Calling Functions - Scoping - Practice file

In-class practice

Marcus Birkenkrahe (pledged)

January 15, 2023

README

This file covers the lecture "Calling Functions - Scoping" in the Advanced Introduction to Data Science course (DSC 205):

- Scoping rules
- Environments: global, local
- Search path
- Reserved names

Most of this material can be found in Davies, Book of R, Chapter 9. Solutions can be found in GitHub (PDF).

DONE Identify and pledge yourself

1. In Emacs, replace the placeholder `[yourname]` at the top of this file by your own name and write `(pledged)` next to it
2. Go with the cursor on the headline and hange the `TODO` label to `DONE` by entering `S-<right>` ("Shift + right-arrow").

DONE Scoping

Example: `data` as an argument, and as a function -

1. create a row-wise 3x3 matrix of the numbers {1..9}

2. list all datasets in the built-in dataset `ToothGrowth`

```
matrix(
  data=1:9,
  nrow=3,
  ncol=3,
  byrow=TRUE)

data(ToothGrowth)
str(ToothGrowth)
head(ToothGrowth,n=3)
```

```
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
'data.frame': 60 obs. of  3 variables:
 $ len : num  4.2 11.5 7.3 5.8 6.4 10 11.2 11.2 5.2 7 ...
 $ supp: Factor w/ 2 levels "OJ","VC": 2 2 2 2 2 2 2 2 2 2 ...
 $ dose: num  0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 ...
   len supp dose
1  4.2   VC  0.5
2 11.5   VC  0.5
3  7.3   VC  0.5
```

DONE Global environments

Example: create three new objects and confirm their existence in the global environment:

1. a `numeric` variable `foo`
2. a `character` variable `bar`
3. an anonymous (non-argument) function `hello`
4. check the global environment
5. run `hello`

```
foo <- 3
foo
```

```
bar <- "Pedro"
bar
hello <- function() print("Hello, Marcus")
ls()
hello()
```

```
[1] 3
[1] "Pedro"
[1] "bar"         "foo"         "hello"       "ToothGrowth"
[1] "Hello, Marcus"
```

CANCELLED Package environments and namespaces

1. List the content of built-in data set `datasets` (no functions)

   ```
   ls("package:datasets")
   ```

   ```
    [1] "ability.cov"          "airmiles"             "AirPassengers"
    [4] "airquality"           "anscombe"             "attenu"
    [7] "attitude"             "austres"              "beaver1"
   [10] "beaver2"              "BJsales"              "BJsales.lead"
   [13] "BOD"                  "cars"                 "ChickWeight"
   [16] "chickwts"             "co2"                  "CO2"
   [19] "crimtab"              "discoveries"          "DNase"
   [22] "esoph"                "euro"                 "euro.cross"
   [25] "eurodist"             "EuStockMarkets"       "faithful"
   [28] "fdeaths"              "Formaldehyde"         "freeny"
   [31] "freeny.x"             "freeny.y"             "HairEyeColor"
   [34] "Harman23.cor"         "Harman74.cor"         "Indometh"
   [37] "infert"               "InsectSprays"         "iris"
   [40] "iris3"                "islands"              "JohnsonJohnson"
   [43] "LakeHuron"            "ldeaths"              "lh"
   [46] "LifeCycleSavings"     "Loblolly"             "longley"
   [49] "lynx"                 "mdeaths"              "morley"
   [52] "mtcars"               "nhtemp"               "Nile"
   [55] "nottem"               "npk"                  "occupationalStatus"
   [58] "Orange"               "OrchardSprays"        "PlantGrowth"
   [61] "precip"               "presidents"           "pressure"
   [64] "Puromycin"            "quakes"               "randu"
   [67] "rivers"               "rock"                 "Seatbelts"
   ```

```
 [70] "sleep"                 "stack.loss"           "stack.x"
 [73] "stackloss"             "state.abb"            "state.area"
 [76] "state.center"          "state.division"       "state.name"
 [79] "state.region"          "state.x77"            "sunspot.month"
 [82] "sunspot.year"          "sunspots"             "swiss"
 [85] "Theoph"                "Titanic"              "ToothGrowth"
 [88] "treering"              "trees"                "UCBAdmissions"
 [91] "UKDriverDeaths"        "UKgas"                "USAccDeaths"
 [94] "USArrests"             "UScitiesD"            "USJudgeRatings"
 [97] "USPersonalExpenditure" "uspop"                "VADeaths"
[100] "volcano"               "warpbreaks"           "women"
[103] "WorldPhones"           "WWWusage"
```

2. List the visible objects of the `graphics` package:

```
ls("package:graphics")
```

```
 [1] "abline"          "arrows"          "assocplot"       "axis"
 [5] "Axis"            "axis.Date"       "axis.POSIXct"    "axTicks"
 [9] "barplot"         "barplot.default" "box"             "boxplot"
[13] "boxplot.default" "boxplot.matrix"  "bxp"             "cdplot"
[17] "clip"            "close.screen"    "co.intervals"    "contour"
[21] "contour.default" "coplot"          "curve"           "dotchart"
[25] "erase.screen"    "filled.contour"  "fourfoldplot"    "frame"
[29] "grconvertX"      "grconvertY"      "grid"            "hist"
[33] "hist.default"    "identify"        "image"           "image.default"
[37] "layout"          "layout.show"     "lcm"             "legend"
[41] "lines"           "lines.default"   "locator"         "matlines"
[45] "matplot"         "matpoints"       "mosaicplot"      "mtext"
[49] "pairs"           "pairs.default"   "panel.smooth"    "par"
[53] "persp"           "pie"             "plot"            "plot.default"
[57] "plot.design"     "plot.function"   "plot.new"        "plot.window"
[61] "plot.xy"         "points"          "points.default"  "polygon"
[65] "polypath"        "rasterImage"     "rect"            "rug"
[69] "screen"          "segments"        "smoothScatter"   "spineplot"
[73] "split.screen"    "stars"           "stem"            "strheight"
[77] "stripchart"      "strwidth"        "sunflowerplot"   "symbols"
[81] "text"            "text.default"    "title"           "xinch"
[85] "xspline"         "xyinch"          "yinch"
```

3. Load (after installation in the R console buffer) the `dplyr` package and run the function `dplyr::filter`.

   ```
   library(dplyr)
   dplyr::filter
   ```

   ```
   function (.data, ..., .preserve = FALSE)
   {
       UseMethod("filter")
   }
   <bytecode: 0x0000016a25a7d808>
   <environment: namespace:dplyr>
   ```

DONE Local or lexical environments

Example: create a 2x2 `matrix` named `youthspeak` and pass as `data` in the argument: "OMG", "LOL", "WTF", "YOLO".

```
youthspeak <- matrix(
  data=c("OMG","LOL","WTF","YOLO"),
  nrow=2)
youthspeak
```

```
     [,1]  [,2]
[1,] "OMG" "WTF"
[2,] "LOL" "YOLO"
```

DONE Search Path

1. You can view the search path with `search()`:

   ```
   search()
   ```

   ```
    [1] ".GlobalEnv"        "package:car"       "package:carData"
    [4] "package:dplyr"     "ESSR"              "package:stats"
    [7] "package:graphics"  "package:grDevices" "package:utils"
   [10] "package:datasets"  "package:methods"   "Autoloads"
   [13] "package:base"
   ```

2. Example: create a vector of 5 elements with values between 0 and 3 with `seq`, and print it:

```
seq(from=0,to=3,length.out=5)
```

```
[1] 0.00 0.75 1.50 2.25 3.00
```

3. You can look up the environment of any function using `environment`, e.g.

   - `seq`
   - `abline`
   - `filter`

```
environment(seq)
environment(abline)
environment(filter)
```

```
<environment: namespace:base>
<environment: namespace:graphics>
<environment: namespace:dplyr>
```

4. When a package is loaded with `library`, it is inserted in the search path right after the global environment, along with all its dependencies: load the package `car` and print the search path.

```
library(car)
search()
```

```
 [1] ".GlobalEnv"        "package:car"       "package:carData"
 [4] "package:dplyr"     "ESSR"              "package:stats"
 [7] "package:graphics"  "package:grDevices" "package:utils"
[10] "package:datasets"  "package:methods"   "Autoloads"
[13] "package:base"
```

5. List the contents of the dependency, the dataset `carData`

```
ls('package:carData')
```

```
 [1] "Adler"           "AMSsurvey"       "Angell"          "Anscombe"
 [5] "Arrests"         "Baumann"         "BEPS"            "Bfox"
 [9] "Blackmore"       "Burt"            "CanPop"          "CES11"
[13] "Chile"           "Chirot"          "Cowles"          "Davis"
[17] "DavisThin"       "Depredations"    "Duncan"          "Ericksen"
[21] "Florida"         "Freedman"        "Friendly"        "Ginzberg"
[25] "Greene"          "GSSvocab"        "Guyer"           "Hartnagel"
[29] "Highway1"        "KosteckiDillon"  "Leinhardt"       "LoBD"
[33] "Mandel"          "Migration"       "Moore"           "MplsDemo"
[37] "MplsStops"       "Mroz"            "OBrienKaiser"    "OBrienKaiserLong"
[41] "Ornstein"        "Pottery"         "Prestige"        "Quartet"
[45] "Robey"           "Rossi"           "Sahlins"         "Salaries"
[49] "SLID"            "Soils"           "States"          "TitanicSurvival"
[53] "Transact"        "UN"              "UN98"            "USPop"
[57] "Vocab"           "WeightLoss"      "Wells"           "Womenlf"
[61] "Wong"            "Wool"            "WVS"
```

6. An error is thrown if you request a function or object

   - that you haven't **defined**,
   - that doesn't **exist**,
   - that is in a contributed package that you've forgotten to **load**

```
neither.here() # undefined function
nor.there      # undefined object
```

```
Error in neither.here() : could not find function "neither.here"
Error: object 'nor.there' not found
```

DONE Reserved and protected names

1. What happens when you assign a value to an `NaN`?

```
NaN <- 5
```

```
Error in NaN <- 5 : invalid (do_set) left-hand side to assignment
```

2. Since R is case-sensitive, you can assign values to case variants of these keywords, causing much confusion:

```
nan <- "this"
inf <- "is"
Null <- "very"
False <- "confusing"
paste(nan,inf,Null,False)
```

```
[1] "this is very confusing"
```

3. `T` and `F` can also be overwritten - don't do it since they are the abbreviations for `TRUE` and `FALSE`:

```
T <- FALSE
F <- TRUE
paste(T, "is", F)
paste("2+2=5 is", (2+2==5)==T)
(2+2==5)==TRUE
```

```
[1] "FALSE is TRUE"
[1] "2+2=5 is TRUE"
[1] FALSE
```

4. With all these confusing changes, clear the global environment now!

```
ls()
rm(list=ls()) ## remove the list of user-defined R objects
ls()
```

```
 [1] "bar"         "F"           "False"       "foo"         "hello"
 [6] "inf"         "nan"         "Null"        "T"           "ToothGrowth"
[11] "youthspeak"
character(0)
```

TODO Wow! Congratulations!

Well done! You've reached the end of the first in-class practice file.

If you've completed all steps, you upload the Org-mode file to Canvas (see `Assignments > In-class practice`).