Homework 1, TSCI 5050

Alex F. Bokov, [bokov@uthscsa.edu](mailto:bokov@uthscsa.edu)

Student Name:\_Omar Gabriel Catano\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Good work! I have reviewed this homework and my comments are in yellow.

This homework is graded pass/fail and cooperation is encouraged. Bring the homework (hard-copy) to class, but you can take it back with you and hand it in during the following class if you need more time.

Part 1.

The point of these exercises is to get you to appreciate patterns in code even when you can't fully rely on prior knowledge. Without running this code or looking anything up, see how far you can get writing down in the white space what you think each of the following blocks of code is intended to do.

1.1

riddle <- function (object = nm, nm)

{

names(object) <- nm

object

}

**This code creates a function called *riddle*. It requires 2 arguments**

Correct.

**and it delivers a pair of outcomes.**

Actually, no. A function in R can only have one output-- the last expression that is evaluated within the {} brackets before the function exits. There do exist tricks to work around this, but riddle is an ordinary function that accepts an argument that it internally refers to as object, modifies it, and returns the modified version.

***nm* is defined for another function called *names*.**

A number of people got confused by this. Several functions in R can be the targets of assignment. Here, FOO <- names(object) would extract the names of an object and save them to another object named FOO (or show them on the screen and forget about them if you leave out FOO <- ). But names(object) <- nm does the reverse-- it assumes that nm is a vector of names and tries to assign it to the values contained in object. Not all R functions can accept assignments in this way, but here are some others that do: length(), rownames(), colnames(), levels(), attr(). If you're curious you can get a more complete list with this command: grep("<-$",ls(package:base),val=T). The most commonly used of these are the ones which we don't even notice are functions but we use them every time we subset or assign to a vector or data.frame: `[`, `[[`, and `$`.

1.2

conundrum <- function (len)

{

out <- c(1,1)

for (ii in 3:len){

out <- c(out,out[ii-1] + out[ii-2])

}

out

}

**It creates a function called *conundrum*.**

Yes

**After the vector *out* is created, the function renders a vector where the first two elements are those of the vector out (1,1), and then the function calls for the *ii* position (length)in the sequence.**

...for each value of ii between 3 and the length specified by the function's len argument the latest two values of out are summed together and appended to out.

What do you think “<-” does?

Assignment

Yes

What does “function” do?

Define functions

Yes

1.3 This one is more challenging. But it is not one monolithic blob of gibberish, right?

mystery <- function (object, ..., digits = max(3L, getOption("digits") - 3L))

{

if (is.factor(object))

return(summary.factor(object, ...))

else if (is.matrix(object))

return(summary.matrix(object, digits = digits, ...))

value <- if (is.logical(object))

c(Mode = "logical", {

tb <- table(object, exclude = NULL)

if (!is.null(n <- dimnames(tb)[[1L]]) && any(iN <- is.na(n))) dimnames(tb)[[1L]][iN] <- "NA's"

tb

})

else if (is.numeric(object)) {

nas <- is.na(object)

object <- object[!nas]

qq <- stats::quantile(object)

qq <- signif(c(qq[1L:3L], mean(object), qq[4L:5L]), digits)

names(qq) <- c("Min.", "1st Qu.", "Median", "Mean", "3rd Qu.", "Max.")

if (any(nas))

c(qq, `NA's` = sum(nas))

else qq

}

else if (is.recursive(object) && !is.language(object) && (n <- length(object))) {

sumry <- array("", c(n, 3L), list(names(object), c("Length", "Class", "Mode")))

ll <- numeric(n)

for (i in 1L:n) {

ii <- object[[i]]

ll[i] <- length(ii)

cls <- oldClass(ii)

sumry[i, 2L] <- if (length(cls))

cls[1L]

else "-none-"

sumry[i, 3L] <- mode(ii)

}

sumry[, 1L] <- format(as.integer(ll))

sumry

}

else c(Length = length(object), Class = class(object), Mode = mode(object))

class(value) <- c("summaryDefault", "table")

value

}

This creates a function call *mystery* with multiple options of entries. It renders information regarding the type of object used.

Yes

What is the purpose of the parentheses immediately after the “function”?

The parentheses are passing arguments to a function

Basically yes. The way I would put it is they tell R where to look for function arguments ( ( )and when to stop looking ( ) ).

What is the purpose of “{” and “}”? How about “[” and “]”?

{} Denotes a block of code in a function

[] Subsets vectors and dataframes; subscripting; index position in a list or dataframe.

Yes, excellent.

Part 2.

Please read the following sections in <http://www.burns-stat.com/pages/Spoetry/Spoetry.pdf> (S Poetry by Patrick J. Burns, 1998). If anything you read inspires you to update your answers in Part 1, please do so.

The basics.

1.1 – 1.3 (pp 1-13)

1.6 (pp 16-17)

1.8 – 1.9 (pp 18-19)

Now we get into Zen stuff. Once you get the point he's trying to make in each section, just skim until the next section.

2.1 – 2.4 (pp 23-30)

Some useful functions.

4.3, first part (pp 81- 83, the rest is optional but the abbreviate command is kind of cool)

We'll skip 4.4, but remind me to mention the simple and almost-universal way to get data into R.

4.8 (pp 110 – 113)

We'll skip 4.9, but remind me to mention the main graphics commands to know about.

Of course your code will always be perfect the first time you write it ;-) but maybe some of the following material can be helpful in troubleshooting your colleagues' work:

6.3 (pp 138 – 139, the rest is optional; if you do keep reading note that inspect is not a command in modern R, instead look at the help file for debug)

6.5 – 6.6 (pp 149 - 151)

Part 3.

Start up Rstudio. Do you remember from section 6.3 (in the reading, above) and from typing ?debug how to step through the code one command at a time? If there are any expressions in the Part 1 code that are still a mystery to you, underline just the parts you think are blocking your understanding, see how specifically you can identify exactly what parts you don't get.

Part 4

Think of a small but annoying data-related problem you deal with in real life and see how far you can get in RStudio writing a function to solve this problem. Save the function to hand in with the homework.