# Assignment1

## SDGB 7844, Prof. Nagaraja

Yuwen (Suyi) Wu, 330-530PM

9/22/2018

## Question 1.a

Solution:

```
class(state.area)

## [1] "numeric"

class(state.name)

## [1] "character"

class(state.region)

## [1] "factor"
```

## Question 1.b What is the longest state name (including spaces)? How long is it?

Solution:

```
state.name[nchar(state.name)==max(nchar(state.name))]

## [1] "North Carolina" "South Carolina"

nchar(state.name[nchar(state.name)==max(nchar(state.name))])

## [1] 14 14
```

## Question 1.c

Compute the average area of the states which contain the word “New” at the start of the state name. Use the function substr().

Solution:

```
mean(state.area[substr(state.name, start=1, stop=3)=="New"])

## [1] 47095.5
```

## Question 1.d

Use the function table() to determine how many states are in each region. Use the function kable() to include the table in your solutions. (Notes: you will need the R package knitr to be able to use kable(). See the RMarkdown example in the Assignments folder on Blackboard for an example.)

Solution:

```
z<-table(state.region)
kable(z)
```

| state.region | Freq |
|---|---|
| Northeast | 9 |
| South | 16 |
| North Central | 12 |
| West | 13 |

## Question 2.a

Solution:

```
num.perfect<-2
count<-1 ## change the begin number to get first two perfect number rather
than first three perfect number
iter<-2


while(count<=num.perfect){
  divisor<-1
  for(i in 2:iter){ ## when i in 2:iter , i could be the divisor of iter
       if(iter%%i==0 & iter!=i) ## the i could not be the iter in order to
get correct sum of i
         divisor<-c(divisor,i)
         i<-i+1## add this line to iterrate add the i , and make this loop
could be finished
       }# end for loop

       if(sum(divisor)==iter){
    print(paste(iter,"is a perfect number",sep=" ")) ## leave a blank space
between number and string
    count<-count+1
  }#end if

  iter<-iter+1
}#end while loop
```

```
## [1] "6 is a perfect number"
## [1] "28 is a perfect number"
```

## Question 2.b.1

Solution:

```
date()
```

```
## [1] "Wed Jan  1 17:40:55 2020"
```

```
num.perfect<-2
count<-1 ## change the begin number to get first two perfect number rather
than first three perfect number
iter<-2



while(count<=num.perfect){
  divisor<-1
  for(i in 2:iter){ ## when i in 2:iter , i could be the divisor of iter
       if(iter%%i==0 & iter!=i) ## the i could not be the iter in order to
get correct sum of i
         divisor<-c(divisor,i)
         i<-i+1## add this line to iterrate add the i , and make this loop
could be finished
       }# end for loop

       if(sum(divisor)==iter){
    print(paste(iter,"is a perfect number",sep=" ")) ## leave a blank space
between number and string
    count<-count+1
  }#end if

  iter<-iter+1
}#end while loop
```

```
## [1] "6 is a perfect number"
## [1] "28 is a perfect number"
```

```
date()
```

```
## [1] "Wed Jan  1 17:40:55 2020"
```

1 sec

## Question 2.b.2

Solution:

```
date()
```

```
## [1] "Wed Jan  1 17:40:55 2020"
```

```
num.perfect<-4
count<-1 ## change the begin number to get first two perfect number rather
than first three perfect number
iter<-2


while(count<=num.perfect){
  divisor<-1
  for(i in 2:iter){ ## when i in 2:iter , i could be the divisor of iter
       if(iter%%i==0 & iter!=i) ## the i could not be the iter in order to
get correct sum of i
         divisor<-c(divisor,i)
         i<-i+1## add this line to iterrate add the i , and make this loop
could be finished
       }# end for loop

       if(sum(divisor)==iter){
    print(paste(iter,"is a perfect number",sep=" ")) ## leave a blank space
between number and string
    count<-count+1
  }#end if

  iter<-iter+1
}#end while loop
```

```
## [1] "6 is a perfect number"
## [1] "28 is a perfect number"
## [1] "496 is a perfect number"
## [1] "8128 is a perfect number"
```

```
date()
```

```
## [1] "Wed Jan  1 17:41:06 2020"
```

```
v<-1:4
time.sec<-c(1,1,1,37)
time.sec
```

```
## [1]  1  1  1 37
```

```
runTime<-cbind(v,time.sec)
runTime
```

```
##      v time.sec
## [1,] 1        1
## [2,] 2        1
## [3,] 3        1
## [4,] 4       37
```

```r
colnames(runTime) <- c("No.perfect number", "waiting time (s)")
runTime
```

```
##      No.perfect number waiting time (s)
## [1,]                 1                1
## [2,]                 2                1
## [3,]                 3                1
## [4,]                 4               37
```

```r
kable(runTime, caption="Run Time Result")
```

*Run Time Result*

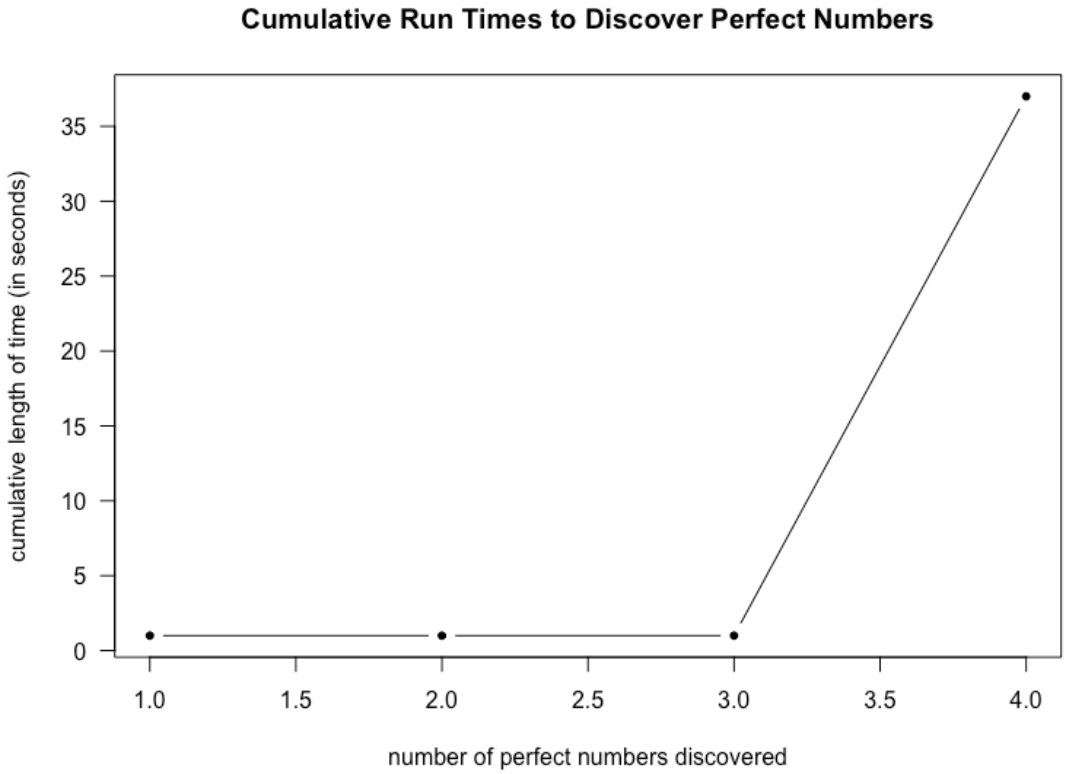
| No.perfect number | waiting time (s) |
|---|---|
| 1 | 1 |
| 2 | 1 |
| 3 | 1 |
| 4 | 37 |

[1] “Sat Sep 15 00:27:16 2018” [1] “6 is a perfect number” [1] “28 is a perfect number” [1] “496 is a perfect number” [1] “8128 is a perfect number” [1] “Sat Sep 15 00:28:04 2018”

## Question 2.c

Solution:

```r
theme.info <- theme(plot.title = element_text(size=30, face = "bold",
hjust=0.5),
                    axis.title = element_text(size=12),
                    axis.text=element_text(size=20, face = "bold"))
x<-1:4
y<-c(1,1,1,37)
plot(x,y,pch=20,type = "b",
     xlab = "number of perfect numbers discovered",
     ylab = "cumulative length of time (in seconds)",
     main = "Cumulative Run Times to Discover Perfect Numbers",
     las=TRUE)+theme.info
```




```
## NULL
```

Not Linear, it seems exponent function.

## Question 3.a

Solution:

```
x<-c(4,67,3)
x<-x[!is.na(x)]
sum.x<-1
x
```

```
## [1]  4 67  3
```

```
for (i in 1:length(x)){
  if(x[i]<=0){
    print('error')
    sum.x<-NaN
    break

  }
  else{
    sum.x<-sum.x*x[i]
```

```
  }
  }
geometric.mean<-sum.x^(1/length(x))
geometric.mean
```

```
## [1] 9.298624
```

## Question 3.b.1

Solution:

```
x<-c(NA,4,67,3)
x<-x[!is.na(x)]
sum.x<-1
x
```

```
## [1]  4 67  3
```

```
for (i in 1:length(x)){
  if(x[i]<=0){
    print('error')
    sum.x<-NaN
    break

  }
  else{
    sum.x<-sum.x*x[i]

  }
  }
geometric.mean<-sum.x^(1/length(x))
geometric.mean
```

```
## [1] 9.298624
```

## Question 3.b.2

Solution:

```
x<-c(0,NA,6)
x<-x[!is.na(x)]
sum.x<-1
x
```

```
## [1] 0 6
```

```
for (i in 1:length(x)){
  if(x[i]<=0){
    print('error')
    sum.x<-NaN
    break
```

```
  }
  else{
    sum.x<-sum.x*x[i]

  }
  }
```

```
## [1] "error"
```

```
geometric.mean<-sum.x^(1/length(x))
geometric.mean
```

```
## [1] NaN
```

## Question 3.b.3

Solution:

```
x<-c(67,3,Inf)
x<-x[!is.na(x)]
sum.x<-1
x
```

```
## [1]  67   3 Inf
```

```
for (i in 1:length(x)){
  if(x[i]<=0){
    print('error')
    sum.x<-NaN
    break

  }
  else{
    sum.x<-sum.x*x[i]

  }
  }
geometric.mean<-sum.x^(1/length(x))
geometric.mean
```

```
## [1] Inf
```

## Question 3.b.4

```
x<-c(67,3,-Inf)
x<-x[!is.na(x)]
sum.x<-1
x
```

```
## [1]   67    3 -Inf
```

```r
for (i in 1:length(x)){
  if(x[i]<=0){
    print('error')
    sum.x<-NaN
    break

  }
  else{
    sum.x<-sum.x*x[i]

  }
  }
```

```
## [1] "error"
```

```r
geometric.mean<-sum.x^(1/length(x))
geometric.mean
```

```
## [1] NaN
```