HW4

Chris Porter

Saturday, October 01, 2016

1. Create the Vectors:

a) (1,2,3,. . . ,19,20)

I'll use the combine (c) generic funtion to combine the list from 1 to 20.

```
a1 <- c(1:20)
a1
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

b) (20,19,. . . ,2,1)

I'll use the combine (c) generic funtion to combine the list from 20 to 1.

```
b1 <- c(20:1)
b1
```

```
##  [1] 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1
```

c) (1,2,3,. . . ,19,20,19,18,. . . ,2,1)

I'll use the combine (c) generic funtion to combine the list from 1 to 20 and another list from 19 to 1.

```
c1 <- c(1:20,19:1)
c1
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 19 18 17
## [24] 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1
```

d) (4,6,3) and assign it to the name tmp

I 'll use the combine ('c') generic function to combine a list of 4,6 and 3.

```
tmp <- c(4,6,3)
tmp
```

```
## [1] 4 6 3
```

e) (4,6,3,4,6,3,. . . ,4,6,3) where there are 10 occurences of 4.

I'll use the rep function to repeat the vector tmp ten times

```r
e1 <- rep(tmp,10)
e1
```

```
##  [1] 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3
```

f) (4,6,3,4,6,3,. . . ,4,6,3,4) where there are 11 occurences of 4, 10 occurrences of 6 and 10 occurrences of 3.

I'll use the rep function to repeat the vector tmp ten times, and then add a 4 to the end of the vector

```r
f1 <- c(rep(tmp,10),4)
f1
```

```
##  [1] 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4
```

g) (4,4,. . . ,4,6,6,. . . ,6,3,3,. . . ,3) where there are 10 occurrences of 4, 20 occurrences of 6 and 30 occurrences of 3.

I'll use 3 instances of the rep function in conjunction with the combine function to create this vector.

```r
g1 <- c(rep(4,10),rep(6,20),rep(3,30))
g1
```

```
##  [1] 4 4 4 4 4 4 4 4 4 4 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 3 3 3 3 3
## [36] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
```

2. Create a vector of the values of e^xcos(x) at x = 3,3.1,3.2,. . . ,5.9,6.

I'll use the sequence function to create a vector of values from 3 to 6 in increments of one tenth. Then define a function to calculate the product of the exponent and the cosine

```r
x <- seq(3,6,.1)
x
```

```
##  [1] 3.0 3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 4.0 4.1 4.2 4.3 4.4 4.5 4.6
## [18] 4.7 4.8 4.9 5.0 5.1 5.2 5.3 5.4 5.5 5.6 5.7 5.8 5.9 6.0
```

```r
eXcosX <- function(x) exp(x) * cos(x)
eXcosX(x)
```

```
##  [1] -19.884531 -22.178753 -24.490697 -26.773182 -28.969238 -31.011186
##  [7] -32.819775 -34.303360 -35.357194 -35.862834 -35.687732 -34.685042
## [13] -32.693695 -29.538816 -25.032529 -18.975233 -11.157417  -1.362099
## [19]  10.632038  25.046705  42.099201  61.996630  84.929067 111.061586
## [25] 140.525075 173.405776 209.733494 249.468441 292.486707 338.564378
## [31] 387.360340
```

3. Create the following vectors:

a) $(0.1^3 0.2^1, 0.1^6 0.2^4, 0.1^9 0.2^7, \ldots, 0.1^{36} 0.2^{34})$

I'll use the repeat function to create vectors of 1 tenth and 2 tenths, and then use the sequence function to create vectors from 3 to 36 and 1 to 34 by 3, then calculate the product of the two exponent vaules using the correct combination of our 4 vectors, all of length 12.

```
tenth <- rep(0.1, 12)
twentieth <- rep(0.2,12)
expA <- seq(3,36,3)
expB <- seq(1,34,3)
a3<- tenth^expA*twentieth^expB
a3
```

```
##  [1] 2.000000e-04 1.600000e-09 1.280000e-14 1.024000e-19 8.192000e-25
##  [6] 6.553600e-30 5.242880e-35 4.194304e-40 3.355443e-45 2.684355e-50
## [11] 2.147484e-55 1.717987e-60
```

b) $(\mathbf{2}, \frac{2^2}{2}, \frac{2^3}{3}, \ldots, \frac{2^25}{25})$

I'll create two vectors, one consisting of 2's and another consisting of the series from 1 to 25, by increments of 1. Then use the element wise exponentiation and division to calculate the vector asked in the question.

```
numerator <- rep(2,25)
expDenom <- seq(1,25,1)
b3 <- numerator^expDenom/expDenom
b3
```

```
##  [1] 2.000000e+00 2.000000e+00 2.666667e+00 4.000000e+00 6.400000e+00
##  [6] 1.066667e+01 1.828571e+01 3.200000e+01 5.688889e+01 1.024000e+02
## [11] 1.861818e+02 3.413333e+02 6.301538e+02 1.170286e+03 2.184533e+03
## [16] 4.096000e+03 7.710118e+03 1.456356e+04 2.759411e+04 5.242880e+04
## [21] 9.986438e+04 1.906502e+05 3.647221e+05 6.990507e+05 1.342177e+06
```

4. Calculate the following:

a) $\sum_{i=10}^{100}(i^3 + 4i^2)$

Create a vector from 10 to 100 by 1, and a function which adds the cube and the square of twice our independent variable

```
i <- seq(10,100,1)
i
```

```
##  [1]  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26
## [18]  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43
## [35]  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
## [52]  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77
## [69]  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94
## [86]  95  96  97  98  99 100
```

```
cubeQuadSquare <- function(x) (x^3 + 4*x^2)
x <- cubeQuadSquare(i)
print(sum(cubeQuadSquare(seq(10,100,1))))
```

```
## [1] 26852735
```

```
print(sum(x))
```

```
## [1] 26852735
```

b) $\sum_{i=1}^{25}(\frac{2^i}{i} + \frac{3^i}{i^2})$

I'll define the function above in r, then calculate the sum from 1 to 25, by 1, and print the result

```
func4b <- function(x) ((2^x/x)+(3^x/x^2))
print(sum(func4b(seq(1,25,1))))
```

```
## [1] 2129170437
```

5. Use the function paste to create the following character vectors of length 30:

a) ("label 1", "label 2", . . . , "label 30"). Note that there is a single space between label and the number following.

I'll use paste to combine the text 'Label' with a list from 1 to 30, separated by a space.

```
a5 <- paste("Label", c(1:30), sep=" ")
a5
```

```
##  [1] "Label 1"  "Label 2"  "Label 3"  "Label 4"  "Label 5"  "Label 6"
##  [7] "Label 7"  "Label 8"  "Label 9"  "Label 10" "Label 11" "Label 12"
## [13] "Label 13" "Label 14" "Label 15" "Label 16" "Label 17" "Label 18"
## [19] "Label 19" "Label 20" "Label 21" "Label 22" "Label 23" "Label 24"
## [25] "Label 25" "Label 26" "Label 27" "Label 28" "Label 29" "Label 30"
```

b) ("fn1", "fn2", . . . , "fn30"). In this case, there is no space between fn and the number following.

I'll use paste to combine the text 'fn' with a list from 1 to 30, separated by no space.

```
b5<- paste("fn",c(1:30),sep="")
b5
```

```
##  [1] "fn1"  "fn2"  "fn3"  "fn4"  "fn5"  "fn6"  "fn7"  "fn8"  "fn9"  "fn10"
## [11] "fn11" "fn12" "fn13" "fn14" "fn15" "fn16" "fn17" "fn18" "fn19" "fn20"
## [21] "fn21" "fn22" "fn23" "fn24" "fn25" "fn26" "fn27" "fn28" "fn29" "fn30"
```

6. Execute the following lines which create two vectors of random integers which are chosen with replacement from the integers 0,1,. . . ,999. Both vectors have length 250.

```
set.seed(50)
xVec <- sample(0:999, 250, replace=T)
yVec <- sample(0:999, 250, replace=T)
```

```
set.seed(50)
xVec <- sample(0:999, 250, replace=T)
yVec <- sample(0:999, 250, replace=T)
```

Suppose x=(x1,x2,. . . ,xn) denotes the vector xVec and y=(y1,y2,. . . ,yn) denotes the vector yVec.

a)Create the vector $(y_2 - x_1, ..., y_n - x_{n-1})$

I create two new vectors of the same length, but one is the xVec from 1 to n-1, and the other is the yVec from 2 to n, Then I will use element-wise subtraction to create the new vector.

```
x <- xVec[1:length(xVec)-1]
y <- yVec[2:length(yVec)]
a6 <- y-x
a6
```

```
##   [1]  163 -122  317 -146  417  393  249 -489  741  771   81  402 -549  338
##  [15]  583 -403  -67  217  307 -121 -269   36 -706 -563  102   48  397  297
##  [29]  -45 -152  497  405  339 -400  499  -89  211 -670   87   74  554  149
##  [43] -183  612  193 -453  -70 -141  127 -709 -708 -722  -64  388 -184 -212
##  [57]  242  430  275  672 -150  275  -96 -255  512  577  264  439  149 -916
##  [71]  374 -889 -332  324 -553  394  -87  -75  345 -735  -55  100  -40   15
##  [85]  279  409  790 -547 -487 -399 -619 -168 -185   19  645  551  227 -366
##  [99]  242  147  247 -499 -614  758   63 -227  247  379 -472  566 -762  152
## [113]  493  360   69  190  544 -176  216 -676 -205  782 -109  189 -233  505
## [127] -219  288  -57  487  256  300 -192 -263  704  674  217  280   17  -68
## [141]  259  612 -127    1  545 -231 -191 -338  333  495  -21   -4  294 -668
## [155] -814  420  793  631  -67  655  143  611 -220 -518 -285  327  523  -13
## [169] -679 -241   39  193  342  588  469   68  895 -658  232 -331   27  441
## [183] -733 -182 -399   79 -469  371  475  265 -407  211   59 -974  -90  218
## [197]  396 -486 -963 -327  425  220  128  235  294 -107 -365  146 -588  449
## [211] -434  221  846  386 -910  161  206  109  712 -334 -434    7  640 -350
## [225]  923  353 -579  225  327  410  568 -195  -83  154 -486 -195  667 -144
## [239]  272  410  546  380 -559  414  674  193  222  -92  553
```

b) Create the vector $(\frac{sin(y_1)}{cos(x_2)}, \frac{sin(y_2)}{cos(x_3)}, ..., \frac{sin(y_{n-1})}{cos(x_n)})$

I create two new vectors of the same length, but one is the yVec from 1 to n-1, and the other is the Vec from 2 to n, Then I will use element-wise division on the sin of y divided by the cosine of x to create the new vector.

```
y <- yVec[1:length(yVec)-1]
x <- xVec[2:length(xVec)]
b6 <- sin(y) / cos(x)
b6
```

```
##   [1]   0.88603405  -1.44184825   0.82807258  -1.61591717  -0.86017343
##   [6]  20.26356465  -0.79930406   1.72414444  -0.08094240  -0.74895634
##  [11]  -2.59866958  -0.37361045  31.11471579   0.12355916  -0.35925226
##  [16]  -0.90743608   0.34374436   5.78205917  -2.57418558  -0.78661325
##  [21]  -0.59855406   0.98936263   0.33042931  -1.75124647  -0.59435547
##  [26]   1.05374692   0.65497397  -0.11596582  -0.97176537   0.57180267
##  [31]   0.75799030  -0.49259143  -0.99433357   0.05377148  -3.77616264
##  [36]  20.54902944   0.77784817   1.28146891  -0.51650728   6.66902699
##  [41]  -0.92970072 -10.93066299  -3.13102962  30.87943423  -1.14281543
##  [46]   0.36757630   1.18479716   0.94594159   0.93339520   0.93632658
##  [51] -11.05384468   2.76893270   0.97488334  -0.08932225  -1.33616578
##  [56]  -3.30065552   0.62663162  -1.96486337   0.08653876   0.56695489
##  [61]  44.07630714  -1.11764853   0.11230330  -0.46073106  -0.13860882
##  [66]   0.84026052   2.64708780  -1.63174570  -9.63022830  -2.15553419
##  [71]  -0.42770826   3.24955062  -4.23453154   0.93067452  -0.88388390
##  [76]   0.69339350   1.72841015  -8.22082884   1.69276461   1.02074555
##  [81]  -3.21968328  -0.90739226   1.11331935   0.59579467   0.19571363
##  [86]  -0.17975474   4.38929818   0.64431266  -1.54509170  -0.26536991
##  [91]  -0.81679156   1.34164181  -1.03400420  -1.33639979  -0.44444499
##  [96]   0.96777754  -0.09545121  -0.63686070  -2.30844090  -0.11384497
## [101]   1.08800453   1.06851885  -0.30428029  -1.77044888  -1.45269351
## [106]   0.97943716  -2.15021752   1.56128032   0.61018741   5.59692239
## [111]  -1.03020002  -1.14632240  -0.81548097   0.95359082  74.12815803
## [116]  -0.20329495  -0.08875385  -0.76023984  -0.42372635  -0.68385723
## [121]   1.28860542   0.94117702   1.89561343   0.69369539   4.15021756
## [126]  -1.08026240   1.26615554   0.02147428   3.32694398   0.22930300
## [131]   1.14217476   0.73847767   8.72339712 -17.15727240   0.90435970
## [136]   1.07791792   0.75391899  -0.26297571   0.83894657  -1.22542984
## [141]  -0.57277292  -1.22429033   2.10719833  -1.35745285  -0.84117115
## [146]  -0.69663176  -0.99207337  -1.17363312  -5.50814669  -1.12309426
## [151]   0.60767585   0.32903697  -0.08845387  -4.42251048  -1.31360561
## [156]  -1.05268827  -1.45007537  -1.03184453   0.38034305   2.06381128
## [161]  -1.64568068   0.47938401  46.18666528   1.75988821  14.03349520
## [166]   1.99884446  -1.02170635   1.02445028  -0.15250370  -1.11793279
## [171]  -4.12228606   1.02355677   0.89546497   0.74732250  -2.09533197
## [176]  -2.40630344  -0.73530615   0.90759126  -0.87474163  -4.22536917
## [181]  -2.04450866  -7.41320483   0.03607946  -0.85674969  -0.85648584
## [186]   2.58973778   8.68248704  -0.74202802   1.07347586   1.37638585
## [191]   1.73104746  -0.57596355  -0.49915725   0.11786229  -0.45584137
## [196]  -0.97726281  -6.86428063  -0.60929448  -0.72132361   0.00000000
## [201]   1.00734878   4.20789995  -0.81616263  -1.72455176  10.00784534
## [206]   0.71310632   8.77005056  -0.64297796   0.24086573  -6.12424634
## [211]   0.94848253   9.22132979  -5.85933168  -0.77292827  -0.85749485
## [216]   0.80000340 -10.45187777   2.91489552   0.86914823   0.93956496
## [221]   1.15020196  -4.25009579  -0.97278301   1.05669698  23.96919924
## [226]  -0.11659711   0.58615433  -1.23512544   1.08111948   3.37846777
## [231]   0.96204558  -1.18727215   0.77801767   2.39161655   1.01270315
## [236]   0.30508064  -1.13987140   1.35085069   2.13213714   0.95034702
## [241]   0.48941676  -1.03804260   1.11768517  -0.25446052 -15.07630921
## [246]   1.12429826   0.28067653  -0.75125301  -1.91160477
```

c) Create the vector $(x_1 + 2x_2 - x_3, ..., x_{n-2} + 2x_{n-1} - x_n)$

I create three new vectors of the same length, but one is the xVec from 2 to n-1 multiplied by 2, and the other is the xVec from 1 to n-2, and the third is the xVec from 3 to n. Then I will use element-wise addition and subtraction to calculate the new vector

```r
x1 <- xVec[1:(length(xVec)-2)]
x2 <- 2 * xVec[2:(length(xVec)-1)]
x3 <- xVec[3:length(xVec)]
c6 <- x1 + x2 - x3
c6
```

```
##   [1] 1382   70 1221 1749  -98  796 1949  623 -134  618  288 1472  517  -45
##  [15]  794 1982 1489  344 -206 1207  292  771 2085  810 1032 1547  767  537
##  [29]  702  676  737  664 1451  435 1355  168 1150  989  926  348 1757 1299
##  [43]  409 -497  501 2150 1157 1081 1323 2030 1887 1744  879  590  493 1330
##  [57] 1254 1281  465  767 1691  464 1238  805 -519 1425  710 -611 1517  963
##  [71] 1836 2243 -158 1860  606  506 1917 1304 2021 2025  238  226  733 1538
##  [85]  581 -659  824 1109 1136 1339 1239 1584 2300  562  567 -375 1372  761
##  [99] 1142  714 1801 2220  624 -806 1738  268  398 1941  668 2037  829  345
## [113]  337  -45  635 -285 1225  691 1792 2216  123  538 1130 1124 1172  944
## [127]  271  -62  229  785  -70 1346 1622  381  104 1036 1015  199  589 1399
## [141]  601  506  560 -145  171 1204 1427 1278 1128  615  269   37 1521 2172
## [155] 1602  464   74 1575  599   88 -267 1185 1655 1564 1420  880  229 1651
## [169]  959 1306 2008 1243  267 1110  556 -791 1300  844 1578 2427  708 1554
## [183] 1439 1150 1269 2274 1419 1067  187 2071  781 -148 1767 1851 1019 -196
## [197]  554 2223 1710  -90  788 1209  876 1322  275 1191  323 1570 1234  768
## [211] 1715  903 -768 1546 1452  -47 1125 -330  871 2463  894  133  975  201
## [225] -137 1553  299  865  746  184  267  839  -63  863 2411  133 1739 1145
## [239] 1015   47  209 1468  846   10 1146   31 1405 1058
```

d)Calculate $\sum_{i=1}^{n-1} \frac{e^{-x_{i+1}}}{x_i+10}$

Create two new x vectors of the same length, but where one is offset from the other by one element. Then calculate the above for each element in the vector, and sum all the elements.

```r
x1 <- xVec[1:length(xVec)-1]
x2 <- xVec[2:length(xVec)]
d6 <- sum(exp(-x2) / (x1 + 10))
d6
```

```
## [1] 0.01269872
```

7. This question uses the vectors xVec and yVec created in the previous question and the functions sort, order, mean, sqrt, sum and abs.

a) Pick out the values in yVec which are > 600

Create a boolean vector using yVec>600, then use this vector as an index to create a vector only with elements from yVec which exceed 600.

```
a7 <- yVec[yVec>600]
a7
```

```
##   [1] 709 871 621 930 948 783 878 671 860 768 698 974 855 813 776 721 917
##  [18] 985 705 884 840 687 957 955 786 938 930 641 615 988 881 881 997 823
##  [35] 791 643 779 693 845 815 752 766 635 993 919 686 635 613 660 800 743
##  [52] 965 743 615 615 803 948 760 604 800 772 863 902 689 881 941 924 693
##  [69] 835 632 872 876 850 961 681 791 947 915 712 665 921 798 866 828 942
##  [86] 841 645 681 827 884 890 970 632 717 846 952 609 824 695 675 777 813
## [103] 792 783 611 853 738 668 791
```

b) What are the index positions in yVec of the values which >600?

Use the resulting vector from the previous questions and the match function to get the index values in yVec which exceed 600

```
b7 <- match(a7,yVec)
b7
```

```
##   [1]   1   2   5   6   8  10  11  13  16  18  27  28  32  33  34  36  42
##  [18]  43  45  48  50  55  58  59  60  61   6  66  67  68  72  72  80  86
##  [35]  88  94  95  96  97 101 102 105 107 109 111 114 107 119 120 123 125
##  [52] 127 125  67  67 136   8 138 139 123 143 150 151 154  72 158 159  96
##  [69] 163 164 167 168 172 173 174  88 176 178 180 181 182 183 187 189 190
##  [86] 203 204 174 206  48 213 214 164 220 224 226 227 230 232 237 238  33
## [103] 241  10 245 246 247 249  88
```

c) What are the values in xVec which corrspond to the values in yVec which are > 600?

Using the index positions from the previous question, but on the xVec.

```
c7 <- xVec[b7]
c7
```

```
##   [1] 708 437 513  44 646 107 390 640 676 364 577 257 408 437 618 627 836
##  [18] 278  55 458 803 358 525 511 266 578  44  38 724  61 995 995 956  19
##  [35] 680 760  48 294  69 505 964  24  10 840 878 113  10 444 986 537 515
##  [52] 263 515 724 724 274 646 324 176 537 260 407 216 977 995 293 660 294
##  [69] 852 743 353 371 768 339 203 680  49 880 996 894 357 900 972 467 324
##  [86] 517 446 203 190 458 124  14 743 863 399 256 678 188 258 110 957 437
## [103]  34 107 179 545 123 238 680
```

d) Create the vector$(|x_1 - \bar{x}|^{1/2}, |x_2 - \bar{x}|^{1/2}, ..., |x_n - \bar{x}|^{1/2})$ where $\bar{x}$ denotes the mean of the vector $\mathbf{x} = (x_1, x_2, ..., x_n)$

Use the abs and mean function on the x vector to calculate the above.

```
d7 <- abs( xVec - mean(xVec) )^(1/2)
d7
```

```
##   [1] 16.0044994  3.8543482 15.8699716 17.7522956  7.8194629 20.1954450
##   [7] 15.7208142 13.9335566 20.2449006 18.5702989  7.8648585 13.5224258
##  [13] 13.7165593 19.3611983 13.2233127 14.9714395 19.5740645  9.3731532
##  [19] 19.4385185 16.8480266 12.8118695 16.0890025 16.0668603 19.7520632
##  [25] 11.9522383 14.0763632 11.1867779 13.9590831 11.3073427  9.1572922
##  [31]  9.6879306  6.6223863  3.8543482 12.8896858 15.1610026 13.2341981
##  [37] 18.1894475 15.7842960  8.8800901  2.4787093  9.4263461 19.5995918
##  [43] 13.1854465 18.9434949 19.9212449 15.7525871 22.4085698  2.4787093
##  [49] 16.1599505 18.7388367 23.3268943 17.6958752 13.6800585 12.3634947
##  [55]  9.6879306  5.1822775 16.2217138  8.5524266  7.6905136 13.6329014
##  [61] 11.2313846 14.2528594 15.9642100 11.5388041 17.9681941 20.3434510
##  [67] 16.4967876 19.7700784 17.7723381 22.1843188  7.4259006 23.3054500
##  [73] 14.4618118 19.4385185 22.6967839 17.4314658 14.3228489 22.4531512
##  [79] 14.1472259 22.4531512  9.5469367 20.8532012 10.6233705  4.1405314
##  [85]  9.5991666 20.8051917 21.2333700 15.1044364  9.2273506 13.8976257
##  [91] 15.4642814 15.3669776 19.3944322 17.5540309 20.0961688 12.5640758
##  [97] 19.5667064 18.8452647 11.8682770 14.7018366  7.2899931 22.6305988
## [103] 13.4217734 21.0678903 20.6846803 20.2520122 21.0203711 12.7335777
## [109] 19.7013705  9.9426355 20.6432556 19.4898948 16.0890025 18.4080417
## [115] 19.2316406 11.3954377 18.9962101 18.3614814  2.8028557 23.1115556
## [121] 13.1203658 20.8292103  9.2273506 10.1066315  7.9463199  2.8537694
## [127] 13.7424889 20.2449006 19.3870060 13.9948562  9.6361818 16.2128344
## [133] 18.8452647  2.2680388 18.7844617 13.3362663  9.5469367 11.3073427
## [139] 16.6089133  5.0143793  9.4416100 17.0837935 13.8512093 16.6690132
## [145] 20.0961688  6.0709143 15.9732276 13.1584194  8.8399095  6.6974622
## [151] 15.3576040 15.0948998  7.5402918 22.9160206 19.3944322  3.0239048
## [157] 17.4314658 12.6038089 14.4271965 20.3434510 17.7441821 15.0948998
## [163] 20.0035997 17.0629423 15.2034207  9.6511139  9.9426355  8.9919964
## [169] 20.3505282  0.3794733 18.9510950 17.7804387 10.6233705 15.7751704
## [175]  5.1131204 20.0712730 20.7811453 20.6916408  5.3050919 23.3268943
## [181] 21.0272205  9.7394045 21.1694119 12.2940636 14.6677878 18.3069386
## [187] 22.8066657  2.2680388  3.8915293 11.3073427 21.8207241 18.5163711
## [193]  9.3196566 23.1331796 10.9610219 13.1093860 18.4080417 15.8159413
## [199] 22.6084940  6.8451443 19.7194320 13.0055373  8.0711833  2.4199174
## [205]  9.0079964 16.1819653 13.6434600 13.2987217 20.3259440  4.1056059
## [211]  7.0102782 14.7358067 18.1067943 20.9250090 21.6366356 11.9939985
## [217] 19.1795725  8.4346903 21.1389688 20.2766861 20.2025741 18.2169152
## [223] 15.6797959  7.2702132 20.5634627 13.9948562 15.0380850 19.8205953
## [229]  6.7189285 16.2436449 18.0237621 13.9232180  8.7095350 16.7587589
## [235] 18.1423262 20.4485696 18.4893483 22.4754088 12.9172753  8.3579902
## [241] 20.4415264  6.9897067 13.3844686 15.9642100 16.5183534  9.6511139
## [247] 18.1343872 17.5540309 14.6238162 16.5485951
```

e)How many values in yVec are within 200 of the maximum value of the terms in yVec?

Create a boolean vector with the above condition being TRUE (or =1), then sum this vector.

```
e7 <- sum(yVec>=(max(yVec)-200))
e7
```

```
## [1] 57
```

f) How many number in xVec are divisble by 2?

Similar to g, but the condition uses the modulo to determine if it is divisible by 2.

```
evens <- sum(xVec %% 2)
evens
```

```
## [1] 126
```

g) Sort the numbers in the vector xVec in the order of increasing values in yVec

Use the sort function to sort the y vector, use the match function to determine it's location in the original yVec, and then use this index to arrange the xVec.

```
sortY <- sort(yVec)
ySortIndex <- match(sortY,yVec)
sortX <- xVec[ySortIndex]
sortX
```

```
##   [1] 405 842 308 572 461   8 256 507 373 639  42 616  29 645 376 669 688
##  [18] 688  63 638 862  77 996  93  59 585 661  72 339 339 206 537 537 322
##  [35]  42 603 425  48 707 452 477  99 224 811 715 358 358 222 395 543 480
##  [52] 193 683 710 691 954 700 614 787 835 835 435 309 309 224 460 497 944
##  [69] 530 765 523 171 870 807 469 828 624 200 713 365 781  74 129 129 701
##  [86] 760 193 866 353 168 967 967 920 541 650 148 277  18 667 667 987 120
## [103] 655 655 655 699 311 458 632  84 269  82 280 544  17  17 807 113 136
## [120] 457 702  91 625 767 828 109 860 363 121 657 668 324 382 956 299 403
## [137]  74 928 415 415 127 176 678 179 444 724 724 724 513 743 743  10  10
## [154]  38 760 446 986 894 238 640 110 203 203 113 358 977 294 294 258 577
## [171]  55 708 996 863 627 123 515 515 964 324  24 364 260 618 957  48 107
## [188] 107 266 680 680 680  34 900 537 537 274 437 437 505  19 188 190 467
## [205] 852 803 517  69 399 768 545 408 676 407 972 437 353 371 390 995 995
## [222] 995 458 458 124 216 880 836 878 357 660  44  44 578 293 324  49 646
## [239] 646 256 511 525 339 263  14 257 278  61 840 956
```

h) Pick out the elements in yVec at index positions 1,4,7,10,13,...

Create a sequence from 1 to 250 in increments of 3, and use this to index the yVec

```
x <- seq(1,250,3)
h7 <- yVec[x]
h7
```

```
##  [1] 709 517 437 783 671 860 581 347 279 974 216 776 538 460 985 248 317
## [18] 288 687 957 938 101 615 285 106 414 881 488 484 791 246 643 845 553
## [35] 465  87 993 116 473 635 310 428 965  19 489 803 604 800 175 516 902
## [52] 689 881 593 835 398 358 850 791 915 665 167 866 942 320 482 216 488
## [69] 681 273 884 970 469 717 127 952 284 695 325 777 792  72 738 791
```

8. By using the function cumprod or otherwise, calculate: $1 + \frac{2}{3} + \frac{(2*4)}{(3*5)} + \frac{(2*4*6)}{(3*5*7)} + \ldots + \frac{(2*4*\ldots*38)}{(3*5*\ldots*39)}$

Use the cuumulative product to calculate the numerators and denominators in the above sequence, then sum it and add 1.

```r
numerator <- cumprod(seq(2,38,2))
denominator <- cumprod(seq(3,39,2))
numerator
```

```
##  [1] 2.000000e+00 8.000000e+00 4.800000e+01 3.840000e+02 3.840000e+03
##  [6] 4.608000e+04 6.451200e+05 1.032192e+07 1.857946e+08 3.715891e+09
## [11] 8.174961e+10 1.961991e+12 5.101175e+13 1.428329e+15 4.284987e+16
## [16] 1.371196e+18 4.662066e+19 1.678344e+21 6.377707e+22
```

```r
denominator
```

```
##  [1] 3.000000e+00 1.500000e+01 1.050000e+02 9.450000e+02 1.039500e+04
##  [6] 1.351350e+05 2.027025e+06 3.445942e+07 6.547291e+08 1.374931e+10
## [11] 3.162341e+11 7.905854e+12 2.134580e+14 6.190283e+15 1.918988e+17
## [16] 6.332660e+18 2.216431e+20 8.200795e+21 3.198310e+23
```

```r
answer <- 1 + (numerator / denominator)
answer
```

```
##  [1] 1.666667 1.533333 1.457143 1.406349 1.369408 1.340992 1.318260
##  [8] 1.299538 1.283773 1.270260 1.258510 1.248169 1.238978 1.230737
## [15] 1.223294 1.216528 1.210341 1.204656 1.199409
```