MATH 6359, Statistical Computing, Homework 1

Andrey Skripnikov

August 25, 2017

SUBMISSION GUIDELINES:

- **Bring the hard-copy of typed up solutions to the class on Tuesday, September 5.**
- **Try to keep it under 5 pages total (all included - text, code, plots, tables). DON'T (!) repeat the problem formulation, go straight to solution.**
- **For problems 1 and 3 - try and follow the example format in terms of conciseness.**
- **For problem 2 - simply provide the code.**
- **Point total is 55 (100%), and on top of that one can get 8 extra credit points total (as a grade cushion for later in the course).**

PROBLEM #1 - 20 points.

Define your own function in R and demonstrate its functionality. Function needs to contain:

- **at least one assignment operation ('<-')**
- **at least one *required* argument (without a default value)**
- **at least one argument with a default value**
- **at least one conditional if-expression**

Demonstrate:

- **2-3 calls (not more, not less) of that function for specific *required* inputs, showing that it works properly**

- **a call of that function in which you specify a non-default value (see in example below)**

Among millions of other ideas, you can consider a function that - checks if a string/sequence has a certain property, or makes a certain cyclic algebraic calculation, or does an unorthodox matrix operation, or provides various plots for inputted sequence/data as a result of the call, and that's just to name a few.

EXAMPLE: Function *is.increasing*() allows to check if a sequence of numbers is monotonically increasing (returns TRUE) or not (returns FALSE). Upon user's request, it can also print out the consecutive differences.

```
is.increasing <- function(x,printout=FALSE){
  n <- length(x)
  dif.vec <- x[2:n]-x[1:(n-1)]
  if (printout == TRUE) print(dif.vec)
  if (sum(dif.vec<=0) != 0) return(F)  # Important note: function stops after executing the
  return(T)                            # first 'return' statement it gets to
}

x <- c(1:10)
is.increasing(x)
[1] TRUE

y <- c(5,5,5)
is.increasing(y)
[1] FALSE

z <- c(1,5,3,7,9)
is.increasing(z)
[1] FALSE

is.increasing(z,printout=T)   # The call where you specify the non-default value
[1]  4 -2  4  2
[1] FALSE

w <- 2
is.increasing(w)
# This will give an error, and could've been accounted for in the function definition,
# BUT you don't have to be that meticulous for a full grade.
# Just make the function work on reasonable examples, for which it was created.
```

PROBLEM #2 (simply provide the code) - 15 points (+5 EXTRA).

Count the total number of boys in data set juul (from package ISwR), who have insulin-like growth factor greater than 400, in three different ways (make sure the output is the same though):

1. **Using for-loop**
2. **Using vectorized operations** (can be done in one line)
3. **Using data frame subsetting** (can be done in one line)

FOR EXTRA CREDIT (+5 points): Have R outputting a table which shows the total number of kids with igf >400, broken down by sex like this (those are not correct numbers for obvious reasons, but the format should be the same):

```
  M   F
150 198
```

Below is the code to get you started - it extracts the needed two variables ('sex' and 'igf1') from *juul*, and **cleans it up** (removes missing observations):

```
library(ISwR)
juul.sex.igf1 <- juul[c("sex","igf1")]
head(juul.sex.igf1)                              # - Both head() and summary() calls.
summary(juul.sex.igf1)                           # show there are missing observations (NA).
juul.sex.igf1.noNA <- na.omit(juul.sex.igf1)  # - Excludes the missing observations.
head(juul.sex.igf1.noNA)                         # - Now NA's are gone.
summary(juul.sex.igf1.noNA)                      # - Yep, they are gone.

attach(juul.sex.igf1.noNA)

## for-loop - goes through values of 'sex' and 'igf1' scalar-by-scalar

....

## vectorized solution - operates on 'sex' and 'igf1' as vectors

....


# subsetting solution - via function subset() for the 'juul.sex.igf1.noNA' data frame
#                       + another function that actually counts the number

....
```

PROBLEM #3 - 20 points (+3 EXTRA)

Calculate various descriptive statistics, make some basic plot(s) and provide short summary for a data set. Data set can be borrowed from any R package BUT the 'ISwR' (check `https://vincentarelbundock.github.io/Rdatasets/datasets.html` for several packages and the near-infinite number of data sets contained in them). One gets EXTRA CREDIT (+3 points) if they import the data from a file (might as well download '.csv' from that link above). Data set should include at least two variables being observed on each subject, and you should focus your calculations on those two variables.

Summary statistics: calculate such basic summary statistics for variables as mean, median, standard deviation and range. Preferably summarize them in a table.

Plots: Depending on the type of variables you have in your data you may either 1) plot two continuous variables against each other in a scatter plot, or 2) plot sorted values of a single continuous variable separately for different levels of the categorical variable (e.g. weights for boys and for girls), or 3) use whichever other graphical display you think is more appropriate for your data.

Short summary: Summarize (very briefly, 2-3 sentences) your findings from summary statistics and graphics in plain English.

EXAMPLE: GPA and SAT scores data from 'Stat2Data' package. The data set contains data on student GPAs and their respective SAT scores. We will focus on Math SAT and GPAs (picked two variables).

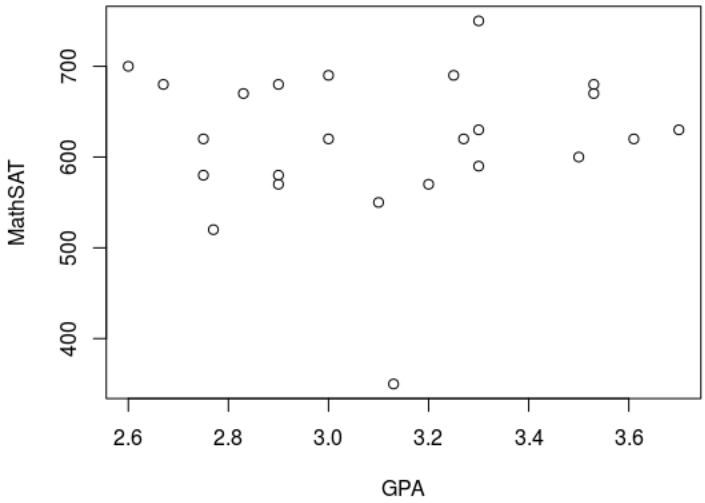
Code:

```
library(Stat2Data) # - 'Stat2Data' is one of those packages
data(SATGPA)       #  where you have to call 'data()' to load the frame.
attach(SATGPA)
head(SATGPA)
nrow(SATGPA)       # - Sample size.
summary(MathSAT)   # - Gives you most summary statistics for Math SAT.
sd(MathSAT)        # - Standard deviation.
summary(GPA)       # - Same for GPA.
sd(GPA)
plot(GPA,MathSAT)  # - For graphics.
```

Descriptive statistics:

	Mean (St.Dev)	Median	Min	Max
GPA	3.116 (0.32)	3.115	2.6	3.7
Math SAT	619.2 (79.77)	620.0	350.0	750.0

Graphics:



Short summary: A sample survey of 24 statistics students revealed GPAs ranging from 2.6 to 3.7 with *mean* $\pm$ *sd* interval being 3.115 ± 0.32, while their math SATs ranged from 350 to 750 with 619 ± 80 interval. Graphics revealed very low dependence of stats students' overall GPA and their performance on math SAT - vast majority scored within 500-700 irrespective of their GPA.