Lab 3 - Miscellaneous Tasks

Stat 20 Summer 2019

Due August 2, 2019 by 11:59pm via upload to CCLE

General Instructions

1. The lab should be written as a .Rmd file and knitted to a .html file or .PDF file. An upload link can be found on CCLE under “assignments”. Please upload an .Rmd and the knitted .html.
2. You may work in groups. I would encourage that you collaborate, decide on what to do together, work through constructing whatever additional variables together. **BUT** you are individually responsible for generating a lab for submission and you are responsible for writing your own answers to the questions (as indicated below). This includes the function writing portion of the lab. Working in groups – what this means is that we encourage you to have other people around you to discuss possible solutions and to ask for assistance and advice **BUT the help stops when you are composing your lab submission, please do not copy and paste a friend’s answer and call it yours, we want you to learn how to do this type of work on your own – in other words, you should “own” your work** Here is an example from office hours, when people visit in office hours, they approach me with a question and we work through a solution, usually I don’t give complete solutions but I give the visitor enough to solve the problem on their own OR I do things like examine code that they have written and point out what needs to be fixed. They do their own typing. It’s important, your fingers will help your brain to remember…

1 Examine an R Function and modify it slightly

R will reveal the contents of its functions if you simply type the function name without any parentheses and argunents so typing `sd’ for example with no quotes reveals the following:

```
sd
```

```
function (x, na.rm = FALSE)
sqrt(var(if (is.vector(x) || is.factor(x)) x else as.double(x),
    na.rm = na.rm))
<bytecode: 0x7ffee4dc7d50>
<environment: namespace:stats>
```

Please rewrite the R code for the sd function and save it to a new function called *mysd1*. It should work the same as sd, but the code could be made a little more readable. Test your new function by apply the built in data named ``rivers" (noquotes) to it. So for example, here is my result compared with the sd function in base R:

```
mysd1(rivers)
```

```
[1] 493.8708
```

```
sd(rivers)
```

```
[1] 493.8708
```

2 Modify an R function to accommodate something different.

If I attempt to insert a data frame into these functions, I will get errors:

```
mysd1(USArrests)
```

```
Error in is.data.frame(x): (list) object cannot be coerced to type 'double'
```

```
sd(USArrests)
```

```
Error in is.data.frame(x): (list) object cannot be coerced to type 'double'
```

Please modify your function and save it as ``mysd2" (no quotes) so that it will work when you insert a dataframe as the first argument. For example:

```
mysd2(rivers)
```

```
[1] 493.8708
```

```
mysd2(USArrests)
```

```
   Murder   Assault  UrbanPop      Rape 
 4.355510 83.337661 14.474763  9.366385 
```

3 Write your own function: the Pythagorean Theorem.

The Pythagorean Theorem states that the square of the hypotenuse (the side opposite the right angle) is equal to the sum of the squares of the other two sides.

$$a^2 + b^2 = c^2$$

Please create the following

a. Name your function “pythag” and it should work with single values and vectors, for example:

```
pythag(4,5)
```

```
## [1] 6.403124
```

```
pythag(4:10,2:8)
```

```
## [1]  4.472136  5.830952  7.211103  8.602325 10.000000 11.401754 12.806248
```

b. Modify your function so that it verifies the input before proceeding. You should also make the output look a little nicer

```
pythag(4,5)
```

```
## $hypoteneuse
## [1] 6.403124
##
## $sidea
## [1] 4
##
## $sideb
## [1] 5
```

```
pythag("A","B")
```

```
## Error in pythag("A", "B"): I need numeric values to make this work
```

```
pythag(-4,-5)
```

```
## Error in pythag(-4, -5): values need to be positive
```

```
pythag(4:10,2:8)
```

```
## $hypoteneuse
## [1]  4.472136  5.830952  7.211103  8.602325 10.000000 11.401754 12.806248
##
## $sidea
## [1]  4  5  6  7  8  9 10
##
## $sideb
## [1] 2 3 4 5 6 7 8
```

4. Loops vs. lapply

Different ways to loop, ordered by typical speed (fastest to slowest)

We will make some data

```
set.seed(1)
a <- data.frame(matrix(sample(1:1000,1000000, replace=TRUE), nrow=100000, ncol=10))
names(a) <- c("alpha", "bravo", "charlie", "delta", "echo", "foxtrot",
                            "golf", "hotel", "india", "juliet")
```

Timing for loop style 1.

```
system.time({out <- data.frame(matrix(NA, nrow(a), ncol(a)))
names(out) <- names(a)
for(i in seq_along(a)){
  out[[i]] <- sqrt(a[,i])
}
})
```

```
##    user  system elapsed
##   0.017   0.005   0.022
```

Timing for loop style 2.

```
system.time({out2 <- data.frame(matrix(NA, nrow(a), ncol(a)))
names(out2) <- names(a)
nm <- names(a)
for(nm in names(a)){
  out2[[nm]] <-sqrt(a[[nm]])
}
})
```

```
##    user  system elapsed
##   0.015   0.005   0.019
```

Timing for loop style 3. This is slow because each time you extend the vector, R has to copy all of the existing elements. Note this one is 1/100 the size of the previous 2.

```
xs <- as.vector(unlist(a[1:1000,]))
system.time({
res <- c()
for (x in xs) {
  res <-c(res, sqrt(x))
}
  res <-data.frame(matrix(res, nrow=1000, ncol=10))
})
```

```
##    user  system elapsed
##   0.154   0.100   0.253
```

Timing for lapply style 1, as it was designed to be used

```
system.time({a2 <- lapply(a, function(x) sqrt(x))
names(a2) <- names(a)
})
```

```
##    user  system elapsed
##   0.009   0.004   0.012
```

Timing for lapply style 2 using an index (which could be preserved if you needed the position)

```
system.time({a2 <- lapply(seq_along(a), function(i) sqrt(a[[i]]))
names(a2) <- names(a)
})
```

```
##    user  system elapsed
##   0.009   0.003   0.012
```

Timing for lapply style 3 using names (which could be preserved if you needed the name)

```
system.time({nm <- names(a)
  a2 <- lapply(names(a), function(nm) sqrt(a[[nm]]))
names(a2) <- names(a)
})
```

```
##    user  system elapsed
##   0.007   0.000   0.008
```

For your part 4

1. Use the built in data frame “baseball” from package plyr and then
2. Show us how to display the class of each column, e.g.,

```
        id        year       stint        team          lg           g          ab
"character"   "integer"   "integer" "character" "character"   "integer"   "integer"
         r           h         X2b         X3b          hr         rbi          sb
 "integer"   "integer"   "integer"   "integer"   "integer"   "integer"   "integer"
        cs          bb          so         ibb         hbp          sh          sf
 "integer"   "integer"   "integer"   "integer"   "integer"   "integer"   "integer"
      gidp
 "integer"
```

3. Show us how to divide the value of variables g through hr (variables 6 through 12) in baseball by the maximum of each of those columns for each year (1871-2007). The result should be one large data frame with the following variables: id, year and the 7 modified variables (so 9 variables total)

```
head(baseball[,c(1,2,6:12)], n=7)
```

	id <chr>	year <int>	g <int>	ab <int>	r <int>	h <int>	X2b <int>	X3b <int>	hr <int>
4	ansonca01	1871	25	120	29	39	11	3	0
44	forceda01	1871	32	162	45	45	9	4	0
68	mathebo01	1871	19	89	15	24	3	1	0
99	startjo01	1871	33	161	35	58	5	1	1
102	suttoez01	1871	29	128	35	45	3	7	3
106	whitede01	1871	29	146	40	47	6	5	1
113	yorkto01	1871	29	145	36	37	5	7	2

7 rows

```
head(baseball2, n=7)
```

	id <chr>	y… <int>	g <dbl>	ab <dbl>	r <dbl>	h <dbl>	X2b <dbl>	X3b <dbl>	h <dbl>
4	ansonca01	1871	0.7575758	0.7407407	0.6444444	0.6724138	1.0000000	0.4285714	0.000000
44	forceda01	1871	0.9696970	1.0000000	1.0000000	0.7758621	0.8181818	0.5714286	0.000000
68	mathebo01	1871	0.5757576	0.5493827	0.3333333	0.4137931	0.2727273	0.1428571	0.000000
99	startjo01	1871	1.0000000	0.9938272	0.7777778	1.0000000	0.4545455	0.1428571	0.333333
102	suttoez01	1871	0.8787879	0.7901235	0.7777778	0.7758621	0.2727273	1.0000000	1.000000
106	whitede01	1871	0.8787879	0.9012346	0.8888889	0.8103448	0.5454545	0.7142857	0.333333
113	yorkto01	1871	0.8787879	0.8950617	0.8000000	0.6379310	0.4545455	1.0000000	0.666666

7 rows

```
str(baseball2)
```

```
## 'data.frame':    21699 obs. of  9 variables:
##  $ id  : chr  "ansonca01" "forceda01" "mathebo01" "startjo01" ...
##  $ year: int  1871 1871 1871 1871 1871 1871 1871 1872 1872 1872 ...
##  $ g   : num  0.758 0.97 0.576 1 0.879 ...
##  $ ab  : num  0.741 1 0.549 0.994 0.79 ...
##  $ r   : num  0.644 1 0.333 0.778 0.778 ...
##  $ h   : num  0.672 0.776 0.414 1 0.776 ...
##  $ X2b : num  1 0.818 0.273 0.455 0.273 ...
##  $ X3b : num  0.429 0.571 0.143 0.143 1 ...
##  $ hr  : num  0 0 0 0.333 1 ...
```

5. Interacting with files outside of R

A. Suppose there is a zip archive stored on the internet. We can download it to our computer and read it into R while in R:

```
someURL<-"http://www.stat.ucla.edu/~vlew/datasets/ucpay.zip"
download.file(someURL,"ucpay.zip")
unzip("ucpay.zip", exdir = "ucpaydata")
list.files("ucpaydata")
```

```
[1] "__MACOSX"      "ucpay2009.csv" "ucpay2010.csv" "ucpay2011.csv"
```

```
uc09 <- read.csv("ucpaydata/ucpay2009.csv", header = FALSE)
tail(uc09[order(uc09$V7),])
```

	V1 <int>	V2 <int>	V3 <fctr>	V4 <fctr>	V5 <fctr>
138464	1385122	2009	LOS ANGELES	RUBIN , AMIR DAN	(FUNCTL AREA) OFFICER-EXEC
249973	1496631	2009	UCOP	STOBO , JOHN DAVID , DR.	SR. VICE PRES--DESIGNATE
63499	1310157	2009	DAVIS	RICE , ANN MADDEN	CHIEF EXEC OFFICER - MED CI
250230	1496888	2009	UCOP	YUDOF , MARK GEORGE	PRESIDENT OF THE UNIVERSIT
119493	1366151	2009	LOS ANGELES	FEINBERG , DAVID T	DIRECTOR (FUNCTL AREA)-EXI
209601	1456259	2009	SAN FRANCISCO	LARET , MARK R	DIRECTOR (FUNCTL AREA)-EXI

6 rows | 1-6 of 11 columns

Please demonstrate how to do the same for the zip archive "http://www.stat.ucla.edu/~vlew/datasets/spssSTUFF.zip (http://www.stat.ucla.edu/~vlew/datasets/spssSTUFF.zip)". You only need to program up to and including the list.files() call for full credit.

B. Suppose you have many files with similar names on your computer's hard drive:

```
## [1] "death00.csv" "death01.csv" "death02.csv" "death03.csv" "death04.csv"
```

Please demonstrate how you could read them all into R using *at most* one line of code (no semi-colons allowed but nesting and piping are OK).