Problem\_Set\_1

Kristopher C. Toll

January 26, 2018

# R Programing exercises

### 1. Calculate the square root of 729

b = sqrt(729)  
print(b)

## [1] 27

### 2. Create a new variable a with value 1947.0

a = as.integer(1947.0)  
print(a)

## [1] 1947

### 3. Create a vector b contaning number form 1 to 6 and find out it's class.

b is a numeric variable.

b = seq(1, 6, by = 1)  
b.1 <- class(b)  
print(c(b, b.1))

## [1] "1" "2" "3" "4" "5" "6" "numeric"

### 4. Create a vector c containg following mixed elements

c = c(1, "a", 2, "b")  
print(c)

## [1] "1" "a" "2" "b"

1. Find out its class. It is a character variable

class(c)

## [1] "character"

1. Get the length of the vector. The length is four

length(c) # Figuring out the length of c

## [1] 4

1. Get the 2nd and 3rd elements, which is "a" and "2".

print(c[2]) # Printing the 2nd element

## [1] "a"

print(c[3]) # Printing the 3rd element

## [1] "2"

### 5. Create a vector d containing following elements c(1, 2, NA, 4, 5, 6, NA, NA, NA, 10)

Remove missing values from d

d = c(1, 2, NA, 4, 5, 6, NA, NA, NA, 10)  
d = as.numeric(na.omit(d)) # Removing NA valuse and converting the vector to a numeric one  
print(d)

## [1] 1 2 4 5 6 10

### 6. Create a vector of values of cos 3 at x = 3, 3.1, 3.2, ... 6

x = seq(3, 6, by = 0.1)  
x.1 = exp(x) \* cos(x) # cosine is reading in x as a radian unit  
print(x.1)

## [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

#x.2 = exp(x) \* cos(x/180) # this funtions will read it in as degrees  
#print(x.2)

### 7. Calculate

s = seq(10, 100, by = 1) # Creating the sequence  
s.1 <- s^3 + 4 \* s^2 # transforming the sequence  
s.2 = sum(s.1) # Calculating the sum of the transformed sequence  
print(s.2)

## [1] 26852735

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

x <- sample(0:999, size = 250, replace = TRUE)  
y <- sample(0:999, size = 250, replace = TRUE)

1. Pick out the values in Y which are > 600

y.1 <- subset(y, y>600) # use the subset comand to pull out elements of a vector  
print(y.1)

## [1] 939 646 844 850 783 933 670 755 797 857 791 911 632 975 906 757 605  
## [18] 705 625 748 906 805 978 780 832 829 998 823 603 759 716 819 764 740  
## [35] 820 821 839 705 742 825 957 923 771 671 852 724 751 871 801 661 835  
## [52] 793 964 783 699 805 660 768 658 946 938 947 744 898 927 788 817 744  
## [69] 997 866 689 919 685 686 857 979 890 822 724 931 718 930 766 755 940  
## [86] 907 936 654 655 618 662 807 854 987

1. How many values in y are within 200 of the maximum value of the terms in y?

y.2 <- subset(y, y >= max(y)-200)  
print(length(y.2))

## [1] 49

1. Create the vector e

e <- abs(x-mean(x))^(1/2)  
print(e)

## [1] 17.807976 8.069944 18.442451 20.442016 15.070634 14.598767 20.177314  
## [8] 8.492585 22.200811 18.360937 15.520180 1.695877 14.692719 17.744746  
## [15] 10.398269 9.226267 9.226267 20.955286 12.850058 16.518959 8.238689  
## [22] 18.624607 2.622213 20.096666 20.319350 20.121531 10.764572 20.448081  
## [29] 12.323798 15.783662 16.465843 12.654011 12.762288 21.139442 13.670260  
## [36] 20.417541 15.807467 7.132741 7.491595 9.893230 18.189997 15.520180  
## [43] 22.402589 21.705391 11.624285 11.537937 18.944023 12.732792 21.020847  
## [50] 7.008852 17.919933 21.491301 12.693463 12.414669 13.071955 10.529767  
## [57] 12.364303 14.903825 21.980082 15.293005 16.907868 4.015470 6.846605  
## [64] 19.927970 4.886307 20.325452 12.879286 16.766753 18.678223 18.597742  
## [71] 19.496769 15.592434 18.224270 18.031195 19.055813 6.717440 20.949368  
## [78] 7.866130 14.487098 6.092947 14.243455 21.957322 15.775804 5.086846  
## [85] 10.240898 20.594271 17.230090 20.251518 19.023039 10.914394 15.846892  
## [92] 9.480295 17.836031 19.541648 20.220682 13.752236 9.893230 5.488534  
## [99] 4.257229 8.767896 10.577145 14.004428 19.206145 15.037420 19.802121  
## [106] 15.656436 17.975650 19.284087 5.841575 3.335266 14.031251 14.252158  
## [113] 22.093528 19.522397 2.263625 8.551257 6.846605 21.092084 3.622706  
## [120] 15.037420 10.289606 13.224069 11.624285 16.151656 17.432039 8.418789  
## [127] 10.253000 19.522397 9.048536 7.491595 16.465843 10.204117 17.940903  
## [134] 13.851931 11.308227 21.427179 12.444919 18.516911 16.404999 20.733451  
## [141] 22.223321 5.396666 12.564872 10.301650 16.996353 20.374592 18.107347  
## [148] 2.263625 4.886307 6.072561 17.638707 12.879286 10.301650 20.877644  
## [155] 19.669164 18.604408 20.399118 15.712288 17.857099 16.759356 20.618535  
## [162] 16.244261 12.129468 9.440551 18.791594 9.740431 13.337016 8.595115  
## [169] 11.005635 17.410457 7.881878 16.275012 13.788546 13.743217 14.794458  
## [176] 18.711601 12.693463 19.821100 17.150044 22.469446 3.142610 21.286709  
## [183] 13.670260 2.622213 19.368118 15.839066 11.440105 10.386337 15.775804  
## [190] 13.337016 11.185884 10.588862 15.909871 17.638707 22.200811 16.282629  
## [197] 19.490408 21.768693 21.468023 5.733760 17.779876 13.670260 21.002952  
## [204] 10.093364 21.092084 21.263208 8.477971 20.399118 10.857071 5.578889  
## [211] 19.770584 22.138744 19.003263 18.244890 13.642727 8.192924 14.555961  
## [218] 14.426503 21.537781 5.578889 19.921747 12.534911 19.258141 19.871487  
## [225] 8.881216 9.006886 11.581192 14.348380 18.496594 15.495935 16.282629  
## [232] 21.636173 10.301650 19.776855 13.715830 19.342285 6.566887 8.653092  
## [239] 13.308794 21.121648 21.216126 16.282629 16.855978 11.319187 9.006886  
## [246] 21.928885 8.433505 9.387438 12.364303 16.937414

1. Create another vector

n=2:length(y)  
d <- (y[n]-x[n-1]) # It worked!  
print(d)

## [1] -81 77 786 -790 -198 366 307 423 -667 405 -364 -43 -150 42  
## [15] 398 -169 525 486 342 -11 -381 -471 297 -40 -115 -683 -448 109  
## [29] -230 204 689 299 -437 35 226 -704 14 61 -89 114 -199 -723  
## [43] -247 -12 146 -121 -717 575 -424 361 387 23 -87 441 -315 -404  
## [57] -383 31 548 105 -367 -323 289 902 -320 255 164 391 -503 -80  
## [71] 216 -131 322 291 421 -292 -459 -40 -210 39 -403 705 77 245  
## [85] -206 302 -50 737 -34 465 -97 -199 530 -358 -160 521 366 460  
## [99] 296 101 -417 555 -445 231 623 503 35 -485 -233 -358 181 63  
## [113] 39 343 313 21 -114 -835 104 284 62 167 435 -713 167 219  
## [127] -136 100 124 368 438 379 -157 261 -434 123 -205 -664 714 -785  
## [141] -40 133 93 -5 -562 331 -360 -84 -363 -498 228 239 -248 -454  
## [155] -736 780 440 -209 -24 -347 70 60 51 49 308 156 -444 -91  
## [169] 625 -118 -66 -241 563 7 -180 776 73 -201 -199 -312 34 389  
## [183] 177 -462 861 146 198 -268 -283 53 563 -89 -148 536 -959 157  
## [197] 57 -702 -188 -394 -108 -133 207 160 -572 486 -272 252 -500 -387  
## [211] -508 435 808 -301 600 -179 -208 651 -523 192 -826 319 -588 -745  
## [225] 46 -366 48 -669 511 -234 -137 108 1 308 215 60 357 -211  
## [239] -129 182 39 626 778 225 -232 -426 164 118 -607

### 9. In this exercise, we will consider a quadratic equations of the form . Create a vector of coefficients for a quadratic equations.

coeffs <- sample(-20:20, size = 3, replace = TRUE)

1. Determine the length of the object coeffs.

print(length(coeffs))

## [1] 3

1. Create 200 values of x from a regularly spaced vector between -3 and 3

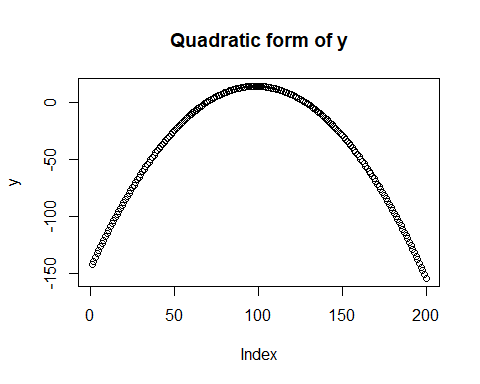
x <- seq(from = -3, to = 3, length.out = 200)

1. Now obtain the value of the quadratic function (y) at each value of x

y = coeffs[1] + coeffs[2] \* x + coeffs[3] \* x^2

1. Construct the plot

plot(y, main = "Quadratic form of y")



### 10. Without using R, determine the result of the following computation

### 11. Create the following matrix with 15 rows

A = matrix(c(rep(c(10, -5, 10), times = 15)), nrow = 15, byrow = TRUE)  
print(A)

## [,1] [,2] [,3]  
## [1,] 10 -5 10  
## [2,] 10 -5 10  
## [3,] 10 -5 10  
## [4,] 10 -5 10  
## [5,] 10 -5 10  
## [6,] 10 -5 10  
## [7,] 10 -5 10  
## [8,] 10 -5 10  
## [9,] 10 -5 10  
## [10,] 10 -5 10  
## [11,] 10 -5 10  
## [12,] 10 -5 10  
## [13,] 10 -5 10  
## [14,] 10 -5 10  
## [15,] 10 -5 10

A.1 = A # Copy the matrix  
A.1[,3] = A.1[,1] + A.1[,2] # rewrite the 3rd column as a sum of the first two  
print(A.1)

## [,1] [,2] [,3]  
## [1,] 10 -5 5  
## [2,] 10 -5 5  
## [3,] 10 -5 5  
## [4,] 10 -5 5  
## [5,] 10 -5 5  
## [6,] 10 -5 5  
## [7,] 10 -5 5  
## [8,] 10 -5 5  
## [9,] 10 -5 5  
## [10,] 10 -5 5  
## [11,] 10 -5 5  
## [12,] 10 -5 5  
## [13,] 10 -5 5  
## [14,] 10 -5 5  
## [15,] 10 -5 5

### 12. Create a function that given two number will return he sum of those two number

add <- function(a,b){  
 c = a + b  
 return(c)  
}  
  
add(5,10)

## [1] 15

### 13. Create a function that given a vector and an integer will return how many times the integer appears inside the vector

count <- function(x, int){  
 y <- vector()  
 for(i in 1:length(x)){  
 ifelse(x[i] == int, y[i] <- 1, y[i] <- 0)  
 }  
 z = sum(y)  
 return(z)  
}  
  
# testing the function  
  
x <- c(4,5, 6, 6, 7, 8)   
count(x = x, int = 6) # The argument should return a 2 given vector x

## [1] 2

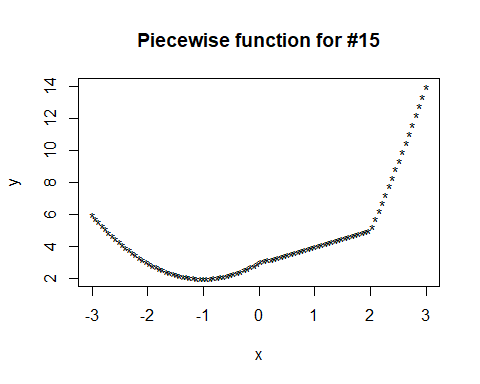
### 14. Create a function that given an integer vector will return

zsquared <- function(x){  
 z <- numeric(length(x))  
 for(i in 1:length(x)){  
 z[i] <- x[i]^2  
 }  
 return(z)  
}  
  
# Testing the function  
  
x <- c(2, 2)  
zsquared(x) # should return two 4s given vector x

## [1] 4 4

### 15. Create a piecewise function

piecewise <- function(x){  
 y <- numeric(length(x))  
 for(i in 1:length(x)){  
 if(x[i] < 0 ){  
 y[i] = (x[i]^2 + 2 \* x[i] + 3)  
 }  
 else if(x[i] >= 0 & x[i] < 2 ){  
 y[i] = (x[i] + 3)  
 }  
 else if( 2 <= x[i]){  
 y[i] = (x[i]^2 + 4 \* x[i] - 7)  
 }  
 }  
 return(y)  
}  
  
# Testing the piecewise function  
  
x <-seq(-3, 3, length = 100)  
y <- piecewise(x)  
  
plot(x, y, main = "Piecewise function for #15", pch = "\*", col = 617, bg = 456)



# Theory

### Problem 1:

Show that for , with and

### Problem 2:

Show that the autocovariance function can be written as where and

### Problem 3:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| time, t | Yt | Yt-1 | Yt-3 | µ̂Y(t) | Yt - µ̂Y(t) | (Yt - µ̂Y(t))2 | Yt-1 - µ̂Y(t) | ( Yt-1 - µ̂Y(t))2 | ( Yt - µ̂Y(t))( Yt-1 - µ̂Y(t)) |
| Jan-49 | 112.00 |  |  | 126.67 | -14.67 | 215.11 |  |  |  |
| Feb-49 | 118.00 | 112.00 |  | 126.67 | -8.67 | 75.11 | -14.67 | 215.11 | 127.11 |
| Mar-49 | 132.00 | 118.00 | 112.00 | 126.67 | 5.33 | 28.44 | -8.67 | 75.11 | -46.22 |
| Apr-49 | 129.00 | 132.00 | 118.00 | 126.67 | 2.33 | 5.44 | 5.33 | 28.44 | 12.44 |
| May-49 | 121.00 | 129.00 | 132.00 | 126.67 | -5.67 | 32.11 | 2.33 | 5.44 | -13.22 |
| Jun-49 | 135.00 | 121.00 | 129.00 | 126.67 | 8.33 | 69.44 | -5.67 | 32.11 | -47.22 |
| Jul-49 | 148.00 | 135.00 | 121.00 | 126.67 | 21.33 | 455.11 | 8.33 | 69.44 | 177.78 |
| Aug-49 | 148.00 | 148.00 | 135.00 | 126.67 | 21.33 | 455.11 | 21.33 | 455.11 | 455.11 |
| Sep-49 | 136.00 | 148.00 | 148.00 | 126.67 | 9.33 | 87.11 | 21.33 | 455.11 | 199.11 |
| Oct-49 | 119.00 | 136.00 | 148.00 | 126.67 | -7.67 | 58.78 | 9.33 | 87.11 | -71.56 |
| Nov-49 | 104.00 | 119.00 | 136.00 | 126.67 | -22.67 | 513.78 | -7.67 | 58.78 | 173.78 |
| Dec-49 | 118.00 | 104.00 | 119.00 | 126.67 | -8.67 | 75.11 | -22.67 | 513.78 | 196.44 |
| sum | 1520.00 | 1402.00 | 1298.00 | 1520.00 | 0.00 | 2070.67 | 8.67 | 1995.56 | 1163.56 |

Sample Variance: = 2070.67/(12-1) = 188.24

Sample autocovariance: (1/11)\*1163.56 = 105.78

Sample Autocorrelation: 1163.56/ sqrt(2070.67\*1995.56) = 0.57