8.1 Environments

Kira Lai

2024-11-15

Environments

Your computer arranges files into a hierarchy of folders and subfolders. To look at a file, you need to find where it is saved in the file system.

R uses a similar system to save R objects. Each object is saved inside of an environment, a list-like object that resembles a folder on your computer. Each environment is connected to a parent environment, a higher-level environment, which creates a hierarchy of environments.

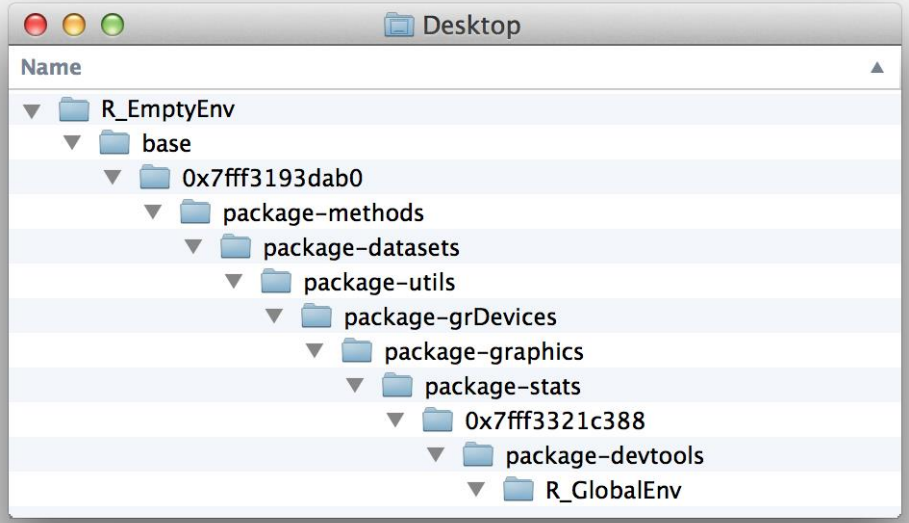
Viewing R's environment

You can see R's environment system with the parenvs function in the pryr package

```
library(pryr)
parenvs(all = TRUE)
##    label                                name
## 1  <environment: R_GlobalEnv>           ""
## 2  <environment: package:pryr>          "package:pryr"
## 3  <environment: package:stats>         "package:stats"
## 4  <environment: package:graphics>      "package:graphics"
## 5  <environment: package:grDevices>     "package:grDevices"
## 6  <environment: package:utils>         "package:utils"
## 7  <environment: package:datasets>      "package:datasets"
## 8  <environment: package:methods>       "package:methods"
## 9  <environment: 0x0000020cfe1634c8>    "Autoloads"
## 10 <environment: base>                  ""
## 11 <environment: R_EmptyEnv>            ""
```

Environment tree

- The lowest-level environment is named `R_GlobalEnv` and is saved inside an environment named `package:pryr`, which is saved inside the environment named `0x7fff3321c388`, and so on, until you get to the final, highest-level environment, `R_EmptyEnv`.
- `R_EmptyEnv` is the only R environment that does not have a parent environment.



Working with environments

R comes with some helper functions that you can use to explore your environment tree.

```
as.environment("package:pryr")
## <environment: package:pryr>
## attr(,"name")
## [1] "package:pryr"
## attr(,"path")
## [1] "C:/Users/laiki/AppData/Local/R/win-
library/4.4/pryr"
```

Access an object in a specific environment

You can use R's `$` syntax to access an object in a specific environment. For example, you can access deck from the global environment:

```
deck <- data.frame(face = c("ace", "two", "six"),
                   suit = c("clubs", "clubs", "clubs"),
value = c(1, 2, 3),
                   stringsAsFactors = FALSE)
head(globalenv()$deck, 3)
##   face  suit value
## 1  ace clubs     1
## 2  two clubs     2
## 3  six clubs     3
```

Save an object into a particicular environment

1. Give assign the name of the new object
2. Give assign the value of the new object
3. The environment to save the object in: Works similarly to <-

```
assign("new", "Hello Global", envir =
globalenv())

globalenv()$new
## [1] "Hello Global"
```

The Active Environment

Use `environment()` to see the current active environment

```
environment()
## <environment: R_GlobalEnv>
```

- The global environment is the active environment for every command that you run at the command line. You can think of the global environment as your user workspace.
- R will search for an object by name in the active environment, here the global environment. If R does not find the object there, it will search in the active environment's parent, and then the parent's parent, and so on until R finds the object or runs out of environments.

Assignment

When you assign a value to an object, R saves the value in the active environment under the object's name. If an object with the same name already exists in the active environment, R will overwrite it.

```
new
## [1] "Hello Global"
new <- "Hello Active"

new
## [1] "Hello Active"
```

Assignment in functions

Many functions save temporary objects that help them do their jobs. For example, the `roll` function from Project 1: Weighted Dice saved an object named die and an object named dice:

```
roll <- function() {
  die <- 1:6
  dice <- sample(die, size = 2, replace = TRUE)
  sum(dice)
}
```

R must save these temporary objects in the active environment; but if R does that, it may overwrite existing objects. Therefore, every time R runs a function, it creates a new active environment to evaluate the function in.

Evaluation

R will use the new environment as the active environment while it runs the function, and then R will return to the environment that you called the function from, bringing the function's result with it. `show_env` is designed to illustrate how R saves the temporary data in a new environment

```
show_env <- function(){
  list(ran.in = environment(),
    parent = parent.env(environment()),
    objects = ls.str(environment()))
}
show_env()
## $ran.in
## <environment: 0x0000020d03c30ed8>
##
## $parent
## <environment: R_GlobalEnv>
##
## $objects
```

Evaluation

Here `deal` saves (and then returns) the top card of deck. In between, Instead of overwriting the global copy of `deck` with `deck[-1, ]`, `deal` will just create a slightly altered copy of `deck` in its runtime environment

```
deal <- function() {
  card <- deck[1, ]
  deck <- deck[-1, ]
  card
}
deal()
##   face  suit value
## 1  ace clubs     1
```

Overwrite the deck

Rewrite the `deck <- deck[-1, ]` line of deal to assign `deck[-1, ]` to an object named deck in the global environment using the `assign` function

- Using the `assign` function, you can assign an object to a specific environment:

```
deal <- function() {
  card <- deck[1, ]
  assign("deck", deck[-1, ], envir = globalenv())
  card
}

deal()
##   face  suit value
## 1  ace clubs     1
deal()
##   face  suit value
## 2  two clubs     2
deal()
##   face  suit value
## 3  six clubs     3
deal()
##    face suit value
## NA <NA> <NA>    NA
```

Summary

- R saves its objects in an environment system that resembles your computer's file system.
- When you call an object at the command line, R will look for object in the global environment and then the parents of the global environment.
- R uses a different search path when you call an object from inside a function. In which it uses the child environment.