

Example Homework

Ruoqing Zhu (rqzhu)

Due: Yesterday

Contents

Question 1 (40 Points) Bivariate Normal Distribution	1
Question 2 (40 Points) Write R Functions	1
Question 3 (20 Points) Using AI Tools	2

You should remove instructions from this file.

Question 1 (40 Points) Bivariate Normal Distribution

Generate 100 random variables from the bivariate Normal Distribution $\mathcal{N}(\mu, \Sigma)$, where $\mu = (1, 2)^T$ and

$$\Sigma = \begin{pmatrix} 1 & 0.5 \\ 0.5 & 2 \end{pmatrix}$$

Use random seed 1. Report the sample mean vector.

Answer:

This can be done using the `mvrnorm` function in the `MASS` package:

```
library(MASS)
set.seed(1)
Sigma <- matrix(c(1,0.5,0.5,2),2,2)
mu <- c(1, 2)
X <- mvrnorm(n = 100, mu, Sigma)
```

The mean vector of these 100 observations is (1.093, 2.137).

Question 2 (40 Points) Write R Functions

Write an R function that outputs the first n terms in the Fibonacci sequence, where n is the only input value of the function. Use the function to find `fibonacci(10)`.

Answer:

```
# construct the function
fibonacci <- function(n)
{
  x = numeric(n)
  x[1:2] = c(1,1)
  for(i in 3:n) x[i] = x[i-1] + x[i-2]
  return(x)
}
```

```
}
```

```
# test the function  
fibonacci(10)
```

```
## [1] 1 1 2 3 5 8 13 21 34 55
```

The 10th value in a Fibonacci sequence is 55.

Question 3 (20 Points) Using AI Tools

You can also use AI tools to help you calculate the Fibonacci sequence. Ask ChatGPT how this can be done. Is the answer efficient? Why? If its not a good solution, ask it to provide a better answer. Make sure to report your prompt.

Answer:

I first used the prompt

"Can you give me a code to calculate the 10th value in a Fibonacci sequence?"

Then ChatGPT provided this answer

```
fibonacci <- function(n) {  
  if (n <= 1) return(n)  
  return(fibonacci(n - 1) + fibonacci(n - 2))  
}  
  
tenth_value <- fibonacci(10)  
print(tenth_value) # Output will be 55
```

```
## [1] 55
```

However, this is very inefficient since it needs to calculate the entire previous sequence. Hence this is very slow for large numbers. Hence, I asked the follow-up prompt:

"This is very inefficient. Can you provide a better solution?"

The new answer is

```
fibonacci <- function(n) {  
  a <- 0  
  b <- 1  
  
  for (i in 2:n) {  
    temp <- a + b  
    a <- b  
    b <- temp  
  }  
  
  return(b)  
}  
  
tenth_value <- fibonacci(10)  
print(tenth_value) # Output will be 55
```

```
## [1] 55
```

The time complexity of this code is $O(n)$.