HW5

2024-09-29

Status

- Q1:
- Q2:
- Q3:
- Q4: WIP
- Q5: WIP
- Q6: ISH
- Q7: WIP

Homework 5

Due October 15

Q1

Q:

Use the state.x77 data matrix and the tapply(), and separately, the aggregate() function to obtain

a.

The mean per capita income of the states in each of the four regions defined by the factor state.region,

b.

Themaximumilliteracy rates for states in each of the nine divisions defined by the factor state.division,

c.

The number of states in each region,

d.

The names of the states in each division,

e.

The median high school graduation rates for groups of states defined by combinations of the factors state.region and state.size.

A:

a.

b.

c.

d.

e.

Q2

Q:

For a sample $x_1, x_2, ..., x_n$, the MAD estimator of scale is defined as $1.4826 \cdot median\{|x_i - \bar{x}|\}$ were $\bar{x} =$ median$\{x_i\}$

Use the matrix object mtcars to compute the MAD estimator of scale for the columns in mtcars

a.

Using the apply() function with the mad() function

b.

By calculating it directly from the definition (i.e., not using the mad() function). You may use the apply() and the sweep() functions but avoid using any loops

A:

a.

b.

Q3

Q:

Let $h(x,n) = 1 + x + x^2 + ... + x^n = \sum_{i=0}^{\infty} x^i$ Answer the following questiuons

a.

Write code to do the above in a for loop.

b.

Rewrite the code that you wrote to use a while loop.

c.

For each of the $x = 0.3, 1.01$ evaluate the performance for $n = 500, 5000$ in terms of time taken by the software. To do so, wrap the code around the R function system.time() with the same code as above inside the parentheses. Report the values returned in the output as per the user time field.

d.

Compare the above with results obtained avoiding loops.

A:

a.

b.

c.

d.

Q4

Q:

The Lotka-Volterra model for a predator-prey system assumes that x(t) is the number of prey animals at the start of year t and that y(t) is the number of predators at the start of year t. Then the number of prey animals and predators at the end of the following year is given by:

$$x(t+1) = x(t) + b_x x(t) - d_x x(t) y(t)$$

$$y(t+1) = y(t) + b_y d_x x(t) y(t) - d_y y(t)$$

Where b_x, b_y, d_x and d_y are as follows: - b_x is the natural birth rate of the prey animals in the absence of predation - d_x is the death rate of prey animal in an encounter with the predator. - d_y is the natural death rate of the predators in the absence of food (prey animals) - b_y is the efficiency of turning predated animals into predators

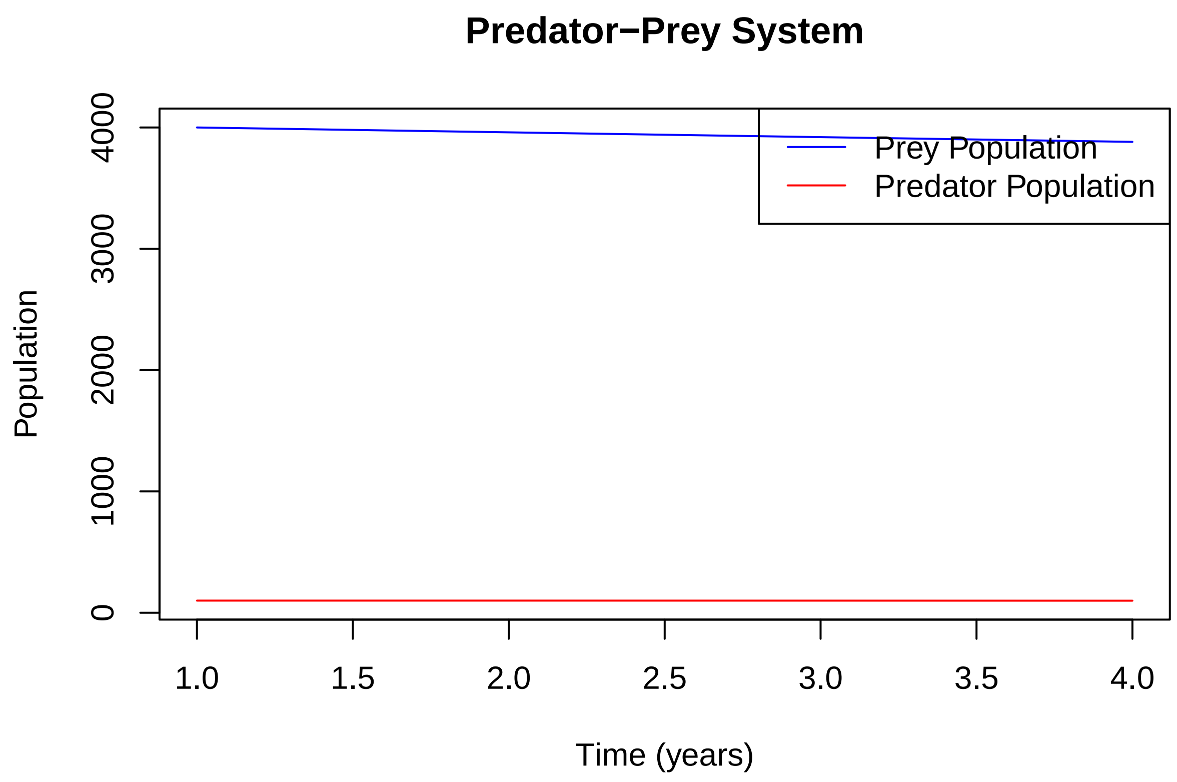
Let $b_x = 0.04$, $d_x = 0.0005$, $b_y = 0.1$, and $d_y = 0.2$.

Suppose that there were 4000 animal of the prey variety at the beginning of the time period. Also, suppose that there were only 100 predators. Write a while loop to show the predator-prey system as long as there are over 3900 prey animals. Save the prey/predator output in a list and plot them using lines on the same plot.

A:

```r
# Set initial parameters
bx <- 0.04  # Natural birth rate of prey
dx <- 0.0005  # Death rate of prey in an encounter with predator
by <- 0.1  # Efficiency of turning predated animals into predators
dy <- 0.2  # Natural death rate of predators

# Initial conditions
prey <- 4000  # Initial number of prey
predator <- 100  # Initial number of predators

# Create lists to store prey and predator values over time
prey_list <- c(prey)
predator_list <- c(predator)

# While loop to simulate the predator-prey system
while (prey > 3900) {
  # Update prey and predator populations using Lotka-Volterra model
  new_prey <- prey + bx * prey - dx * prey * predator
  new_predator <- predator + by * dx * prey * predator - dy * predator

  # Append the new populations to the lists
  prey_list <- c(prey_list, new_prey)
  predator_list <- c(predator_list, new_predator)

  # Update the current prey and predator populations for the next iteration
  prey <- new_prey
  predator <- new_predator
```

```r
}

# Plot the results
plot(prey_list, type="l", col="blue", ylim=c(min(c(prey_list, predator_list)), max(c(prey_list, predato
     xlab="Time (years)", ylab="Population", main="Predator-Prey System")
lines(predator_list, col="red")
legend("topright", legend=c("Prey Population", "Predator Population"), col=c("blue", "red"), lty=1)
```



Q5

Q:

The game of craps is played as follows: first, Player 1 rolls two six-sided die; let x be the sum of the die on the first roll. If x = 7 or x = 11, then Player 1 wins, otherwise the player continues rolling until (s)he gets x again, in which case also Player 1 wins, or until (s)he gets 7 or 11, in which case (s)he loses. Write R code to simulate the game of craps. You can simulate the roll of a fair die using the sample() function in R.

A:

```r
# Function to simulate a game of craps
play_craps <- function() {
  # Roll two dice and sum the result
  first_roll <- sum(sample(1:6, 2, replace = TRUE))

  # Check if the player wins on the first roll
  if (first_roll == 7 || first_roll == 11) {
    return("Player 1 wins on the first roll!")
  } else {
    # Continue rolling until player gets the first roll value or 7 or 11
    point <- first_roll
    repeat {
      roll <- sum(sample(1:6, 2, replace = TRUE))

      # If player rolls the point value again, they win
      if (roll == point) {
        return("Player 1 wins!")
      }

      # If player rolls a 7 or 11, they lose
      if (roll == 7 || roll == 11) {
        return("Player 1 loses!")
      }
    }
  }
}

# Simulate the game
result <- play_craps()
print(result)
```

```
## [1] "Player 1 loses!"
```

Q6

Q:

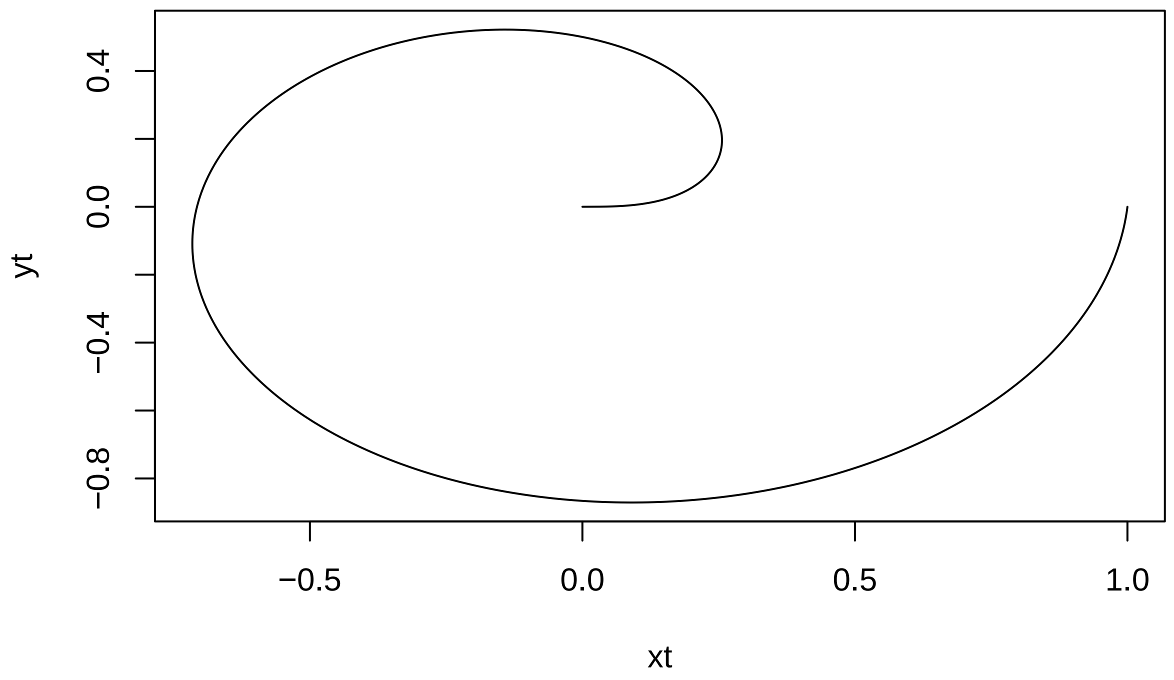
Suppose that $(x(t), y(t))$ has polar coordinates given by $(\sqrt{t}, 2\pi t)$. Write code to plot the curve $(x(t), y(t))$ for $t \in [0, 1]$.

A:

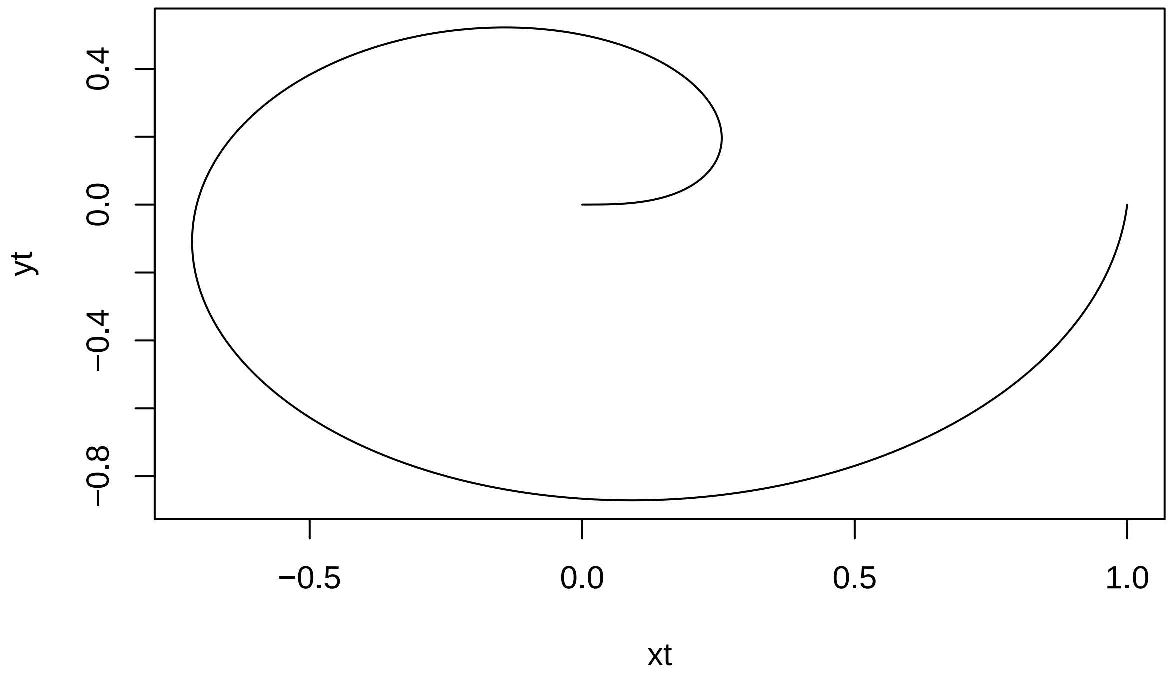
Polar to Cartesian coordinates, r radius x = r cos θ y = r sin θ

$x(t) = \sqrt{t} \cdot \cos(2\pi t)$ $y(t) = \sqrt{t} \cdot \sin(2\pi t)$

```r
t <- seq(from = 0, to = 1, by = 0.001)

# xt <- sqrt(t)
# yt <- 2 * pi * t

xt <- sqrt(t) * cos(2 * pi * t)
yt <- sqrt(t) * sin(2 * pi * t)

plot(x = xt,
     y = yt,
     type = "l")
```



```r
# Define the time sequence
t <- seq(from = 0, to = 1, by = 0.001)

# Initialize vectors for xt and yt
xt <- numeric(length(t))
yt <- numeric(length(t))

# For loop to calculate xt and yt at each time point
for (i in 1:length(t)) {
  xt[i] <- sqrt(t[i]) * cos(2 * pi * t[i])
  yt[i] <- sqrt(t[i]) * sin(2 * pi * t[i])
}

# Plot the results
plot(x = xt,
     y = yt,
     type = "l")
```



Q7

Q:

Consider the following code:

```r
x <- matrix(rnorm(n = 500), ncol = 5)
varx <- var(x)
```

Starting with varx, use two applications of the sweep() function, one dividing each row of the matrix and the other dividing each column, of a covariance matrix to obtain R, the correlation matrix.

A:

```r
# Generate the matrix x
x <- matrix(rnorm(n = 500), ncol = 5)

# Calculate the covariance matrix varx
varx <- var(x)

# Step 1: Extract the standard deviations (square root of the diagonal of varx)
std_devs <- sqrt(diag(varx))

# Step 2: Use sweep to divide each row by the corresponding standard deviation
varx_sweep_row <- sweep(varx, 1, std_devs, FUN = "/")

# Step 3: Use sweep to divide each column by the corresponding standard deviation
R <- sweep(varx_sweep_row, 2, std_devs, FUN = "/")

# Print the correlation matrix R
print(R)
```

```
##             [,1]        [,2]         [,3]         [,4]        [,5]
## [1,]  1.00000000 -0.01187456 -0.121464836 -0.075744965 -0.01335540
## [2,] -0.01187456  1.00000000  0.087560023 -0.062664837 -0.07773441
## [3,] -0.12146484  0.08756002  1.000000000 -0.009745562 -0.06501185
## [4,] -0.07574496 -0.06266484 -0.009745562  1.000000000  0.13925822
## [5,] -0.01335540 -0.07773441 -0.065011846  0.139258217  1.00000000
```