) CPU W3520 @ 2.67GHz having 4 cores and 12.3GB of RAM. It completed the processing 2014-07-19 10:27:24.