Bios 6301: HW 2 Jordan Clark

Jordan Clark

(informally) Due Thursday, 17 September, 1:00 PM

50 points total.

This assignment won't be submitted until we've covered Rmarkdown. Create R chunks for each question and insert your R code appropriately. Check your output by using the `Knit PDF` button in RStudio.

1. **Working with data** In the `datasets` folder on the course GitHub repo, you will find a file called `cancer.csv`, which is a dataset in comma-separated values (csv) format. This is a large cancer incidence dataset that summarizes the incidence of different cancers for various subgroups. (18 points)

    1. Load the data set into R and make it a data frame called `cancer.df`. (2 points)

```
setwd("~/Desktop/Bios6301")
getwd()
```

```
## [1] "/Users/Kasasa/Desktop/Bios6301"
```

```
cancer.df<-read.csv("./datasets/cancer.csv")
head(cancer.df)
```

```
##   year                           site   state    sex     race mortality
## 1 1999 Brain and Other Nervous System alabama Female    Black      0.00
## 2 1999 Brain and Other Nervous System alabama Female Hispanic      0.00
## 3 1999 Brain and Other Nervous System alabama Female    White     83.67
## 4 1999 Brain and Other Nervous System alabama   Male    Black      0.00
## 5 1999 Brain and Other Nervous System alabama   Male Hispanic      0.00
## 6 1999 Brain and Other Nervous System alabama   Male    White    103.66
##   incidence population
## 1        19     623475
## 2         0      28101
## 3       110    1640665
## 4        18     539198
## 5         0      37082
## 6       145    1570643
```

```
names(cancer.df)
```

```
## [1] "year"       "site"       "state"      "sex"        "race"
## [6] "mortality"  "incidence"  "population"
```

    2. Determine the number of rows and columns in the data frame. (2)

```
dim(cancer.df)
```

```
## [1] 42120     8
```

```
   #8 columns and 42120 rows
```

3. Extract the names of the columns in `cancer.df`. (2)

```
col.names<-c(colnames(cancer.df))
col.names
```

```
## [1] "year"       "site"       "state"      "sex"        "race"
## [6] "mortality"  "incidence"  "population"
```

4. Report the value of the 3000th row in column 6. (2)

```
print(cancer.df[3000,6])
```

```
## [1] 350.69
```

5. Report the contents of the 172nd row. (2)

```
print(cancer.df[172,])
```

```
##     year                           site  state  sex  race mortality
## 172 1999 Brain and Other Nervous System nevada Male Black         0
##     incidence population
## 172         0      73172
```

6. Create a new column that is the incidence *rate* (per 100,000) for each row.(3)

```
cancer.df<-transform(cancer.df, i.rate=incidence/100000 )
head(cancer.df)
```

```
##   year                           site   state    sex     race mortality
## 1 1999 Brain and Other Nervous System alabama Female    Black      0.00
## 2 1999 Brain and Other Nervous System alabama Female Hispanic      0.00
## 3 1999 Brain and Other Nervous System alabama Female    White     83.67
## 4 1999 Brain and Other Nervous System alabama   Male    Black      0.00
## 5 1999 Brain and Other Nervous System alabama   Male Hispanic      0.00
## 6 1999 Brain and Other Nervous System alabama   Male    White    103.66
##   incidence population  i.rate
## 1        19     623475 0.00019
## 2         0      28101 0.00000
## 3       110    1640665 0.00110
## 4        18     539198 0.00018
## 5         0      37082 0.00000
## 6       145    1570643 0.00145
```

7. How many subgroups (rows) have a zero incidence rate? (2)

```
zero<-0.0
zero
```

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

```
cancer0.df<-cancer.df[cancer.df$i.rate %in% zero,]
head(cancer0.df)
```

```
##    year                           site   state    sex     race mortality
## 2  1999 Brain and Other Nervous System alabama Female Hispanic         0
## 5  1999 Brain and Other Nervous System alabama   Male Hispanic         0
## 7  1999 Brain and Other Nervous System  alaska Female    Black         0
## 8  1999 Brain and Other Nervous System  alaska Female Hispanic         0
## 9  1999 Brain and Other Nervous System  alaska Female    White         0
## 10 1999 Brain and Other Nervous System  alaska   Male    Black         0
##    incidence population i.rate
## 2          0      28101      0
## 5          0      37082      0
## 7          0      12710      0
## 8          0      11664      0
## 9          0     220036      0
## 10         0      13900      0
```

8. Find the subgroup with the highest incidence rate.(3)

```
which.max(cancer.df[,'i.rate'])
```

```
## [1] 21387
```

```
head(order(cancer.df[,'i.rate'],decreasing=T))
```

```
## [1] 21387 14367   327  7347 26847 28407
```

```
#Looks like row 21387
    cancer.df[21387,]
```

```
##       year   site      state    sex  race mortality incidence population
## 21387 2002 Breast california Female White   3463.74     18774   13690681
##        i.rate
## 21387 0.18774
```

2. **Data types** (10 points)

    1. Create the following vector: `x <- c("5","12","7")`. Which of the following commands will produce an error message? For each command, Either explain why they should be errors, or explain the non-erroneous result. (4 points)

```
x <- c("5","12","7")
class(x)
```

```
## [1] "character"
```

```
max(x)
```

```
## [1] "7"
```

```
#It returns a 7 because the elements are characters. 12 is not the max...
#... because it starts with 1. So max returns a pseudo-alphabetic order.
sort(x)
```

```
## [1] "12" "5"  "7"
```

```
#Because our elements are characters, '1' is like 'a', '5' is like 'e' ...
#... and '7' is like 'g', and they are ordered pseudo-alphabetically.
#(sum(x)) #I'm commenting this out because it won't compile otherwise.
#It is an error because you cannot sum characters like you can numerically.
```

2. For the next two commands, either explain their results, or why they should produce errors. (3 points

```
            y <- c("5",7,12)
    class(y[2])
```

```
## [1] "character"
```

```
 #           y[2] + y[3]
      #It appears that the elements of this vector have taken the class...
    #...of the first element, which is a character.
```

3. For the next two commands, either explain their results, or why they should produce errors. (3 points

```
            z <- data.frame(z1="5",z2=7,z3=12)
            z[1,2] + z[1,3]
```

```
## [1] 19
```

```
    #It appears that this worked because it's a data frame, rather than a vector.
```

3. **Data structures** Give R expressions that return the following matrices and vectors (*i.e.* do not construct them manually). (3 points each, 12 total)

    1. $(1, 2, 3, 4, 5, 6, 7, 8, 7, 6, 5, 4, 3, 2, 1)$

```
xx<-(1:8)
yy<-rev(xx)
zz<-append(xx,yy)
zz<-zz[-8]
zz
```

```
##  [1] 1 2 3 4 5 6 7 8 7 6 5 4 3 2 1
```

```
  #somewhat manually constructed?
```

2. $(1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5)$

```
rep(1:5, 1:5)
```

```
##  [1] 1 2 2 3 3 3 4 4 4 4 5 5 5 5 5
```

```
3. $\begin{pmatrix}
  0 & 1 & 1  \\
  1 & 0 & 1  \\
  1 & 1 & 0  \\
\end{pmatrix}$
```

```
  g<-c(0,1)
  f<-c(0,1)
  h<-c(1,0)
  i<-expand.grid(g,f,h)
  i
```

```
##   Var1 Var2 Var3
## 1    0    0    1
## 2    1    0    1
## 3    0    1    1
## 4    1    1    1
## 5    0    0    0
## 6    1    0    0
## 7    0    1    0
## 8    1    1    0
```

```
  #This is obviously too much.
  i.i<-i[which(rowSums(i)==2),]
  i.i
```

```
##   Var1 Var2 Var3
## 2    1    0    1
## 3    0    1    1
## 8    1    1    0
```

```
    #Just reorder this puppy
  i.ii<-i.i[,c(2,1,3)]
    i.ii
```

```
##   Var2 Var1 Var3
## 2    0    1    1
## 3    1    0    1
## 8    1    1    0
```

```
    #Meandering solution but it worked.
```

```
4. $\begin{pmatrix}
  1 & 2 & 3 & 4 \\
  1 & 4 & 9 & 16 \\
  1 & 8 & 27 & 64  \\
  1 & 16 & 81 & 256 \\
  1 & 32 & 243 & 1024  \\
\end{pmatrix}$
```

```
n=5
g<-matrix(0,nrow=n,ncol=4)
a<-rep(1:n,1)
a
```

```
## [1] 1 2 3 4 5
```

```
x<-c(1,2,3,4)
for (j in 1:n)
{g[j,]<-(x^j)
}
g
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    2    3    4
## [2,]    1    4    9   16
## [3,]    1    8   27   64
## [4,]    1   16   81  256
## [5,]    1   32  243 1024
```

```
#Absolutely ridiculous. I'm shocked that I actually did that.
```

4. **Basic programming** (10 points)

    1. Let $h(x,n) = 1 + x + x^2 + \ldots + x^n = \sum_{i=0}^{n} x^i$. Write an R program to calculate $h(x,n)$ using a `for` loop. (5 points)

```
x<-2
h<-2
n<-5
x.fun<-rep(0,n)
for(i in 0:(n-1)) {
  x.fun[i]<-x^(i+1)
  y<-1+x+sum(x.fun)
}
x.fun
```

```
## [1]  4  8 16 32  0
```

```
y
```

```
## [1] 63
```

```
1+h+h**2+h**3+h**4+h**5
```

```
## [1] 63
```

```
sum(x^(0:n))
```

```
## [1] 63
```

```
#Here's an easy non-loop way to do it.
rowSums(outer(x, 0:n, "^"))
```

```
## [1] 63
```

```
#Here's another super cool way to do it.
```

```
1. If we list all the natural numbers below 10 that are multiples of 3 or 5, we get 3, 5, 6 and 9. The s
```

1. Find the sum of all the multiples of 3 or 5 below 1,000. (3, euler1)

```
#Here's my best shot
x<-numeric(999)
#creating a vector that holds the values
for(i in 1:999) {
if (i %%3 == 0 || i%%5==0) {
x[i]<-i} else {x[i]<-0}}
x
```

```
##   [1]   0   0   3   0   5   6   0   0   9  10   0  12   0   0  15   0   0
##  [18]  18   0  20  21   0   0  24  25   0  27   0   0  30   0   0  33   0
##  [35]  35  36   0   0  39  40   0  42   0   0  45   0   0  48   0  50  51
##  [52]   0   0  54  55   0  57   0   0  60   0   0  63   0  65  66   0   0
##  [69]  69  70   0  72   0   0  75   0   0  78   0  80  81   0   0  84  85
##  [86]   0  87   0   0  90   0   0  93   0  95  96   0   0  99 100   0 102
## [103]   0   0 105   0   0 108   0 110 111   0   0 114 115   0 117   0   0
## [120] 120   0   0 123   0 125 126   0   0 129 130   0 132   0   0 135   0
## [137]   0 138   0 140 141   0   0 144 145   0 147   0   0 150   0   0 153
## [154]   0 155 156   0   0 159 160   0 162   0   0 165   0   0 168   0 170
## [171] 171   0   0 174 175   0 177   0   0 180   0   0 183   0 185 186   0
## [188]   0 189 190   0 192   0   0 195   0   0 198   0 200 201   0   0 204
## [205] 205   0 207   0   0 210   0   0 213   0 215 216   0   0 219 220   0
## [222] 222   0   0 225   0   0 228   0 230 231   0   0 234 235   0 237   0
## [239]   0 240   0   0 243   0 245 246   0   0 249 250   0 252   0   0 255
## [256]   0   0 258   0 260 261   0   0 264 265   0 267   0   0 270   0   0
## [273] 273   0 275 276   0   0 279 280   0 282   0   0 285   0   0 288   0
## [290] 290 291   0   0 294 295   0 297   0   0 300   0   0 303   0 305 306
## [307]   0   0 309 310   0 312   0   0 315   0   0 318   0 320 321   0   0
## [324] 324 325   0 327   0   0 330   0   0 333   0 335 336   0   0 339 340
## [341]   0 342   0   0 345   0   0 348   0 350 351   0   0 354 355   0 357
## [358]   0   0 360   0   0 363   0 365 366   0   0 369 370   0 372   0   0
## [375] 375   0   0 378   0 380 381   0   0 384 385   0 387   0   0 390   0
```

```
## [392]   0 393   0 395 396   0   0 399 400   0 402   0   0 405   0   0 408
## [409]   0 410 411   0   0 414 415   0 417   0   0 420   0   0 423   0 425
## [426] 426   0   0 429 430   0 432   0   0 435   0   0 438   0 440 441   0
## [443]   0 444 445   0 447   0   0 450   0   0 453   0 455 456   0   0 459
## [460] 460   0 462   0   0 465   0   0 468   0 470 471   0   0 474 475   0
## [477] 477   0   0 480   0   0 483   0 485 486   0   0 489 490   0 492   0
## [494]   0 495   0   0 498   0 500 501   0   0 504 505   0 507   0   0 510
## [511]   0   0 513   0 515 516   0   0 519 520   0 522   0   0 525   0   0
## [528] 528   0 530 531   0   0 534 535   0 537   0   0 540   0   0 543   0
## [545] 545 546   0   0 549 550   0 552   0   0 555   0   0 558   0 560 561
## [562]   0   0 564 565   0 567   0   0 570   0   0 573   0 575 576   0   0
## [579] 579 580   0 582   0   0 585   0   0 588   0 590 591   0   0 594 595
## [596]   0 597   0   0 600   0   0 603   0 605 606   0   0 609 610   0 612
## [613]   0   0 615   0   0 618   0 620 621   0   0 624 625   0 627   0   0
## [630] 630   0   0 633   0 635 636   0   0 639 640   0 642   0   0 645   0
## [647]   0 648   0 650 651   0   0 654 655   0 657   0   0 660   0   0 663
## [664]   0 665 666   0   0 669 670   0 672   0   0 675   0   0 678   0 680
## [681] 681   0   0 684 685   0 687   0   0 690   0   0 693   0 695 696   0
## [698]   0 699 700   0 702   0   0 705   0   0 708   0 710 711   0   0 714
## [715] 715   0 717   0   0 720   0   0 723   0 725 726   0   0 729 730   0
## [732] 732   0   0 735   0   0 738   0 740 741   0   0 744 745   0 747   0
## [749]   0 750   0   0 753   0 755 756   0   0 759 760   0 762   0   0 765
## [766]   0   0 768   0 770 771   0   0 774 775   0 777   0   0 780   0   0
## [783] 783   0 785 786   0   0 789 790   0 792   0   0 795   0   0 798   0
## [800] 800 801   0   0 804 805   0 807   0   0 810   0   0 813   0 815 816
## [817]   0   0 819 820   0 822   0   0 825   0   0 828   0 830 831   0   0
## [834] 834 835   0 837   0   0 840   0   0 843   0 845 846   0   0 849 850
## [851]   0 852   0   0 855   0   0 858   0 860 861   0   0 864 865   0 867
## [868]   0   0 870   0   0 873   0 875 876   0   0 879 880   0 882   0   0
## [885] 885   0   0 888   0 890 891   0   0 894 895   0 897   0   0 900   0
## [902]   0 903   0 905 906   0   0 909 910   0 912   0   0 915   0   0 918
## [919]   0 920 921   0   0 924 925   0 927   0   0 930   0   0 933   0 935
## [936] 936   0   0 939 940   0 942   0   0 945   0   0 948   0 950 951   0
## [953]   0 954 955   0 957   0   0 960   0   0 963   0 965 966   0   0 969
## [970] 970   0 972   0   0 975   0   0 978   0 980 981   0   0 984 985   0
## [987] 987   0   0 990   0   0 993   0 995 996   0   0 999
```

```
#this is a vector with all the multiples, with 0's in non-multiples
y<-x[x !=0]
#remove the zero's
y
```

```
##   [1]   3   5   6   9  10  12  15  18  20  21  24  25  27  30  33  35  36
##  [18]  39  40  42  45  48  50  51  54  55  57  60  63  65  66  69  70  72
##  [35]  75  78  80  81  84  85  87  90  93  95  96  99 100 102 105 108 110
##  [52] 111 114 115 117 120 123 125 126 129 130 132 135 138 140 141 144 145
##  [69] 147 150 153 155 156 159 160 162 165 168 170 171 174 175 177 180 183
##  [86] 185 186 189 190 192 195 198 200 201 204 205 207 210 213 215 216 219
## [103] 220 222 225 228 230 231 234 235 237 240 243 245 246 249 250 252 255
## [120] 258 260 261 264 265 267 270 273 275 276 279 280 282 285 288 290 291
## [137] 294 295 297 300 303 305 306 309 310 312 315 318 320 321 324 325 327
## [154] 330 333 335 336 339 340 342 345 348 350 351 354 355 357 360 363 365
## [171] 366 369 370 372 375 378 380 381 384 385 387 390 393 395 396 399 400
## [188] 402 405 408 410 411 414 415 417 420 423 425 426 429 430 432 435 438
```

```
## [205] 440 441 444 445 447 450 453 455 456 459 460 462 465 468 470 471 474
## [222] 475 477 480 483 485 486 489 490 492 495 498 500 501 504 505 507 510
## [239] 513 515 516 519 520 522 525 528 530 531 534 535 537 540 543 545 546
## [256] 549 550 552 555 558 560 561 564 565 567 570 573 575 576 579 580 582
## [273] 585 588 590 591 594 595 597 600 603 605 606 609 610 612 615 618 620
## [290] 621 624 625 627 630 633 635 636 639 640 642 645 648 650 651 654 655
## [307] 657 660 663 665 666 669 670 672 675 678 680 681 684 685 687 690 693
## [324] 695 696 699 700 702 705 708 710 711 714 715 717 720 723 725 726 729
## [341] 730 732 735 738 740 741 744 745 747 750 753 755 756 759 760 762 765
## [358] 768 770 771 774 775 777 780 783 785 786 789 790 792 795 798 800 801
## [375] 804 805 807 810 813 815 816 819 820 822 825 828 830 831 834 835 837
## [392] 840 843 845 846 849 850 852 855 858 860 861 864 865 867 870 873 875
## [409] 876 879 880 882 885 888 890 891 894 895 897 900 903 905 906 909 910
## [426] 912 915 918 920 921 924 925 927 930 933 935 936 939 940 942 945 948
## [443] 950 951 954 955 957 960 963 965 966 969 970 972 975 978 980 981 984
## [460] 985 987 990 993 995 996 999
```

```
#this is a vector with just the multiples
sum(y)
```

```
## [1] 233168
```

```
#The answer is 233168
```

1. Find the sum of all the multiples of 4 or 7 below 1,000,000. (2)

```
#Same deal as before, just different multiples.
a<-numeric(999999)
for(i in 1:999999){
if (i %%4 ==0 || i%%7==0){
a[i]<-i} else {a[i]<-0}
}
head(a)
```

```
## [1] 0 0 0 4 0 0
```

```
b<-a[a !=0]
head(b)
```

```
## [1]  4  7  8 12 14 16
```

```
sum(b)
```

```
## [1] 178571071431
```

```
#answer is 178571071431
#You probably won't like the manual entry and deletion of 0 terms...
#... but this is the way I found that works.
```

1. Each new term in the Fibonacci sequence is generated by adding the previous two terms. By starting with 1 and 2, the first 10 terms will be $(1, 2, 3, 5, 8, 13, 21, 34, 55, 89)$. Write an R program to calculate the sum of the first 15 even-valued terms. (5 bonus points, euler2)

```
#Going to try to determine the 15th even value term first
bin <- 45
fibvals <- numeric(bin)
fibvals[1] <- 1
fibvals[2] <- 1
for (i in 3:bin) {
   fibvals[i] <- fibvals[i-1]+fibvals[i-2]
}
fibvals
```

```
##  [1]          1          1          2          3          5          8
##  [7]         13         21         34         55         89        144
## [13]        233        377        610        987       1597       2584
## [19]       4181       6765      10946      17711      28657      46368
## [25]      75025     121393     196418     317811     514229     832040
## [31]    1346269    2178309    3524578    5702887    9227465   14930352
## [37]   24157817   39088169   63245986  102334155  165580141  267914296
## [43]  433494437  701408733 1134903170
```

```
#Here are the first 45 values of the fibonacci sequence
#(I specified bin with some prior knowledge that fib. sequence produces...
#...an even number every 3rd term. Might be cheating a little)

c<-numeric(15)
for(i in seq(fibvals)) {
if(fibvals[i]%%2==0) {
c[i]<-fibvals[i]} else{c[i]<-0}
}
head(c)
```

```
## [1] 0 0 2 0 0 8
```

```
c
```

```
##  [1]          0          0          2          0          0          8
##  [7]          0          0         34          0          0        144
## [13]          0          0        610          0          0       2584
## [19]          0          0      10946          0          0      46368
## [25]          0          0     196418          0          0     832040
## [31]          0          0    3524578          0          0   14930352
## [37]          0          0   63245986          0          0  267914296
## [43]          0          0 1134903170
```

```
ci<-c[c !=0]
length(ci)
```

```
## [1] 15
```

```
ci
```

```
##  [1]          2          8         34        144        610       2584
##  [7]      10946      46368     196418     832040    3524578   14930352
## [13]   63245986  267914296 1134903170
```

```r
sum(ci)
```

```
## [1] 1485607536
```

```r
#The answer is 1485607536.
#Definitely not the most efficient way of doing it, but given that ...
#... I have almost no prior experience with loops, I'm happy with it.
```

Some problems taken or inspired by projecteuler.