Calling functions in R usually involves typing brackets. And since many of our actions in R involve calling a function, we will have to type a lot of brackets working with R. Often it would make our life a lot easier if we could omit the need to type brackets where convenient. We will do exactly that today.

**Work in R faster with custom bracketless commands**

A good starting example is, well, quitting R altogether. Usually, one may do:

quit()

Which will in turn likely get you and extra question regarding saving a workspace image. So you then finally type n and are done with it. If you want to be a bit faster, you may do:

q("no")

Better, but still an awful lot of typing just to quit R, especially when working in a terminal-like environment with multiple sessions.

Let us be a bit craftier and make R quit just by typing qq

To make a bracketless command, we will (mis)use the fact that typing an object name into R console and pressing enter will often invoke a print method specific for the class of that object.

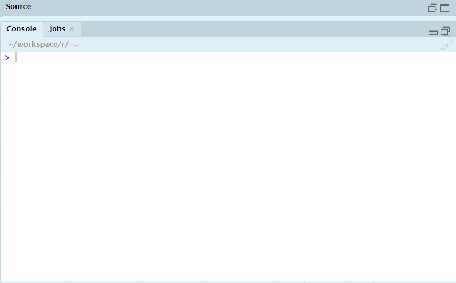
All we have to do to create our very first bracketless command is to create a custom print method for a funky class made for this single purpose. Then we make an object of that class and type its name to the console:

qq <- structure("no", class = "quitter")

print.quitter <- function(quitter) base::quit("no")

# This will quit your session NOT saving a workspace image!

qq



Oopsâ€¦I Did It Again

**Switching debugging modes with ease**

Quitting R quickly is more useful then it may sound when using multiple sessions in a terminal environment, but we can use the above approach to create different useful shortcuts making our life much easier.

One example I use very frequently is to change the error option, which governs how R behaves when encountering non-catastrophic errors such as those generated by stop, etc.

* I find setting the option to options(error = utils::recover) very useful for debugging and at the same time very annoying when undesired.
* Typing options(error = NULL) to change it back is however even more annoying. Or is it options("error") = NULL? Or maybe even options(error) = NULL?

In comes the gg shortcut:

gg <- structure(FALSE, class = "debuggerclass")

print.debuggerclass <- function(debugger) {

if (!identical(getOption("error"), as.call(list(utils::recover)))) {

options(error = recover)

message(" \* debugging is now ON - option error set to recover")

} else {

options(error = NULL)

message(" \* debugging is now OFF - option error set to NULL")

}

}

Now we switch between the options with ease:

# When in need of debugging

gg

## \* debugging is now ON - option error set to recover

# The option is now set to recover

getOption("error")

## (function ()

## {

## if (.isMethodsDispatchOn()) {

## tState <- tracingState(FALSE)

## on.exit(tracingState(tState))

## }

## calls <- sys.calls()

## from <- 0L

## n <- length(calls)

## if (identical(sys.function(n), recover))

## n <- n - 1L

## for (i in rev(seq\_len(n))) {

## calli <- calls[[i]]

## fname <- calli[[1L]]

## if (!is.na(match(deparse(fname)[1L], c("methods::.doTrace",

## ".doTrace")))) {

## from <- i - 1L

## break

## }

## }

## if (from == 0L)

## for (i in rev(seq\_len(n))) {

## calli <- calls[[i]]

## fname <- calli[[1L]]

## if (!is.name(fname) || is.na(match(as.character(fname),

## c("recover", "stop", "Stop")))) {

## from <- i

## break

## }

## }

## if (from > 0L) {

## if (!interactive()) {

## try(dump.frames())

## cat(gettext("recover called non-interactively; frames dumped, use debugger() to view\n"))

## return(NULL)

## }

## else if (identical(getOption("show.error.messages"),

## FALSE))

## return(NULL)

## calls <- limitedLabels(calls[1L:from])

## repeat {

## which <- menu(calls, title = "\nEnter a frame number, or 0 to exit ")

## if (which)

## eval(substitute(browser(skipCalls = skip), list(skip = 7 -

## which)), envir = sys.frame(which))

## else break

## }

## }

## else cat(gettext("No suitable frames for recover()\n"))

## })()

# When done debugging

gg

## \* debugging is now OFF - option error set to NULL

# The option is now back to NULL

getOption("error")

## NULL

**Making it practical (and a bit less barbaric)**

Defining all the shortcuts in the way shown above every time is both tedious and ugly, making a mess in our global environment. We can therefore decrease the tedium and ugliness by:

1. Adding the definitions into our .Rprofile with a proper notice, which will run the definitions and make the shortcuts available every time we start R standardly
2. Enclosing the definitions into a separate environment attached to the search path, potentially with a command to detach it easily

Such an .Rprofile can look similar to:

message("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

message("| |")

message("| SOURCING CUSTOM .Rprofile |")

message("| |")

message("| \* qq => quit('no') |")

message("| \* gg => toggle error = recover/NULL |")

message("| \* dd => detach this madness |")

message("|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|")

message("\n")

customCommands <- new.env()

assign("qq", structure("no", class = "quitterclass"), envir = customCommands)

assign("print.quitterclass", function(quitter) {

message(" \* quitting, not saving workspace")

base::quit(quitter[1L])

}, envir = customCommands)

assign("gg", structure("", class = "debuggerclass"), envir = customCommands)

assign("print.debuggerclass", function(debugger) {

if (!identical(getOption("error"), as.call(list(utils::recover)))) {

options(error = recover)

message(" \* debugging is now ON - option error set to recover")

} else {

options(error = NULL)

message(" \* debugging is now OFF - option error set to NULL")

}

}, envir = customCommands)

assign("dd", structure("", class = "detacherclass"), envir = customCommands)

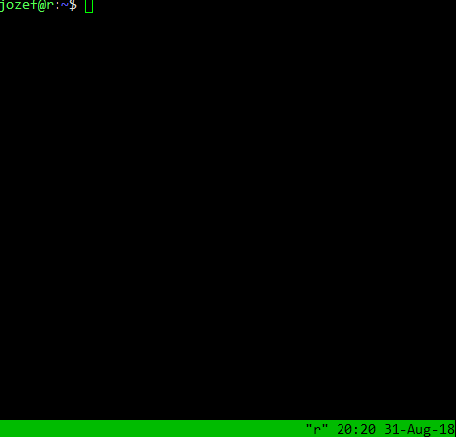
assign("print.detacherclass", function(detacher) {

detach(customCommands, unload = TRUE, force = TRUE)

})

attach(customCommands)

In terminal environments, shortcuts like this can be even more useful:



Tends to be more useful in the terminal