Exercise 3 : Simple R Functions

January 30, 2018

### 1.

1. Write functions and such that if is the vector , then returns vector and returns the vector .

Here is

tmpFn1 <- function(xVec){  
 return(xVec^(1:length(xVec)))  
}  
  
## simple example  
a1 <- c(2, 5, 3, 8, 2, 4)  
  
b1 <- tmpFn1(a1)  
b1

## [1] 2 25 27 4096 32 4096

and now

tmpFn2 <- function(xVec){  
   
 n = length(xVec)  
   
 return(xVec^(1:n)/(1:n))  
}  
  
c1 <- tmpFn2(a1)  
c1

## [1] 2.0000 12.5000 9.0000 1024.0000 6.4000 682.6667

###### 

1. Now write a fuction which takes 2 arguments and where is a single number and is a strictly positive integer. The function should return the value of

tmpFn3 <- function(xVec){  
 n=length(xVec)  
 return ( sum(xVec^(1:n)/(1:n)))  
}  
  
d1 <- tmpFn3(a1)   
d1

## [1] 1736.567

### 2. Write a funciton such that if is the vector then returns the vector of moving averages:

Try out your function. (c(1:5,6:1))

# Assume vector X has 10 elements, the function can be thought as the sum of three different vectors  
# xVec <- 1:11 //1,2,3,4,5,6,7,8,9,10,11  
# xVec[-c(10,11)] = xVec[1:(11-2)] //1,2,3,4,5,6,7,8,9  
# xVec[-c(1,11)] = xVec[2:(11-1)] //2,3,4,5,6,7,8,9,10  
# xVec[-c(1,2)] = xVec[3:11] //3,4,5,6,7,8,9,10,11  
  
tmpFn4 <- function(xVec){  
 n <- length(xVec)  
 ( xVec[1:(n-2)] + xVec[2:(n-1)] + xVec[3:n] )/3  
 }  
  
tmpFn4 <- function(xVec) {  
 n <- length(xVec)  
 ( xVec[ -c(n-1,n) ] + xVec[ -c(1,n) ] + xVec[ -c(1,2) ] )/3  
 }  
  
tmpFn4(c(1:5,6:1))

## [1] 2.000000 3.000000 4.000000 5.000000 5.333333 5.000000 4.000000 3.000000  
## [9] 2.000000

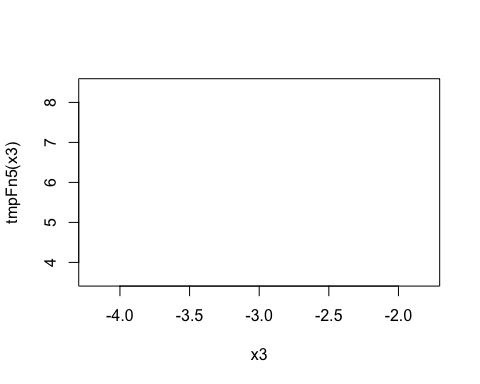
### 3. Consider the continuous function

Write a function which takes a single argument . the function should return the vector the values of the function evaluated at the values in .  
Hence plot the function for .

tmpFn5 <- function(xVec){  
 ifelse(xVec<0,xVec^2+2\*xVec+3, ifelse(xVec<2,xVec+3, xVec^2 + 4\*xVec -7))  
 }  
  
x3 <- seq(from= -3,to=3,by=100)  
x3

## [1] -3

plot(x3, tmpFn5(x3), type="l")



### 4. Write a function which takes a single argument which is a matrix. The function should return a matrix which is the same as the function argument but every odd number is doubled.

Hence the result of using the function on the matrix

should be:

tmpFn6 <- function(mat) {  
 mat[mat%%2 == 1] <- 2 \* mat[mat%%2 == 1]  
 mat   
 }  
mat4 <- matrix (c(1,1,3,5,2,6,-2,-1,-3), nrow=3, ncol=3, byrow = T )  
mat4

## [,1] [,2] [,3]  
## [1,] 1 1 3  
## [2,] 5 2 6  
## [3,] -2 -1 -3

tmpFn6(mat4)

## [,1] [,2] [,3]  
## [1,] 2 2 6  
## [2,] 10 2 6  
## [3,] -2 -2 -6

### 5. Write a function which takes 2 arguements and which are positive integers. It should return the matrix:

tmpFn7 <- function(n,k){  
   
 tempMat <- diag(k,nr = n)  
 tempMat[abs(row(tempMat)-col(tempMat))==1]<- 1  
 return (tempMat)  
}  
  
tmpFn7(5,2)

## [,1] [,2] [,3] [,4] [,5]  
## [1,] 2 1 0 0 0  
## [2,] 1 2 1 0 0  
## [3,] 0 1 2 1 0  
## [4,] 0 0 1 2 1  
## [5,] 0 0 0 1 2

### 6. Suppose an angle is given as a positive real number of degrees.

If then it is quadrant 1. If then it is quadrant 2.  
if then it is quadrant3. if then it is quadrant 4.  
if then it is quadrant 1.  
And so on …

Write a function which returns the quadrant of the angle .

#method1  
quadrant <- function(alpha){  
 tempAlpha <- alpha%%360  
 ifelse(0 <= tempAlpha & tempAlpha < 90, return("quadrant 1"),ifelse(90 <= tempAlpha & tempAlpha < 180, return("quadrant 2"), ifelse(180 <= tempAlpha & tempAlpha < 270, return("quadrant 3"), return("quadrant 4"))))  
}  
  
  
#method2  
quadrant2 <- function(alpha) {  
 1 + (alpha%%360)%/%90  
 }  
  
#method3  
 quadrant3 <- function(alpha)  
 {  
 floor(alpha/90)%%4 + 1  
 }

### 7.

1. Zeller’s congruence is the formula:

where denotes the integer part of ; for example .

Zeller’s congruence returns the day of the week given:

= the day of the month  
 = the year in the century  
 = the first 2 digits of the year (the century number)  
 = the month number (where January is month 11 of the preceding year, February is month 12 of the preceding year, March is month 1, etc.)  
For example, the date 21/07/1963 has ;  
the date 21/2/63 has .  
Write a function which returns the day of the week when given the numerical inputs of the day, month and year.  
Note that the value of 1 for denotes Sunday, 2 denotes Monday, etc.

weekday <-function (day, month, year){  
   
 if(month<=2){  
 y <- year%%100-1  
 m <- month +10  
 }else{  
 y <- year%%100  
 m <- month-2  
 }  
 k <- day  
 c <- year%/%100  
   
 weekDayIndex <- ((floor(2.6\*m - 0.2) + k + y + floor(y/4) + floor(c/4) - 2\*c ) %% 7)+1  
 weekDays <- c("Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday")  
 return(weekDays[weekDayIndex])  
}  
  
#Test: The output of executing c( weekday(27,2,1997), weekday(18,2,1940), weekday(21,1,1963) ) is the vector "Thursday", "Sunday", "Monday"  
c( weekday(27,2,1997), weekday(18,2,1940), weekday(21,1,1963) )

## [1] "Thursday" "Sunday" "Monday"

1. Does your function work if the input parameters are vectors with the same length and valid entries?

#The output of executing weekday2( c(27,18,21), c(2,2,1), c(1997,1940,1963) ) where all three input parameters are vectors is the vector "Thursday", "Sunday", "Monday".  
weekday( c(27,18,21), c(2,2,1), c(1997,1940,1963) )

## Warning in if (month <= 2) {: 條件的長度 > 1，因此只能用其第一元素

## [1] "Thursday" "Sunday" "Monday"

### 8.

1. Suppose and and for j = 1, 2, . . . . Write a function testLoop which takes the single argument n and returns the first n − 1 values of the sequence {xj}j≥0: that means the values of x0, x1, x2, . . . , xn−2.

testLoop <- function(n){  
 xVec <- c(1,2)  
 while(length(xVec)<n)  
 xVec <- c(xVec, xVec[length(xVec)] + 2/xVec[length(xVec)])  
   
 return(xVec[1:n-1])  
}  
  
testLoop(3)

## [1] 1 2

testLoop(4)

## [1] 1 2 3

testLoop(5)

## [1] 1.000000 2.000000 3.000000 3.666667

testLoop(6)

## [1] 1.000000 2.000000 3.000000 3.666667 4.212121

1. Now write a function testLoop2 which takes a single argument yVec which is a vector. The function should return where n is the length of yVec.

testLoop2 <- function(yVec){  
 n <- length(yVec)  
 return(sum(exp(1:n)))  
}  
  
yVec <- c()  
yVec

## NULL

testLoop2(yVec)

## [1] 3.718282

### 9.

Solution of the difference equation , with starting value . . Solution of the difference equation xn = rxn−1(1 − xn−1), with starting value x1.

1. Write a function quadmap( start, rho, niter ) which returns the vector where and niter denotes n, start denotes x1, and rho denotes r. Try out the function you have written: • for r = 2 and 0 < x1 < 1 you should get xn → 0.5 as n → ∞. • try tmp <- quadmap(start=0.95, rho=2.99, niter=500) Now switch back to the Commands window and type: plot(tmp, type=“l”) Also try the plot plot(tmp[300:500], type=“l”)

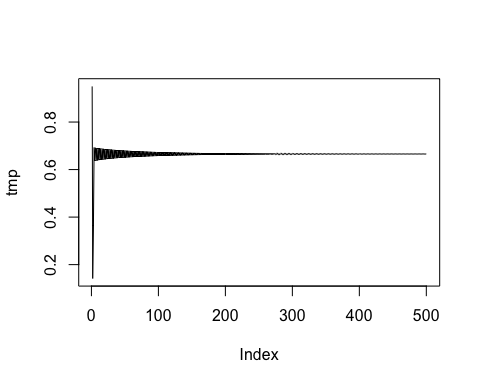
quadmap <- function(x1, r, n){  
 xVec <- c(x1)  
 while(length(xVec)<n)  
 xVec <- c(xVec, r \* xVec[length(xVec)] \*(1-xVec[length(xVec)]))  
   
 return(xVec)  
   
}  
quadmap(0.95, 2.99, 500)

## [1] 0.9500000 0.1420250 0.3643432 0.6924757 0.6367298 0.6916018 0.6377333  
## [8] 0.6907783 0.6386749 0.6900001 0.6395608 0.6892631 0.6403967 0.6885635  
## [15] 0.6411870 0.6878980 0.6419361 0.6872639 0.6426474 0.6866587 0.6433240  
## [22] 0.6860801 0.6439689 0.6855262 0.6445843 0.6849952 0.6451726 0.6844855  
## [29] 0.6457357 0.6839957 0.6462752 0.6835246 0.6467929 0.6830710 0.6472902  
## [36] 0.6826337 0.6477683 0.6822119 0.6482284 0.6818047 0.6486717 0.6814112  
## [43] 0.6490990 0.6810308 0.6495113 0.6806626 0.6499094 0.6803062 0.6502941  
## [50] 0.6799609 0.6506660 0.6796262 0.6510259 0.6793016 0.6513743 0.6789866  
## [57] 0.6517117 0.6786808 0.6520388 0.6783838 0.6523559 0.6780952 0.6526635  
## [64] 0.6778146 0.6529620 0.6775418 0.6532519 0.6772764 0.6535335 0.6770181  
## [71] 0.6538071 0.6767667 0.6540731 0.6765218 0.6543317 0.6762833 0.6545833  
## [78] 0.6760510 0.6548281 0.6758245 0.6550664 0.6756037 0.6552984 0.6753884  
## [85] 0.6555243 0.6751784 0.6557444 0.6749736 0.6559589 0.6747737 0.6561679  
## [92] 0.6745786 0.6563717 0.6743882 0.6565704 0.6742023 0.6567641 0.6740208  
## [99] 0.6569532 0.6738435 0.6571376 0.6736703 0.6573175 0.6735011 0.6574931  
## [106] 0.6733358 0.6576646 0.6731742 0.6578320 0.6730163 0.6579954 0.6728620  
## [113] 0.6581550 0.6727111 0.6583109 0.6725636 0.6584632 0.6724193 0.6586120  
## [120] 0.6722783 0.6587574 0.6721403 0.6588995 0.6720054 0.6590383 0.6718734  
## [127] 0.6591740 0.6717443 0.6593067 0.6716179 0.6594364 0.6714943 0.6595632  
## [134] 0.6713733 0.6596872 0.6712550 0.6598085 0.6711391 0.6599271 0.6710258  
## [141] 0.6600431 0.6709148 0.6601565 0.6708062 0.6602675 0.6706998 0.6603761  
## [148] 0.6705957 0.6604823 0.6704938 0.6605862 0.6703941 0.6606879 0.6702964  
## [155] 0.6607875 0.6702007 0.6608848 0.6701070 0.6609801 0.6700153 0.6610734  
## [162] 0.6699255 0.6611647 0.6698375 0.6612541 0.6697514 0.6613415 0.6696670  
## [169] 0.6614272 0.6695844 0.6615110 0.6695035 0.6615930 0.6694242 0.6616734  
## [176] 0.6693465 0.6617520 0.6692705 0.6618290 0.6691960 0.6619044 0.6691230  
## [183] 0.6619783 0.6690515 0.6620506 0.6689814 0.6621214 0.6689128 0.6621907  
## [190] 0.6688456 0.6622586 0.6687797 0.6623251 0.6687152 0.6623902 0.6686520  
## [197] 0.6624539 0.6685901 0.6625164 0.6685294 0.6625776 0.6684699 0.6626375  
## [204] 0.6684117 0.6626962 0.6683546 0.6627536 0.6682986 0.6628099 0.6682438  
## [211] 0.6628651 0.6681901 0.6629191 0.6681375 0.6629720 0.6680859 0.6630239  
## [218] 0.6680354 0.6630747 0.6679859 0.6631244 0.6679374 0.6631731 0.6678898  
## [225] 0.6632209 0.6678432 0.6632676 0.6677976 0.6633135 0.6677528 0.6633584  
## [232] 0.6677090 0.6634023 0.6676660 0.6634454 0.6676239 0.6634876 0.6675827  
## [239] 0.6635290 0.6675422 0.6635695 0.6675026 0.6636092 0.6674637 0.6636481  
## [246] 0.6674257 0.6636862 0.6673884 0.6637236 0.6673518 0.6637602 0.6673160  
## [253] 0.6637960 0.6672809 0.6638311 0.6672465 0.6638656 0.6672128 0.6638993  
## [260] 0.6671797 0.6639323 0.6671473 0.6639647 0.6671156 0.6639964 0.6670844  
## [267] 0.6640275 0.6670540 0.6640580 0.6670241 0.6640879 0.6669948 0.6641171  
## [274] 0.6669661 0.6641458 0.6669379 0.6641739 0.6669104 0.6642014 0.6668833  
## [281] 0.6642284 0.6668569 0.6642548 0.6668309 0.6642807 0.6668055 0.6643061  
## [288] 0.6667805 0.6643309 0.6667561 0.6643553 0.6667321 0.6643792 0.6667087  
## [295] 0.6644026 0.6666856 0.6644255 0.6666631 0.6644480 0.6666410 0.6644700  
## [302] 0.6666193 0.6644916 0.6665981 0.6645128 0.6665773 0.6645335 0.6665569  
## [309] 0.6645538 0.6665369 0.6645737 0.6665173 0.6645932 0.6664981 0.6646124  
## [316] 0.6664793 0.6646311 0.6664608 0.6646494 0.6664428 0.6646674 0.6664251  
## [323] 0.6646851 0.6664077 0.6647024 0.6663907 0.6647193 0.6663740 0.6647359  
## [330] 0.6663576 0.6647522 0.6663416 0.6647681 0.6663259 0.6647837 0.6663105  
## [337] 0.6647990 0.6662954 0.6648140 0.6662806 0.6648287 0.6662661 0.6648432  
## [344] 0.6662519 0.6648573 0.6662380 0.6648711 0.6662244 0.6648847 0.6662110  
## [351] 0.6648980 0.6661979 0.6649110 0.6661850 0.6649238 0.6661724 0.6649363  
## [358] 0.6661601 0.6649486 0.6661480 0.6649606 0.6661361 0.6649724 0.6661245  
## [365] 0.6649839 0.6661131 0.6649952 0.6661020 0.6650063 0.6660910 0.6650172  
## [372] 0.6660803 0.6650278 0.6660698 0.6650383 0.6660595 0.6650485 0.6660494  
## [379] 0.6650585 0.6660395 0.6650684 0.6660298 0.6650780 0.6660203 0.6650874  
## [386] 0.6660110 0.6650967 0.6660018 0.6651057 0.6659929 0.6651146 0.6659841  
## [393] 0.6651233 0.6659755 0.6651319 0.6659671 0.6651402 0.6659588 0.6651484  
## [400] 0.6659507 0.6651565 0.6659428 0.6651643 0.6659350 0.6651720 0.6659274  
## [407] 0.6651796 0.6659199 0.6651870 0.6659126 0.6651943 0.6659054 0.6652014  
## [414] 0.6658984 0.6652084 0.6658915 0.6652152 0.6658847 0.6652219 0.6658781  
## [421] 0.6652285 0.6658716 0.6652349 0.6658653 0.6652412 0.6658590 0.6652474  
## [428] 0.6658529 0.6652535 0.6658469 0.6652594 0.6658411 0.6652653 0.6658353  
## [435] 0.6652710 0.6658297 0.6652766 0.6658241 0.6652820 0.6658187 0.6652874  
## [442] 0.6658134 0.6652927 0.6658082 0.6652978 0.6658031 0.6653029 0.6657981  
## [449] 0.6653078 0.6657932 0.6653127 0.6657884 0.6653175 0.6657837 0.6653221  
## [456] 0.6657791 0.6653267 0.6657746 0.6653312 0.6657701 0.6653356 0.6657658  
## [463] 0.6653399 0.6657615 0.6653441 0.6657574 0.6653482 0.6657533 0.6653523  
## [470] 0.6657493 0.6653563 0.6657453 0.6653602 0.6657415 0.6653640 0.6657377  
## [477] 0.6653677 0.6657340 0.6653714 0.6657304 0.6653750 0.6657269 0.6653785  
## [484] 0.6657234 0.6653819 0.6657200 0.6653853 0.6657166 0.6653886 0.6657133  
## [491] 0.6653919 0.6657101 0.6653951 0.6657070 0.6653982 0.6657039 0.6654012  
## [498] 0.6657009 0.6654042 0.6656979

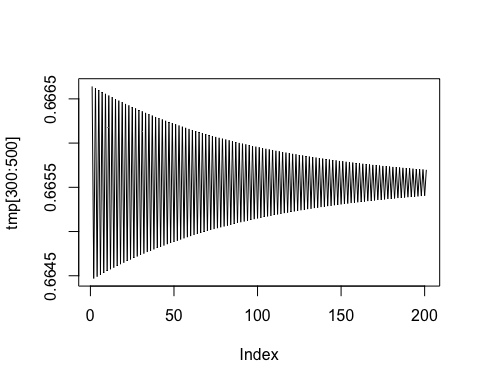
quadmap(0.9, 2, 1000)

## [1] 0.9000000 0.1800000 0.2952000 0.4161139 0.4859263 0.4996039  
## [7] 0.4999997 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [13] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [19] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [25] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [31] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [37] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [43] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [49] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [55] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [61] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [67] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [73] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [79] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [85] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [91] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [97] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [103] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [109] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [115] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [121] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [127] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [133] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [139] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [145] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [151] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [157] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [163] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [169] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [175] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [181] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [187] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [193] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [199] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [205] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [211] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [217] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [223] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [229] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [235] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [241] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [247] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [253] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [259] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [265] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [271] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [277] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [283] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [289] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [295] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [301] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [307] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [313] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [319] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [325] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [331] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [337] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [343] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [349] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [355] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [361] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [367] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [373] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [379] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [385] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [391] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [397] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [403] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [409] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [415] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [421] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [427] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [433] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [439] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [445] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [451] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [457] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [463] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [469] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [475] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [481] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [487] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [493] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [499] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [505] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [511] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [517] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [523] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [529] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [535] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [541] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [547] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [553] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [559] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [565] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [571] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [577] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [583] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [589] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [595] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [601] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [607] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [613] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [619] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [625] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [631] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [637] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [643] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [649] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [655] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [661] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [667] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [673] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [679] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [685] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [691] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [697] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [703] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [709] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [715] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [721] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [727] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [733] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [739] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [745] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [751] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [757] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [763] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [769] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [775] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [781] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [787] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [793] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [799] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [805] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [811] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [817] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [823] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [829] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [835] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [841] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [847] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [853] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [859] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [865] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [871] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [877] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [883] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [889] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [895] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [901] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [907] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [913] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [919] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [925] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [931] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [937] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [943] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [949] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [955] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [961] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [967] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [973] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [979] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [985] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [991] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000  
## [997] 0.5000000 0.5000000 0.5000000 0.5000000

tmp <- quadmap(0.95, 2.99, 500)  
plot(tmp, type="l")



plot(tmp[300:500], type="l")



1. Now write a function which determines the number of iterations needed to get So this function has only 2 arguments: start and rho. (For start=0.95 and rho=2.99, the answer is 84.)

quadmap2 <- function(start, rho, eps = 0.02){  
 x1 <- start  
 x2 <- rho\*x1\*(1 - x1)  
 niter <- 1  
 while(abs(x1 - x2) >= eps) {  
 x1 <- x2  
 x2 <- rho\*x1\*(1 - x1)  
 niter <- niter + 1  
 }  
 niter   
 }  
  
quadmap2(0.95,2.99)

## [1] 84

### 10.

1. Given a vector (x1, . . . , xn), the sample autocorrelation of lag k is defined to be Thus Write a function tmpFn(xVec) which takes a single argument xVec which is a vector and returns a list of two values: r1 and r2. In particular, find r1 and r2 for the vector (2, 5, 8, . . . , 53, 56).

tmpFn8 <- function(xVec)  
 {  
 numerator <- xVec - mean(xVec)  
 denominator <- sum(numerator^2)  
 n <- length(xVec)  
 r1 <- sum( numerator[2:n] \* numerator[1:(n-1)] )/denominator  
 r2 <- sum( numerator[3:n] \* numerator[1:(n-2)] )/denominator  
 list(r1 = r1, r2 = r2)  
}  
  
vec8 <- seq(2,56, by=3)  
vec8

## [1] 2 5 8 11 14 17 20 23 26 29 32 35 38 41 44 47 50 53 56

tmpFn8 (vec8)

## $r1  
## [1] 0.8421053  
##   
## $r2  
## [1] 0.6859649

1. (Harder.) Generalise the function so that it takes two arguments: the vector xVec and an integer k which lies between 1 and n − 1 where n is the length of xVec. The function should return a vector of the values (r0 = 1, r1, . . . , rk). If you used a loop to answer part (b), then you need to be aware that much, much better solutions are possible—see exercises 4. (Hint: sapply.)

tmpFn9 <- function(x, k)  
 {  
 numerator <- x - mean(x)  
 denominator <- sum(xc^2)  
 n <- length(x)  
 tmpFn <- function(j){ sum( xc[(j+1):n] \* xc[1:(n-j)] )/denominator }  
 c(1, sapply(1:k, tmpFn))  
}