Let's add an R↔Swift bridge function to retrieve all available Spotlight attributes for a macOS file:

```
library(swiftr)

swift_function('

  // Add an extension to URL which will retrieve the spotlight
  // attributes as an array of Swift Strings
  extension URL {

  var mdAttributes: [String]? {

    get {
      guard isFileURL else { return nil }
      let item = MDItemCreateWithURL(kCFAllocatorDefault, self as
CFURL)
      let attrs = MDItemCopyAttributeNames(item)!
      return(attrs as? [String])
    }

  }

}

@_cdecl ("file_attrs")
public func file_attrs(path: SEXP) -> SEXP {

  // Grab the attributres
  let outAttr = URL(fileURLWithPath: String(path)!).mdAttributes!

  // send them to R
  return(outAttr.SEXP!)

}
')
```

And, then try it out:

```
fil <-  "/Applications/RStudio.app"

file_attrs(fil)
##  [1] "kMDItemContentTypeTree"                "kMDItemContentType"
##  [3] "kMDItemPhysicalSize"                   "kMDItemCopyright"
##  [5] "kMDItemAppStoreCategory"               "kMDItemKind"
##  [7] "kMDItemDateAdded_Ranking"
"kMDItemDocumentIdentifier"
##  [9] "kMDItemContentCreationDate"
"kMDItemAlternateNames"
## [11] "kMDItemContentModificationDate_Ranking" "kMDItemDateAdded"
## [13] "kMDItemContentCreationDate_Ranking"     "
```

```
kMDItemContentModificationDate"
## [15] "kMDItemExecutableArchitectures"
"kMDItemAppStoreCategoryType"
## [17] "kMDItemVersion"
"kMDItemCFBundleIdentifier"
## [19] "kMDItemInterestingDate_Ranking"         "kMDItemDisplayName"
## [21] "_kMDItemDisplayNameWithExtensions"      "kMDItemLogicalSize"
## [23] "kMDItemUsedDates"                       "kMDItemLastUsedDate"
## [25] "kMDItemLastUsedDate_Ranking"            "kMDItemUseCount"
## [27] "kMDItemFSName"                          "kMDItemFSSize"
## [29] "kMDItemFSCreationDate"
"kMDItemFSContentChangeDate"
## [31] "kMDItemFSOwnerUserID"
"kMDItemFSOwnerGroupID"
## [33] "kMDItemFSNodeCount"                     "kMDItemFSInvisible"
## [35] "kMDItemFSTypeCode"                      "kMDItemFSCreatorCode"
## [37] "kMDItemFSFinderFlags"
"kMDItemFSHasCustomIcon"
## [39] "kMDItemFSIsExtensionHidden"
"kMDItemFSIsStationery"
## [41] "kMDItemFSLabel"
```

No `system()` (et al.) round trip!

Now, lets make R↔Swift bridge function to retrieve the value of an attribute.

Before we do that, let me be up-front that relying on `debugDescription` (which makes a string representation of a Swift object) is a terrible hack that I'm using just to make the example as short as possible. We should do far more error checking and then further check the type of the object coming from the Spotlight API call and return an R-compatible version of that type. This `mdAttr()` method will almost certainly break depending on the item being returned.

```
swift_function('
extension URL {

  // Add an extension to URL which will retrieve the spotlight
  // attribute value as a String. This will almost certainly die
  // under various value conditions.

  func mdAttr(_ attr: String) -> String? {
    guard isFileURL else { return nil }
    let item = MDItemCreateWithURL(kCFAllocatorDefault, self as CFURL)
    return(MDItemCopyAttribute(item, attr as
CFString).debugDescription!)
  }

}

@_cdecl ("file_attr")
public func file_attr(path: SEXP, attr: SEXP) -> SEXP {

  // file path as Swift String
  let xPath = String(cString: R_CHAR(Rf_asChar(path)))
```

```
  // attribute we want as a Swift String
  let xAttr = String(cString: R_CHAR(Rf_asChar(attr)))

  // the Swift debug string value of the attribute
  let outAttr = URL(fileURLWithPath: xPath).mdAttr(xAttr)

  // returned as an R string
  return(Rf_mkString(outAttr))
}
')
```

And try this out on some *carefully* selected attributes:

```
file_attr(fil, "kMDItemDisplayName")
## [1] "RStudio.app"

file_attr(fil, "kMDItemAppStoreCategory")
## [1] "Developer Tools"

file_attr(fil, "kMDItemVersion")
## [1] "1.4.1651"
```

Note that if we try to get fancy and retrieve an attribute value that is something like an array of strings, it doesn't work so well:

```
file_attr(fil, "kMDItemExecutableArchitectures")
## [1] "<__NSSingleObjectArrayI 0x7fe1f6d19bf0>(\nx86_64\n)\n"
```

Again, ideally, we'd make a small package wrapper vs use `swift_function()` for this in production, but I wanted to show how straightforward it can be to get access to some fun and potentially powerful features of macOS right in R with just a tiny bit of Swift glue code.

Also, I hadn't tried {swiftr} on the M1 Mini before and it seems I need to poke a bit to see what needs doing to get it to work properly in the arm64 RStudio `rsession`.

UPDATE (2021-04-14 a bit later)

It dawned on me that a minor tweak to the Swift `mdAttr()` function would make the method more resilient (but still hacky):

```
  func mdAttr(_ attr: String) -> String {
    guard isFileURL else { return "" }
    let item = MDItemCreateWithURL(kCFAllocatorDefault, self as CFURL)
    let x = MDItemCopyAttribute(item, attr as CFString)
    if (x == nil) {
      return("")
    } else {
      return("\(x!)")
    }
  }
```

Now we can (more) safely do something like this:

```
str(as.list(sapply(
  file_attrs(fil),
  function(attr) {
    file_attr(fil, attr)
  }
)), 1)
## List of 41
##  $ kMDItemContentTypeTree                : chr "(\n
\"com.apple.application-bundle\",\n    \"com.apple.application\",\n
\"public.executable\",\n    \"com"| __truncated__
##  $ kMDItemContentType                    : chr
"com.apple.application-bundle"
##  $ kMDItemPhysicalSize                   : chr "767619072"
##  $ kMDItemCopyright                      : chr "RStudio 1.4.1651, ©
2009-2021 RStudio, PBC"
##  $ kMDItemAppStoreCategory               : chr "Developer Tools"
##  $ kMDItemKind                           : chr "Application"
##  $ kMDItemDateAdded_Ranking              : chr "2021-04-09 00:00:00
+0000"
##  $ kMDItemDocumentIdentifier             : chr "0"
##  $ kMDItemContentCreationDate            : chr "2021-03-25 23:08:34
+0000"
##  $ kMDItemAlternateNames                 : chr "(\n    \"RStudio.app
\"\n)"
##  $ kMDItemContentModificationDate_Ranking: chr "2021-03-25 00:00:00
+0000"
##  $ kMDItemDateAdded                      : chr "2021-04-09 13:25:11
+0000"
##  $ kMDItemContentCreationDate_Ranking    : chr "2021-03-25 00:00:00
+0000"
##  $ kMDItemContentModificationDate        : chr "2021-03-25 23:08:34
+0000"
##  $ kMDItemExecutableArchitectures        : chr "(\n    \"x86_64
\"\n)"
##  $ kMDItemAppStoreCategoryType           : chr "public.app-category.
developer-tools"
##  $ kMDItemVersion                        : chr "1.4.1651"
##  $ kMDItemCFBundleIdentifier             : chr "org.rstudio.RStudio"
##  $ kMDItemInterestingDate_Ranking        : chr "2021-04-15 00:00:00
+0000"
##  $ kMDItemDisplayName                    : chr "RStudio.app"
##  $ _kMDItemDisplayNameWithExtensions     : chr "RStudio.app"
##  $ kMDItemLogicalSize                    : chr "763253198"
##  $ kMDItemUsedDates                      : chr "(\n    \"2021-03-26
04:00:00 +0000\",\n    \"2021-03-30 04:00:00 +0000\",\n    \"2021-04-02
04:00:00 +0000\",\n"| __truncated__
##  $ kMDItemLastUsedDate                   : chr "2021-04-15 00:21:45
+0000"
##  $ kMDItemLastUsedDate_Ranking           : chr "2021-04-15 00:00:00
+0000"
##  $ kMDItemUseCount                       : chr "12"
##  $ kMDItemFSName                         : chr "RStudio.app"
```

```
##  $ kMDItemFSSize                         : chr "763253198"
##  $ kMDItemFSCreationDate                 : chr "2021-03-25 23:08:34
+0000"
##  $ kMDItemFSContentChangeDate            : chr "2021-03-25 23:08:34
+0000"
##  $ kMDItemFSOwnerUserID                  : chr "501"
##  $ kMDItemFSOwnerGroupID                 : chr "80"
##  $ kMDItemFSNodeCount                    : chr "1"
##  $ kMDItemFSInvisible                    : chr "0"
##  $ kMDItemFSTypeCode                     : chr "0"
##  $ kMDItemFSCreatorCode                  : chr "0"
##  $ kMDItemFSFinderFlags                  : chr "0"
##  $ kMDItemFSHasCustomIcon                : chr ""
##  $ kMDItemFSIsExtensionHidden            : chr "1"
##  $ kMDItemFSIsStationery                 : chr ""
##  $ kMDItemFSLabel                        : chr "0"
```

We’re still better off (in the long run) checking for and using proper types.