R functions

Vivian Chau (A16913056)

Today we will get more exposure to functions in R. We call functions to do all our work and today we will learn how to write our own.

A first silly function

Note that arguments 2 and 3 have default values because we set y=0 and z=0) so we don't have to supply them when we call our function.

```
add <-function(x,y=0,z=0){
  x+y+z
  }
```

Can I just use this

```
add(1,1)
```

```
[1] 2
```

```
add(1, c(10,100))
```

```
[1]  11 101
```

```
add(100)
```

```
[1] 100
```

```
add(100,10,1)
```

```
[1] 111
```

A second more fun function

Let's write a function that generates random nucleotide sequences.

We can make use of the in-built `sample()` function in R to help us here.

```r
sample(1:10, size=9)
```

```
[1]  7  9  3  1  5  4 10  8  6
```

```r
sample(1:10, size=11, replace=TRUE)
```

```
 [1]  5  6  6 10  2  7 10  9  1  5  5
```

> Q. Can you use `sample()` to generate a random nucleotide sequence of length 5.

```r
sample(x=c("A","T","C","G"),size=5,replace=TRUE)
```

```
[1] "T" "C" "T" "A" "C"
```

> Q. Write a function `generate_dna()` that makes a nucleotide sequence of a use specified length.

Every function in R has at least 3 things:

- a **name** (in our case "generate_dna")
- one or more **input arguments** (the "length" of sequence we want)
- a **body** (R code that does the work)

```r
generate_dna <- function(length) {
  bases <- c("A","T","C","G")
  sample(bases, size=length, replace=TRUE)
  }
```

```r
generate_dna(10)
```

```
 [1] "A" "C" "A" "A" "A" "T" "T" "G" "T" "A"
```

> Q. Can you write a `generate protein()` function that returns amino acid sequence of a user requested length?

```r
generate_protein <- function (length=5) {
  aa<-bio3d::aa.table$aa1[1:20]
  sample(aa, size=length, replace=TRUE)
}
```

```r
generate_protein(10)
```

```
 [1] "Q" "I" "N" "T" "R" "Y" "S" "S" "T" "H"
```

I want my output of this function not to be a vector with one amino acid per element but rather a one element single string.

```r
bases<-c("A","G","C","T")
paste(bases, collapse="")
```

```
[1] "AGCT"
```

```r
generate_protein <- function (length=5) {
  aa<-bio3d::aa.table$aa1[1:20]
  s<-sample(aa, size=length, replace=TRUE)
  paste(s,collapse="")
}
```

```r
generate_protein()
```

```
[1] "YMNWT"
```

> Q. Generate protein sequences from lenght 6 to 12?

```r
generate_protein(length=6)
```

```
[1] "RWRNMP"
```

```r
generate_protein(length=7)
```

```
[1] "DRCHFQP"
```

```r
generate_protein(length=8)
```

```
[1] "HPDAEGLG"
```

We can use the useful utility function `sapply()` to help us "apply" our function over all the values 6 to 12.

```r
ans<-sapply(6:12, generate_protein)
ans
```

```
[1] "RRCIKQ"       "PRACHAM"      "FLAQYEAA"     "LTDWQCAMP"    "QMCNCINTSN"  
[6] "VPTLYDMQEAI"  "LPQPREYWCFTI"
```

```r
cat( paste(">ID.", 6:12, sep="","\n", ans,"\n"), sep="" )
```

```
>ID.6
RRCIKQ
>ID.7
PRACHAM
>ID.8
FLAQYEAA
>ID.9
LTDWQCAMP
>ID.10
QMCNCINTSN
>ID.11
VPTLYDMQEAI
>ID.12
LPQPREYWCFTI
```

> Q. Are any of these sequences unique in nature - i.e. never found in nature. We can search "refseq-protein" and look for 100% Ide and 100% coverage matches with BLASTp

Yes. ID. 6 has 100% identity and coverage match with an uncharacterized protein from Bolinopsis microptera.ID. 7 has 100% identity and coverage match with glutamate synthase large subunit. ID. 9 has 100% identity and coverage match with CPBP family intramembrane glutamic endopeptidase.ID. 8, ID.10, ID.11, and ID.12 has no 100% identity and coverage matches.

Q1. Write a function grade() to determine an overall grade from a vector of student homework assignment scores dropping the lowest single score. If a student misses a homework (i.e. has an NA value) this can be used as a score to be potentially dropped. Your final function should be adquately explained with code comments and be able to work on an example class gradebook such as this one in CSV format: "https://tinyurl.com/gradeinput" [3pts]

```
# Example input vectors to start with
student1 <- c(100, 100, 100, 100, 100, 100, 100, 90)
student2 <- c(100, NA, 90, 90, 90, 90, 97, 80)
student3 <- c(90, NA, NA, NA, NA, NA, NA, NA)
```

```
student1 <- c(100, 100, 100, 100, 100, 100, 100, 90)

mean(student1)
```

```
[1] 98.75
```

Identify the lowest score:

```
#Which element of the vector is the lowest?
which.min(student1)
```

```
[1] 8
```

```
#This will return everything but the element that is the lowest for Student 1
student1[-which.min(student1)]
```

```
[1] 100 100 100 100 100 100 100
```

```
x<-student2
x[is.na(x)]<-0
x
```

```
[1] 100   0  90  90  90  90  97  80
```

```
#This is the mean for student 2, not including their lowest score
x[is.na(x)]<-0
mean(x[-which.min(x)])
```

```
[1] 91
```

```r
#This is the mean for student 3, not including their lowest score
student3<-c(90, NA, NA, NA, NA, NA, NA, NA)
x<-student3
x[is.na(x)]<-0
mean(x[-which.min(x)])
```

```
[1] 12.85714
```

```r
grade<-function(x){
  x[is.na(x)]<-0
  mean(x[-which.min(x)])
  }
```

```r
grade(student1)
```

```
[1] 100
```

```r
grade(student2)
```

```
[1] 91
```

```r
grade(student3)
```

```
[1] 12.85714
```

```r
#Calculate the average score for a vector of student scores dropping the lowest score, where


student<-c(100, NA,90,97)
grade(student)
```

```
[1] 95.66667
```

```r
grade<-function(x){
  #Treat missing values as zero
  x[is.na(x)]<-0
  #Exclude lowest score from mean
  mean(x[-which.min(x)])
  }
```

```r
url<-"https://tinyurl.com/gradeinput"
gradebook<-read.csv(url,row.names=1)
```

Q2.Using your grade() function and the supplied gradebook, Who is the top scoring student overall in the gradebook? [3pts]

```r
results<-apply(gradebook,1,grade)
sort(results,decreasing=TRUE)
```

```
student-18  student-7  student-8 student-13  student-1 student-12 student-16
     94.50      94.00      93.75      92.25      91.75      91.75      89.50
 student-6  student-5 student-17  student-9 student-14 student-11  student-3
     89.00      88.25      88.00      87.75      87.75      86.00      84.25
 student-4 student-19 student-20  student-2 student-10 student-15
     84.25      82.75      82.75      82.50      79.00      78.75
```

```r
#This is the top scoring student overall in the gradebook.
which.max(results)
```

```
student-18
        18
```

Q3.From your analysis of the gradebook, which homework was toughest on students (i.e. obtained the lowest scores overall? [2pts]

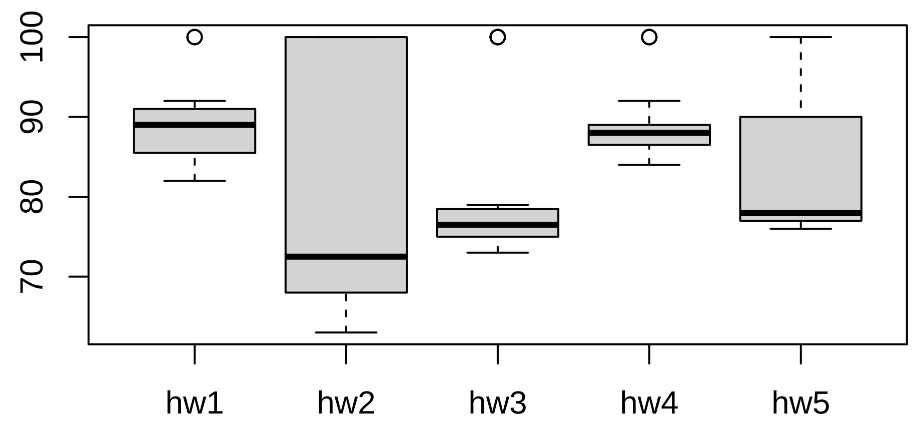
```r
avg.scores<-apply(gradebook,2,mean,na.rm=TRUE)
avg.scores
```

```
     hw1      hw2      hw3      hw4      hw5
89.00000 80.88889 80.80000 89.63158 83.42105
```

```r
which.min(avg.scores)
```

```
hw3
  3
```

```r
boxplot(gradebook)
```



A. HW 2 was toughest on students.

Q4.Optional Extension: From your analysis of the gradebook, which homework was most predictive of overall score (i.e. highest correlation with average grade score)? [1pt]

```r
masked.gradebook<-gradebook
masked.gradebook[is.na(masked.gradebook)]<-0
masked.gradebook
```

```
          hw1 hw2 hw3 hw4 hw5
student-1 100  73 100  88  79
student-2  85  64  78  89  78
student-3  83  69  77 100  77
```

```
student-4    88   0  73 100  76
student-5    88 100  75  86  79
student-6    89  78 100  89  77
student-7    89 100  74  87 100
student-8    89 100  76  86 100
student-9    86 100  77  88  77
student-10   89  72  79   0  76
student-11   82  66  78  84 100
student-12  100  70  75  92 100
student-13   89 100  76 100  80
student-14   85 100  77  89  76
student-15   85  65  76  89   0
student-16   92 100  74  89  77
student-17   88  63 100  86  78
student-18   91   0 100  87 100
student-19   91  68  75  86  79
student-20   91  68  76  88  76
```

```
cor(results,masked.gradebook$hw5)
```

```
[1] 0.6325982
```

```
apply(masked.gradebook,2,cor,x=results)
```

```
      hw1       hw2       hw3       hw4       hw5 
0.4250204 0.1767780 0.3042561 0.3810884 0.6325982 
```

A. HW5 was most predictive of overall score.

Q5.Make sure you save your Quarto document and can click the "Render" (or Rmarkdown"Knit") button to generate a PDF foramt report without errors. Finally, submit your PDF to gradescope. [1pt]