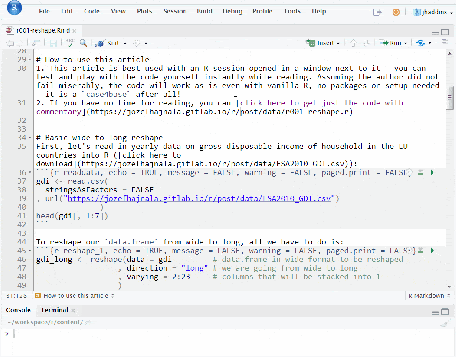
**Introduction**

In this post in the RStudio:addins series we will try to make our work more efficient with an addin for better inspection of objects, functions and files within RStudio. RStudio already has a very useful View function and a Go To Function / File feature with F2 as the default keyboard shortcut and yes, I know I promised automatic generation of @importFrom roxygen tags in the [previous post](https://jozefhajnala.gitlab.io/r/r102-addin-roxytags), unfortunately we will have to wait a bit longer for that one but I believe this one more than makes up for it in usefulness.

**The addin we will create in this article will let us use RStudio to View and inspect a wide range of objects, functions and files with 1 keypress.**



The addins in action

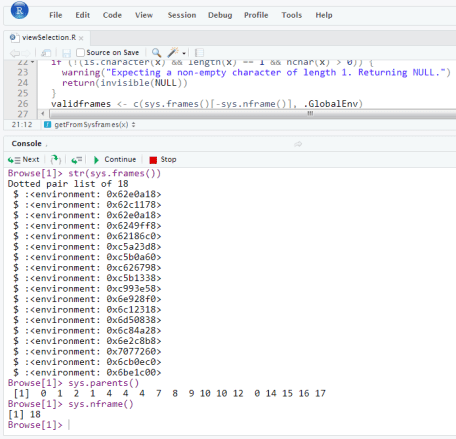
**Contents**

1. [Retrieving objects from sys.frames](https://jozefhajnala.gitlab.io/r/r103-keypress-viewer/#retrieving-objects-from-sys.frames)
2. [Viewing files, objects and functions and more efficiently](https://jozefhajnala.gitlab.io/r/r103-keypress-viewer/#viewing-files-objects-functions-and-more-efficiently)
3. [The addin function, updating the .dcf file and key binding](https://jozefhajnala.gitlab.io/r/r103-keypress-viewer/#the-addin-function-updating-the-.dcf-file-and-key-binding)
4. [The addin in action](https://jozefhajnala.gitlab.io/r/r103-keypress-viewer/#the-addin-in-action)
5. [TL;DR – Just give me the package](https://jozefhajnala.gitlab.io/r/r103-keypress-viewer/#tldr---just-give-me-the-package)
6. [References](https://jozefhajnala.gitlab.io/r/r103-keypress-viewer/#references)

**Retrieving objects from sys.frames**

As a first step, we need to be able to retrieve the value of the object we are looking for based on a character string from a frame within the currently present sys.frames() for our session. This may get tricky, as it is not sufficient to only look at parent frames, because we may easily have multiple sets of “parallel” call stacks, especially when executing addins.

An example can be seen in the following screenshot, where we have a browser() call executed during the Addin execution itself. We can see that our current frame is 18 and browsing through its parent would get us to frames 17 -> 16 -> 15 -> 14 -> 0 (0 being the .GlobalEnv). The object we are looking for is however most likely in one of the other frames (9 in this particular case):



Example of sys.frames

getFromSysframes <- function(x) {

if (!(is.character(x) && length(x) == 1 && nchar(x) > 0)) {

warning("Expecting a non-empty character of length 1. Returning NULL.")

return(invisible(NULL))

}

validframes <- c(sys.frames()[-sys.nframe()], .GlobalEnv)

res <- NULL

for (i in validframes) {

inherits <- identical(i, .GlobalEnv)

res <- get0(x, i, inherits = inherits)

if (!is.null(res)) {

return(res)

}

}

return(invisible(res))

}

**Viewing files, objects, functions and more efficiently**

As a second step, we write a function to actually view our object in RStudio. We have quite some flexibility here, so as a first shot we can do the following:

1. Open a file if the selection (or the selection with quotes added) is a path to an existing file. This is useful for viewing our scripts, data files, etc. even if they are not quoted, such as the links in your Rmd files
2. Attempt to retrieve the object by the name and if found, try to use View to view it
3. If we did not find the object, we can optionally still try to retrieve the value by evaluating the provided character string. This carries some pitfalls, but is very useful for example for
   * viewing elements of lists, vectors, etc. where we need to evaluate [, [[ or $ to do so.
   * viewing operation results directly in the viewer, as opposed to writing them out into the console, useful for example for wide matrices that (subjectively) look better in the RStudio viewer, compared to the console output
4. If the View fails, we can still show useful information by trying to View its structure, enabling us to inspect objects that cannot be coerced to a data.frame and therefore would fail to be viewed.

viewObject <- function(chr,

tryEval = getOption("jhaddins\_view\_tryeval",

default = TRUE)

) {

if (!(is.character(chr) && length(chr) == 1 && nchar(chr) > 0)) {

message("Invalid input, expecting a non-empty character of length 1")

return(invisible(1L))

}

ViewWrap <- get("View", envir = as.environment("package:utils"))

# maybe it is an unquoted filename - if so, open it

if (file.exists(chr)) {

rstudioapi::navigateToFile(chr)

return(invisible(0L))

}

# or maybe it is a quoted filename - if so, open it

if (file.exists(gsub("\"", "", chr, fixed = TRUE))) {

rstudioapi::navigateToFile(gsub("\"", "", chr, fixed = TRUE))

return(invisible(0L))

}

obj <- getFromSysframes(chr)

if (is.null(obj)) {

if (isTRUE(tryEval)) {

# object not found, try evaluating

try(obj <- eval(parse(text = chr)), silent = TRUE)

}

if (is.null(obj)) {

message(sprintf("Object %s not found", chr))

return(invisible(1L))

}

}

# try to View capturing output for potential errors

Viewout <- utils::capture.output(ViewWrap(obj, title = chr))

if (length(Viewout) > 0 && grepl("Error", Viewout)) {

# could not view, try to at least View the str of the object

strcmd <- sprintf("str(%s)", chr)

message(paste(Viewout,"| trying to View", strcmd))

ViewWrap(utils::capture.output(utils::str(obj)), title = strcmd)

}

return(invisible(0L))

}

This function can of course be improved and updated in many ways, for example using the summary method instead of str for selected object classes, or showing contents of .csv (or other data) files already read into a data.frame.

**The addin function, updating the .dcf file and key binding**

If you followed the previous posts in the series, you most likely already know what is coming up next. First, we need a function serving as a binding for the addin that will execute out viewObject function on the active document’s selections:

viewSelection <- function() {

context <- rstudioapi::getActiveDocumentContext()

lapply(X = context[["selection"]]

, FUN = function(thisSel) {

viewObject(thisSel[["text"]])

}

)

return(invisible(NULL))

}

Secondly, we update the inst/rstudio/addins.dcf file by adding the binding for the newly created addin:

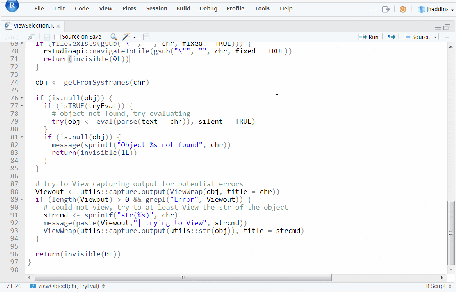
Name: viewSelection

Description: Tries to use View to View the object defined by a text selected in RStudio

Binding: viewSelection

Interactive: false

Finally, we re-install the package and assign the keyboard shortcut in the Tools -> Addins -> Browse Addins... -> Keyboard Shortcuts... menu. Personally I assigned a single F4 keystroke for this, as I use it very often:



Assigning a keyboard shortcut to use the Addin

**The addin in action**

Now, let’s view a few files, a data.frame, a function and a try-error class object just pressing F4.