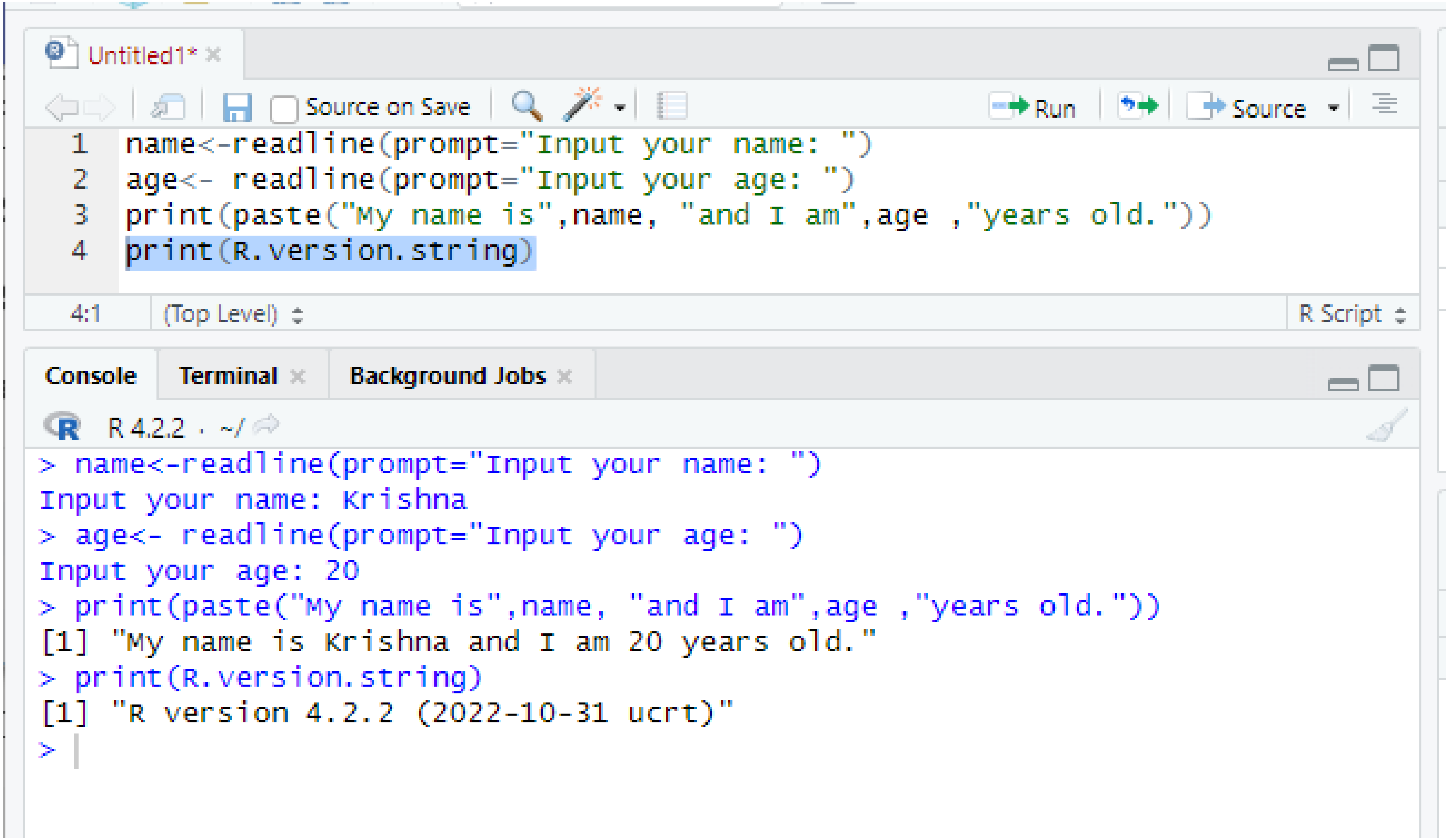
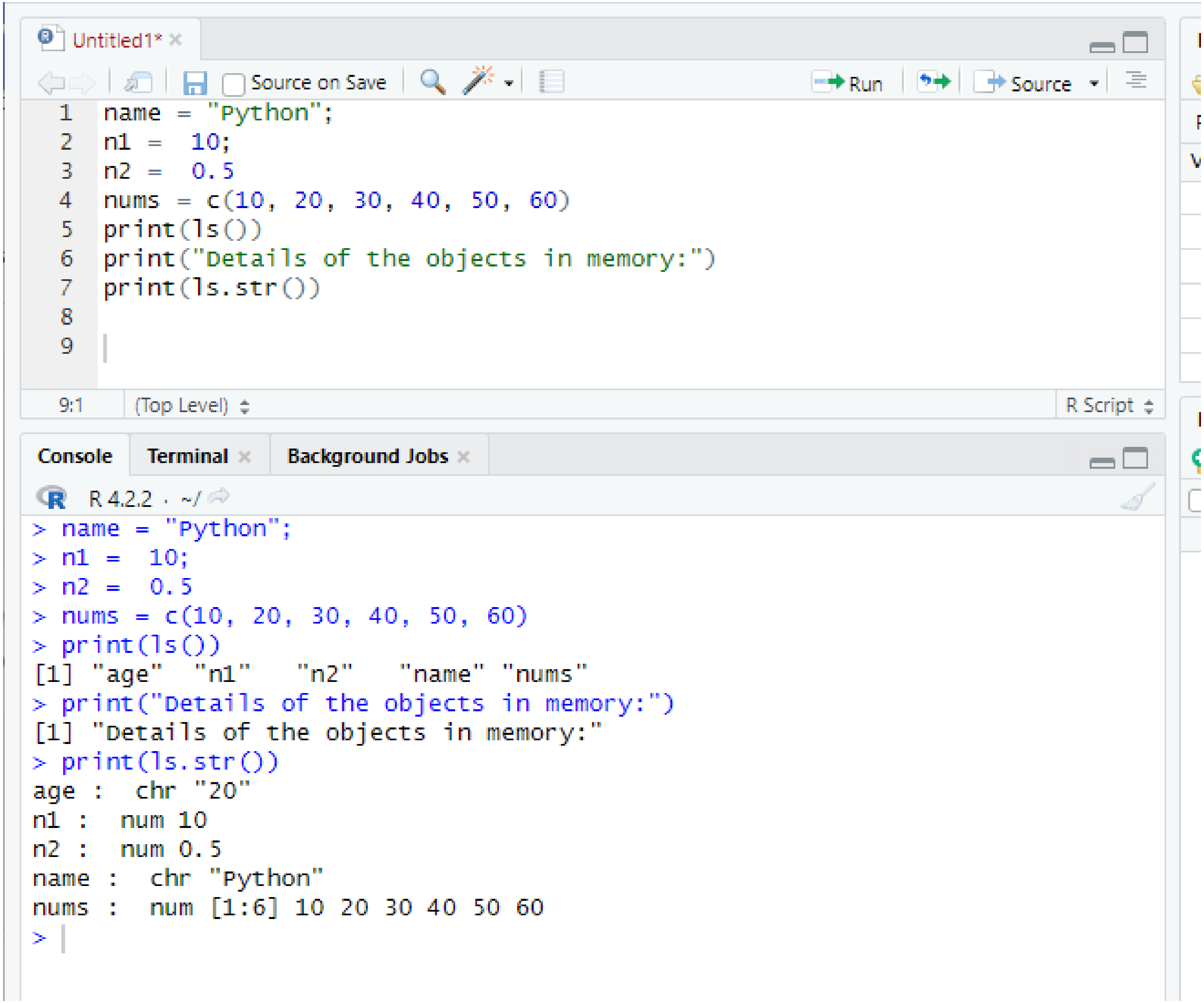


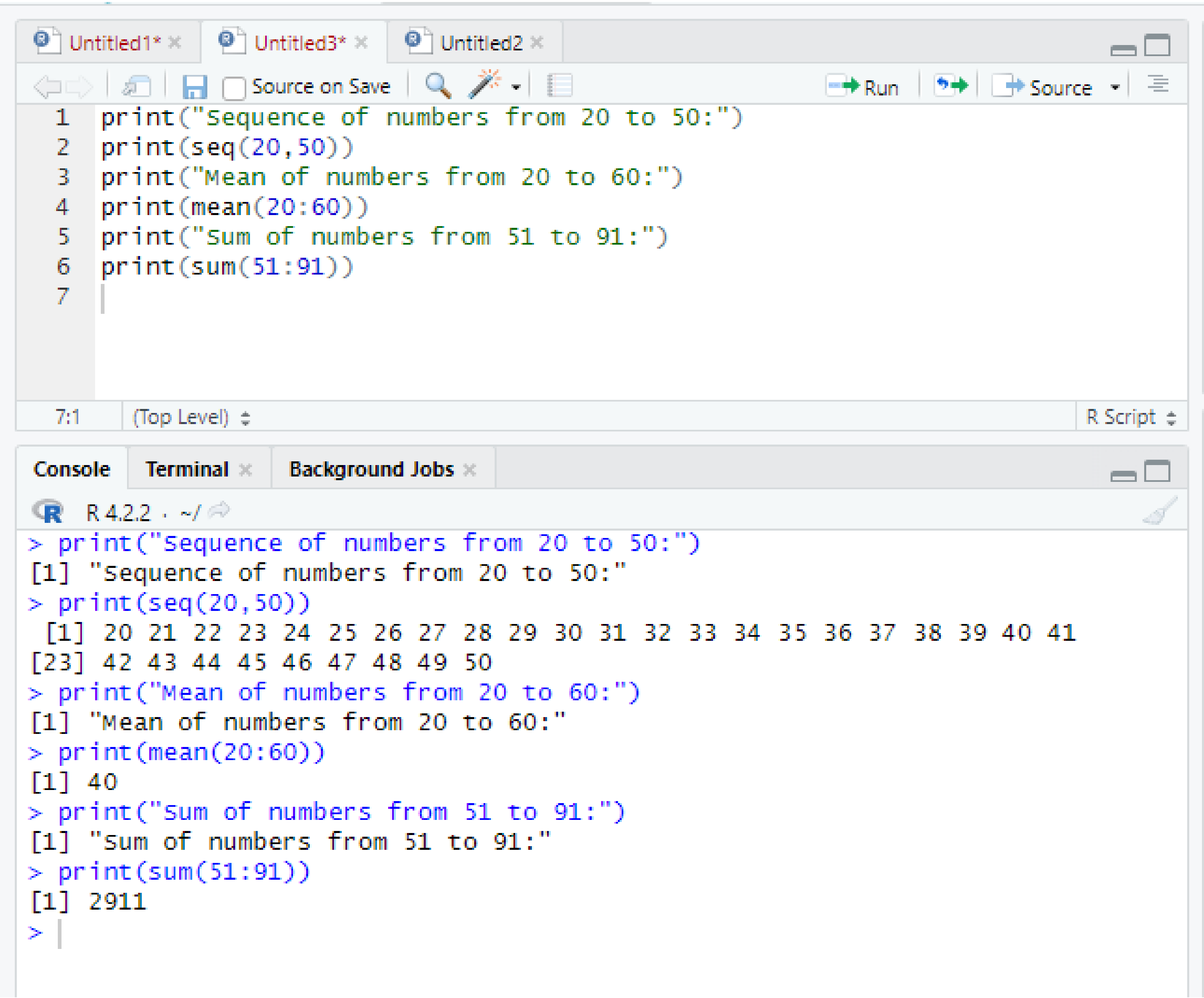
```
name<-readline(prompt="Input your name: ")
age<- readline(prompt="Input your age: ")
print(paste("My name is",name, "and I am",age ,"years old."))
print(R.version.string)
```

```
> name<-readline(prompt="Input your name: ")
Input your name: Krishna
> age<- readline(prompt="Input your age: ")
Input your age: 20
> print(paste("My name is",name, "and I am",age ,"years old."))
[1] "My name is Krishna and I am 20 years old."
> print(R.version.string)
[1] "R version 4.2.2 (2022-10-31 ucrt)"
> 
```



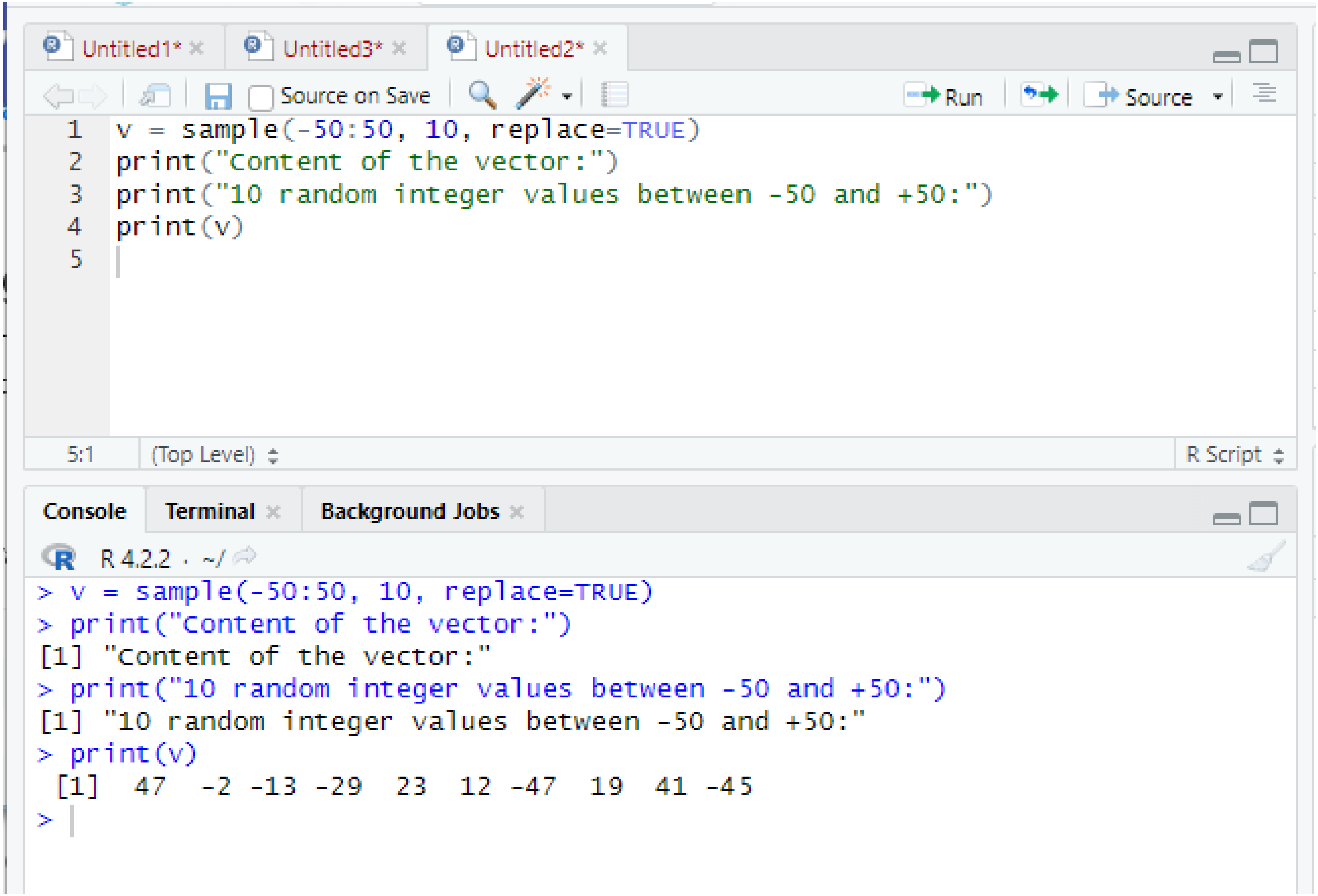
```
name = "Python";
n1 =  10;
n2 =  0.5
nums = c(10, 20, 30, 40, 50, 60)
print(ls())
print("Details of the objects in memory:")
print(ls.str())

```

```
> name = "Python";
> n1 =  10;
> n2 =  0.5
> nums = c(10, 20, 30, 40, 50, 60)
> print(ls())
[1] "age"  "n1"   "n2"   "name" "nums"
> print("Details of the objects in memory:")
[1] "Details of the objects in memory:"
> print(ls.str())
age :  chr "20"
n1 :  num 10
n2 :  num 0.5
name :  chr "Python"
nums :  num [1:6] 10 20 30 40 50 60
> 
```



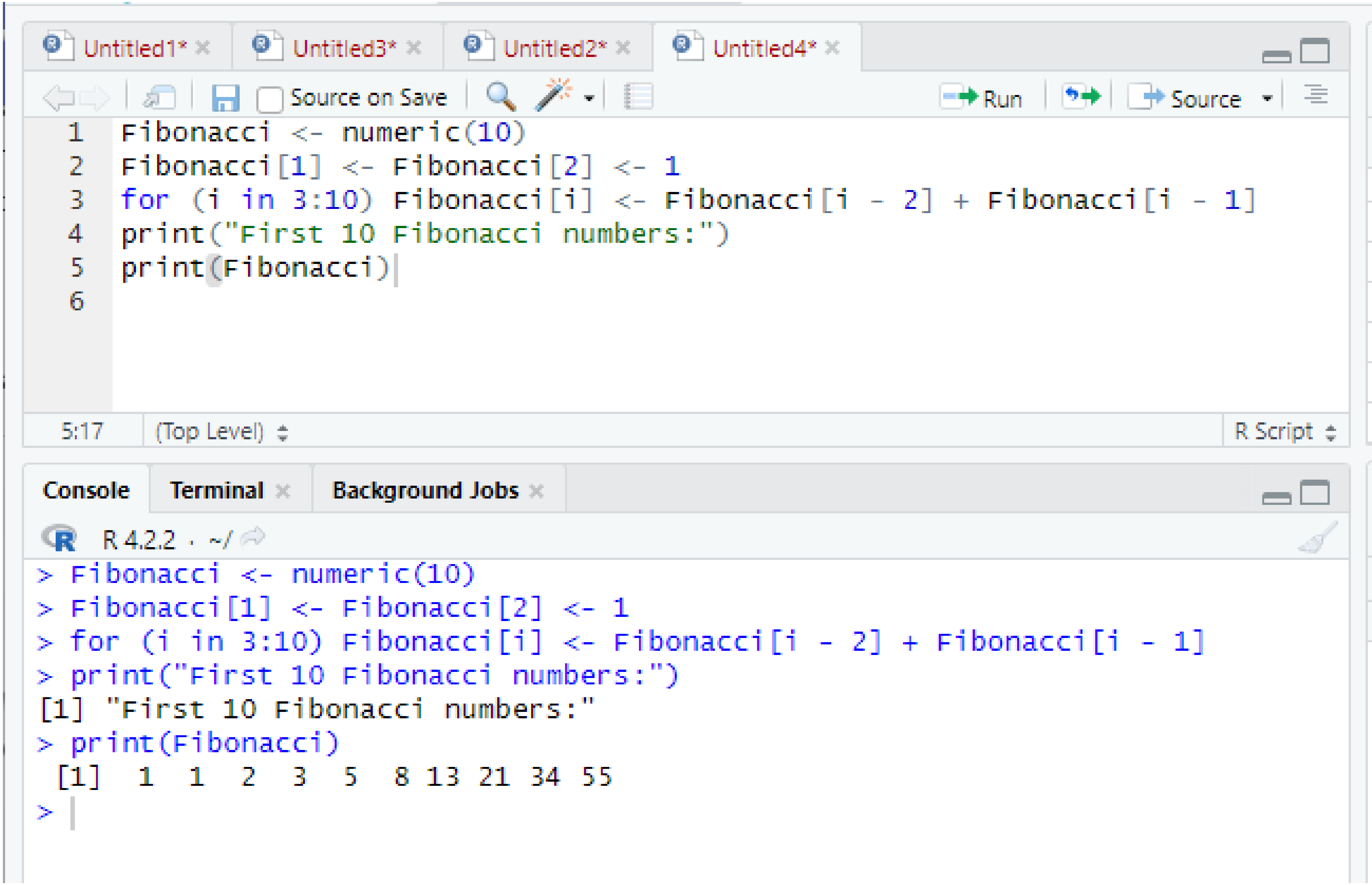
```
print("Sequence of numbers from 20 to 50:")
print(seq(20,50))
print("Mean of numbers from 20 to 60:")
print(mean(20:60))
print("Sum of numbers from 51 to 91:")
print(sum(51:91))
```

```
> print("Sequence of numbers from 20 to 50:")
[1] "Sequence of numbers from 20 to 50:"
> print(seq(20,50))
 [1] 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41
[23] 42 43 44 45 46 47 48 49 50
> print("Mean of numbers from 20 to 60:")
[1] "Mean of numbers from 20 to 60:"
> print(mean(20:60))
[1] 40
> print("Sum of numbers from 51 to 91:")
[1] "Sum of numbers from 51 to 91:"
> print(sum(51:91))
[1] 2911
> 
```



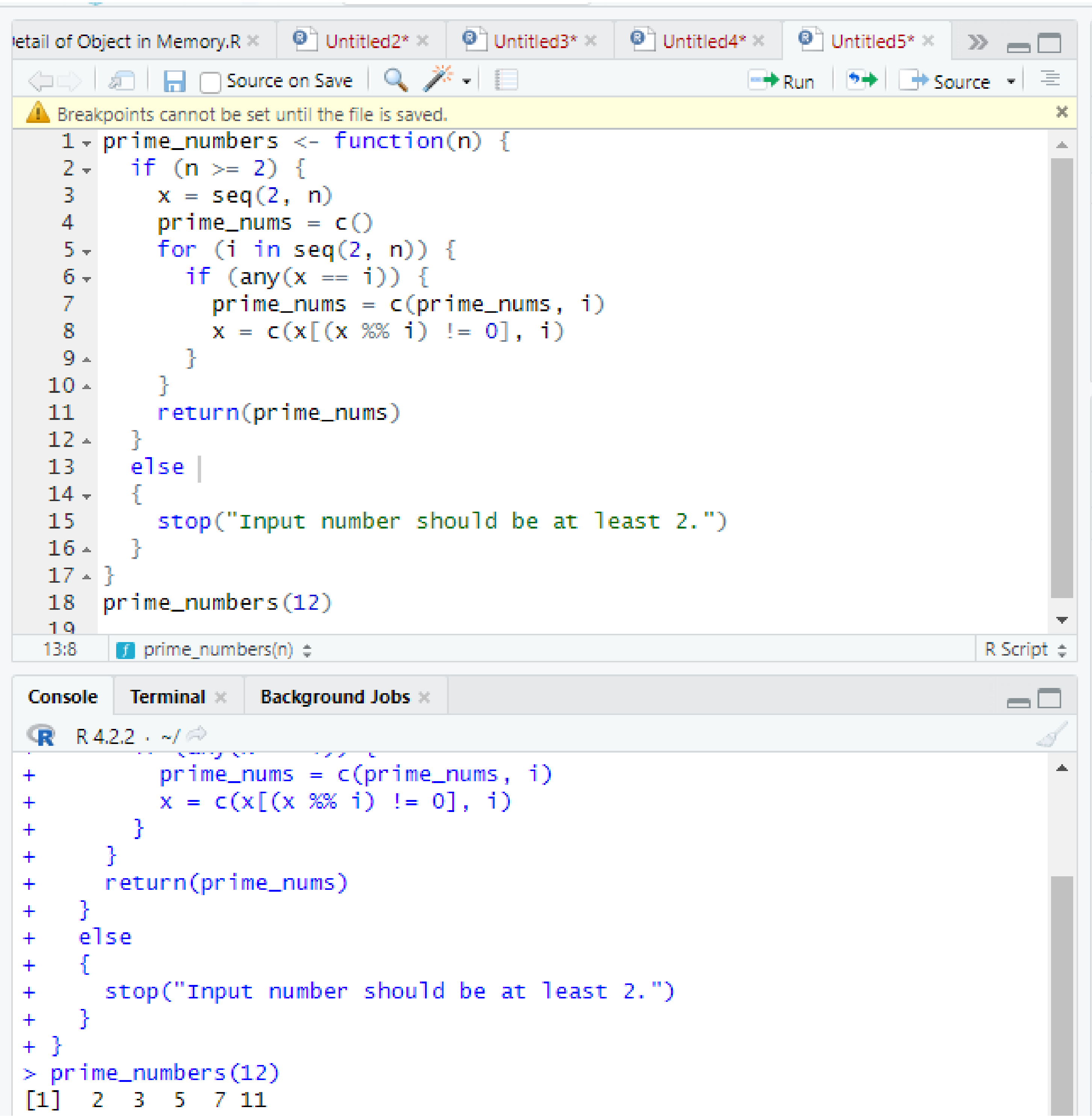
```r
v = sample(-50:50, 10, replace=TRUE)
print("Content of the vector:")
print("10 random integer values between -50 and +50:")
print(v)
```

```
> v = sample(-50:50, 10, replace=TRUE)
> print("Content of the vector:")
[1] "Content of the vector:"
> print("10 random integer values between -50 and +50:")
[1] "10 random integer values between -50 and +50:"
> print(v)
 [1]  47  -2 -13 -29  23  12 -47  19  41 -45
> 
```



```r
Fibonacci <- numeric(10)
Fibonacci[1] <- Fibonacci[2] <- 1
for (i in 3:10) Fibonacci[i] <- Fibonacci[i - 2] + Fibonacci[i - 1]
print("First 10 Fibonacci numbers:")
print(Fibonacci)
```

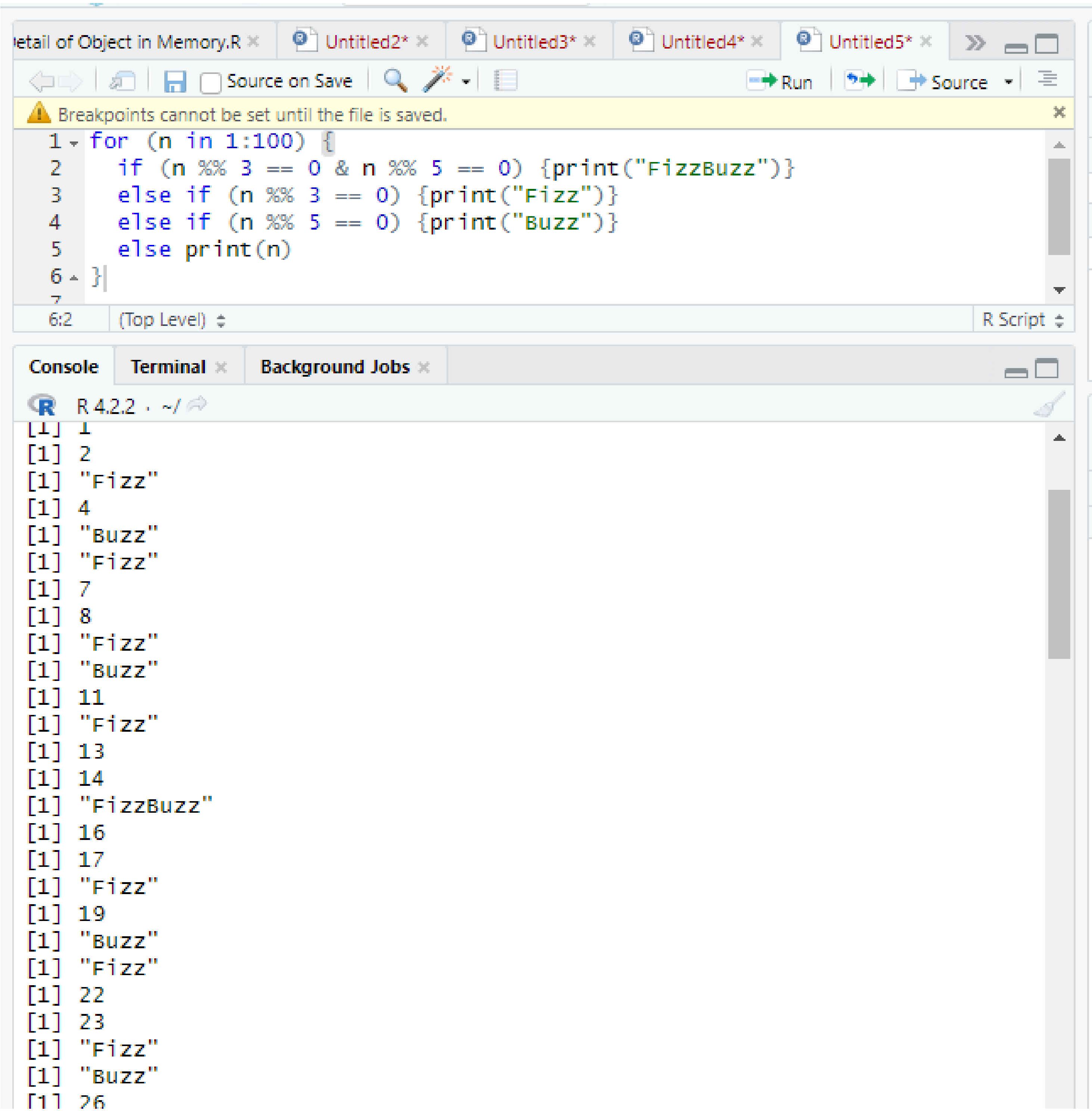
```
> Fibonacci <- numeric(10)
> Fibonacci[1] <- Fibonacci[2] <- 1
> for (i in 3:10) Fibonacci[i] <- Fibonacci[i - 2] + Fibonacci[i - 1]
> print("First 10 Fibonacci numbers:")
[1] "First 10 Fibonacci numbers:"
> print(Fibonacci)
 [1]  1  1  2  3  5  8 13 21 34 55
> 
```



Breakpoints cannot be set until the file is saved.

```r
prime_numbers <- function(n) {
  if (n >= 2) {
    x = seq(2, n)
    prime_nums = c()
    for (i in seq(2, n)) {
      if (any(x == i)) {
        prime_nums = c(prime_nums, i)
        x = c(x[(x %% i) != 0], i)
      }
    }
    return(prime_nums)
  }
  else 
  {
    stop("Input number should be at least 2.")
  }
}
prime_numbers(12)
```

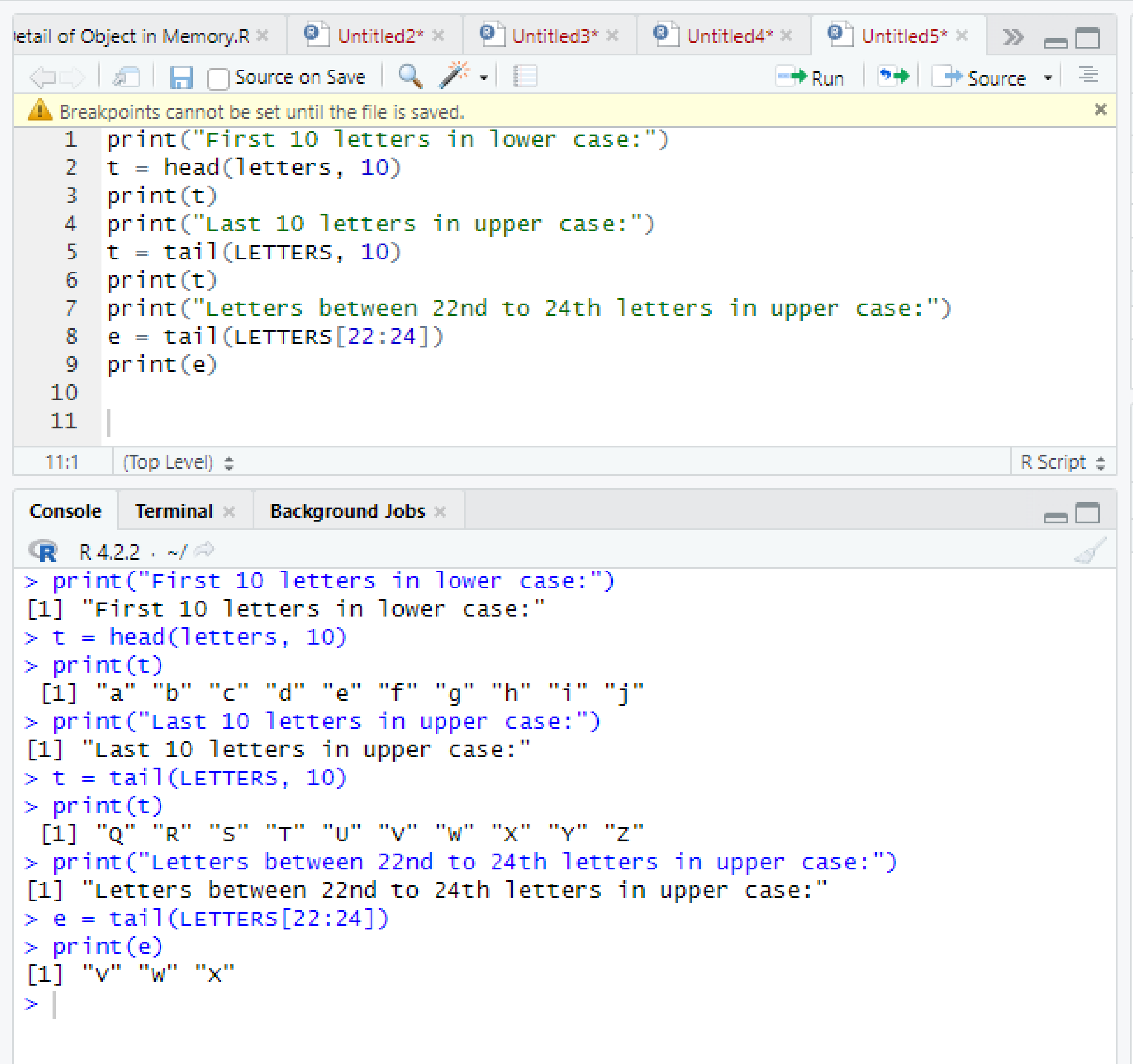
```
+         prime_nums = c(prime_nums, i)
+         x = c(x[(x %% i) != 0], i)
+       }
+     }
+     return(prime_nums)
+   }
+   else
+   {
+     stop("Input number should be at least 2.")
+   }
+ }
> prime_numbers(12)
[1]  2  3  5  7 11
```



Breakpoints cannot be set until the file is saved.

```r
for (n in 1:100) {
  if (n %% 3 == 0 & n %% 5 == 0) {print("FizzBuzz")}
  else if (n %% 3 == 0) {print("Fizz")}
  else if (n %% 5 == 0) {print("Buzz")}
  else print(n)
}
```

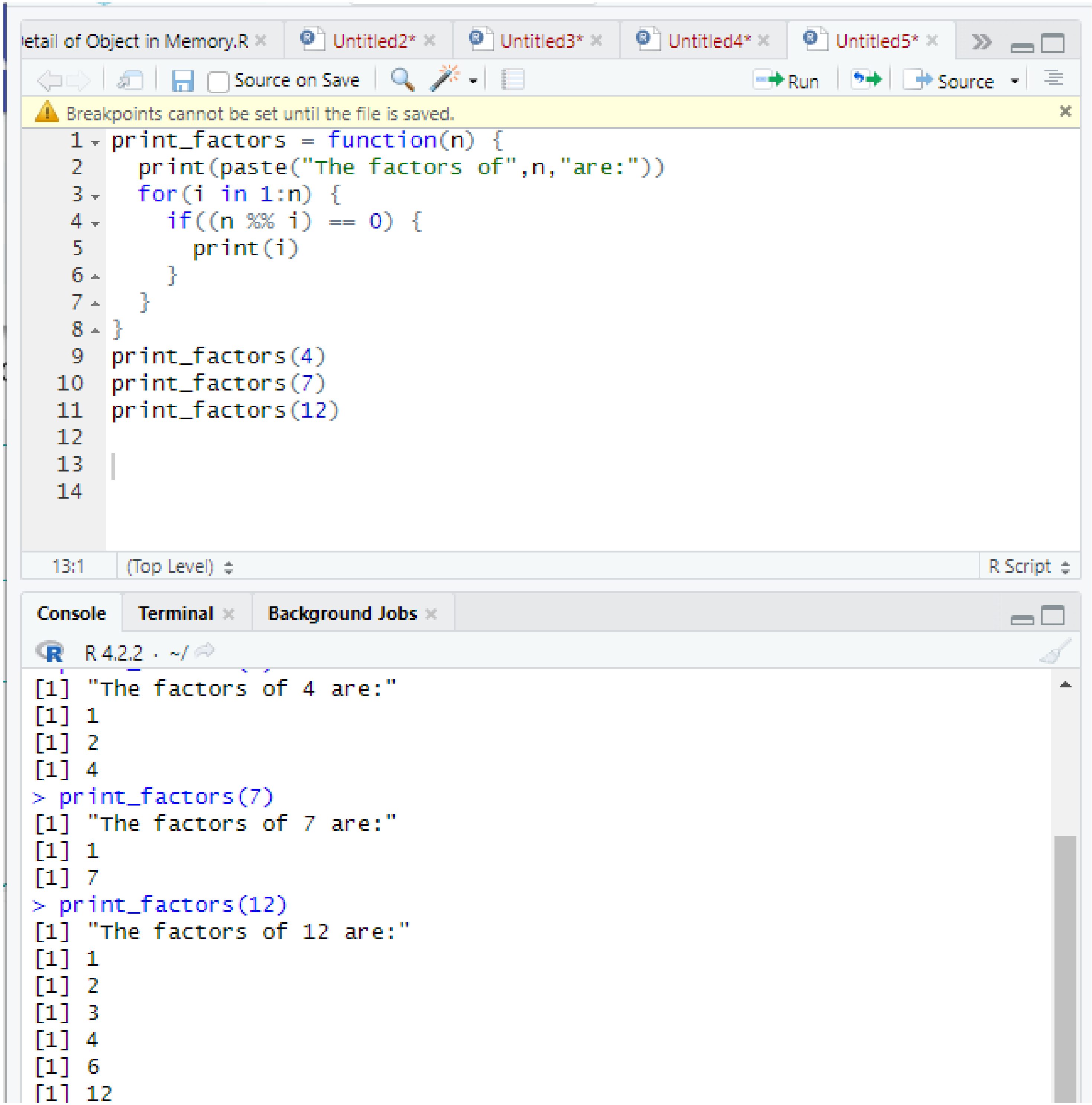
```
[1] 1
[1] 2
[1] "Fizz"
[1] 4
[1] "Buzz"
[1] "Fizz"
[1] 7
[1] 8
[1] "Fizz"
[1] "Buzz"
[1] 11
[1] "Fizz"
[1] 13
[1] 14
[1] "FizzBuzz"
[1] 16
[1] 17
[1] "Fizz"
[1] 19
[1] "Buzz"
[1] "Fizz"
[1] 22
[1] 23
[1] "Fizz"
[1] "Buzz"
[1] 26
```



Breakpoints cannot be set until the file is saved.

```
print("First 10 letters in lower case:")
t = head(letters, 10)
print(t)
print("Last 10 letters in upper case:")
t = tail(LETTERS, 10)
print(t)
print("Letters between 22nd to 24th letters in upper case:")
e = tail(LETTERS[22:24])
print(e)


```

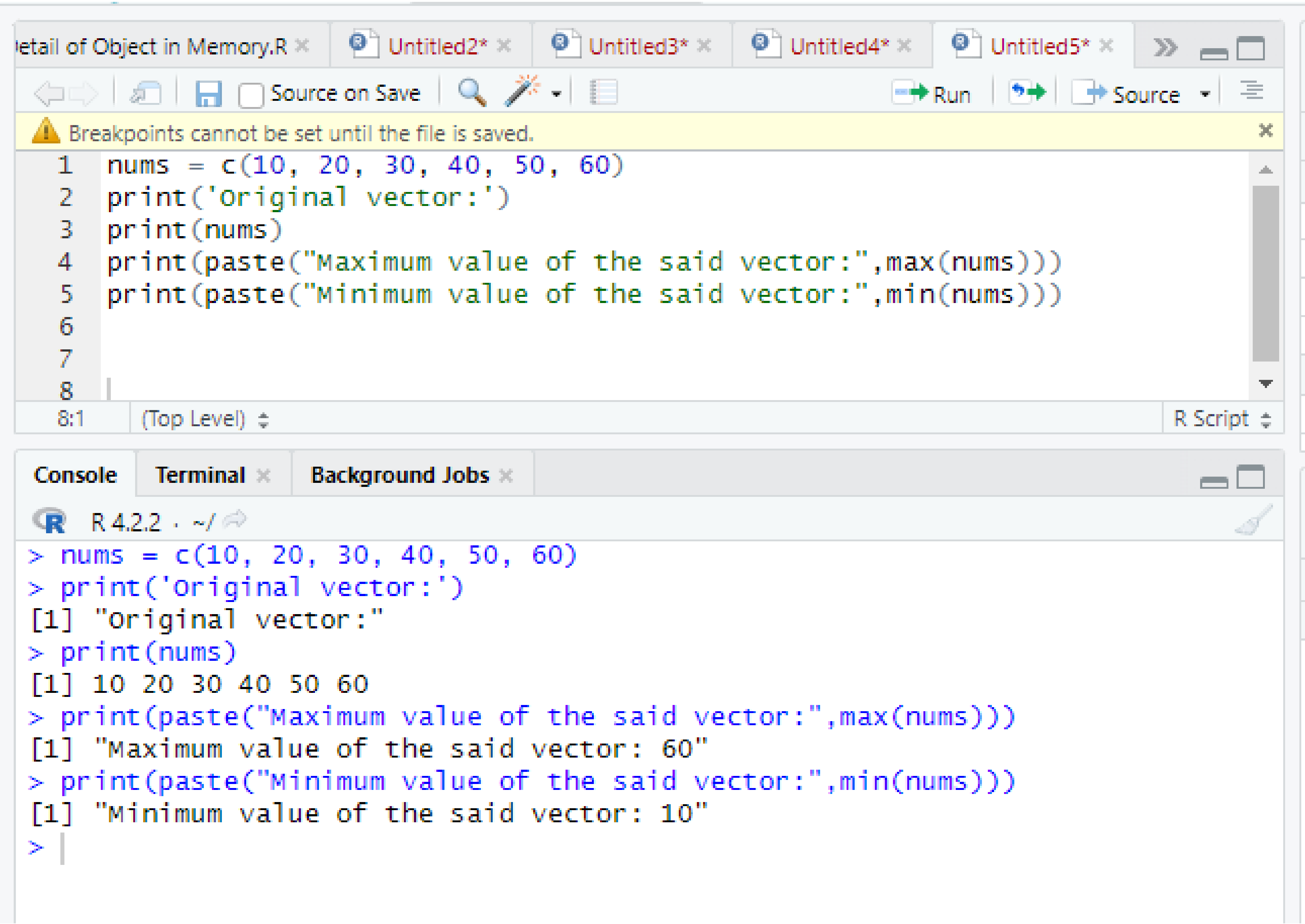
```
> print("First 10 letters in lower case:")
[1] "First 10 letters in lower case:"
> t = head(letters, 10)
> print(t)
 [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j"
> print("Last 10 letters in upper case:")
[1] "Last 10 letters in upper case:"
> t = tail(LETTERS, 10)
> print(t)
 [1] "Q" "R" "S" "T" "U" "V" "W" "X" "Y" "Z"
> print("Letters between 22nd to 24th letters in upper case:")
[1] "Letters between 22nd to 24th letters in upper case:"
> e = tail(LETTERS[22:24])
> print(e)
[1] "V" "W" "X"
> 
```



Breakpoints cannot be set until the file is saved.

```
print_factors = function(n) {
  print(paste("The factors of",n,"are:"))
  for(i in 1:n) {
    if((n %% i) == 0) {
      print(i)
    }
  }
}
print_factors(4)
print_factors(7)
print_factors(12)
```

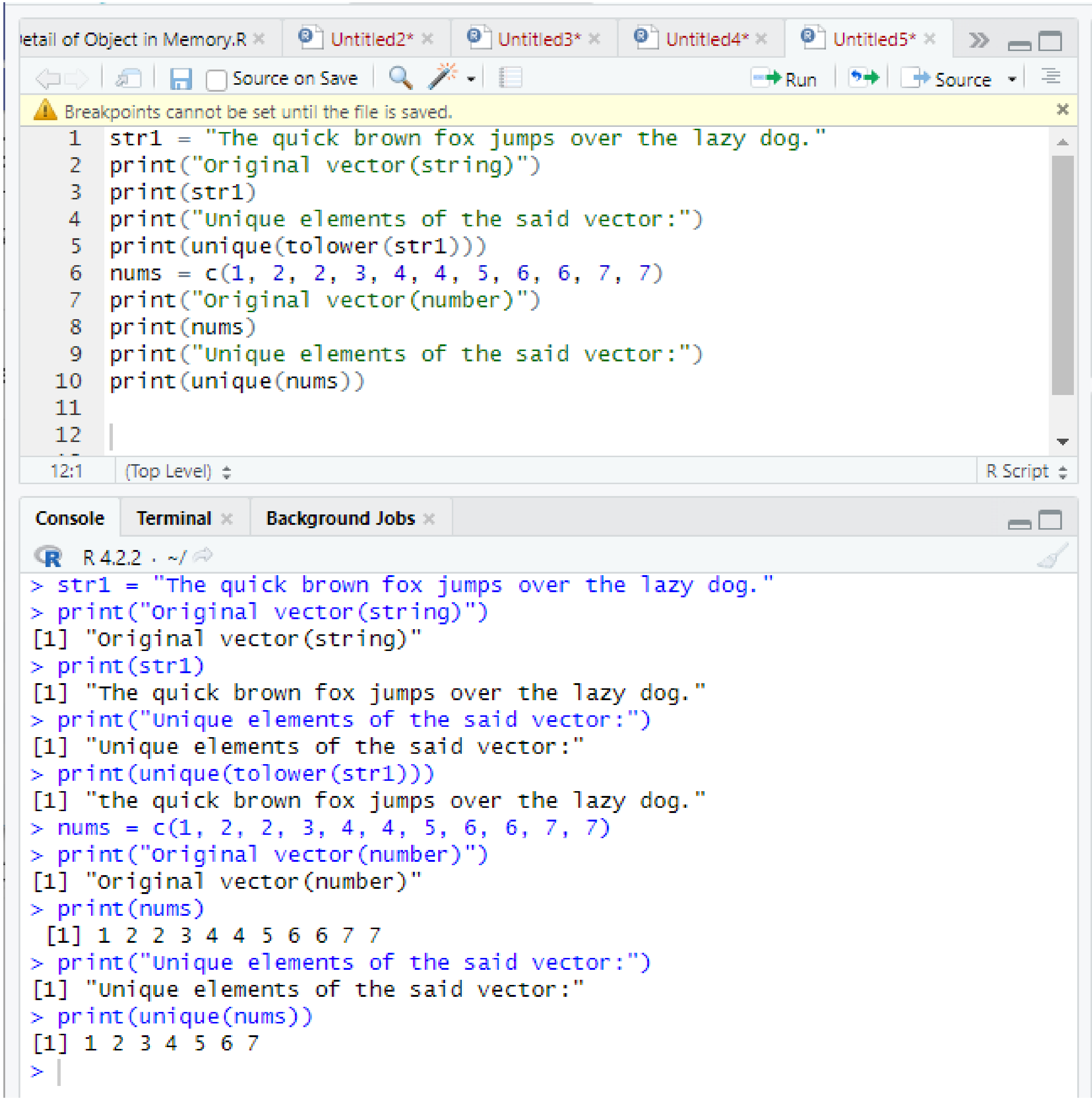
```
[1] "The factors of 4 are:"
[1] 1
[1] 2
[1] 4
> print_factors(7)
[1] "The factors of 7 are:"
[1] 1
[1] 7
> print_factors(12)
[1] "The factors of 12 are:"
[1] 1
[1] 2
[1] 3
[1] 4
[1] 6
[1] 12
```



Breakpoints cannot be set until the file is saved.

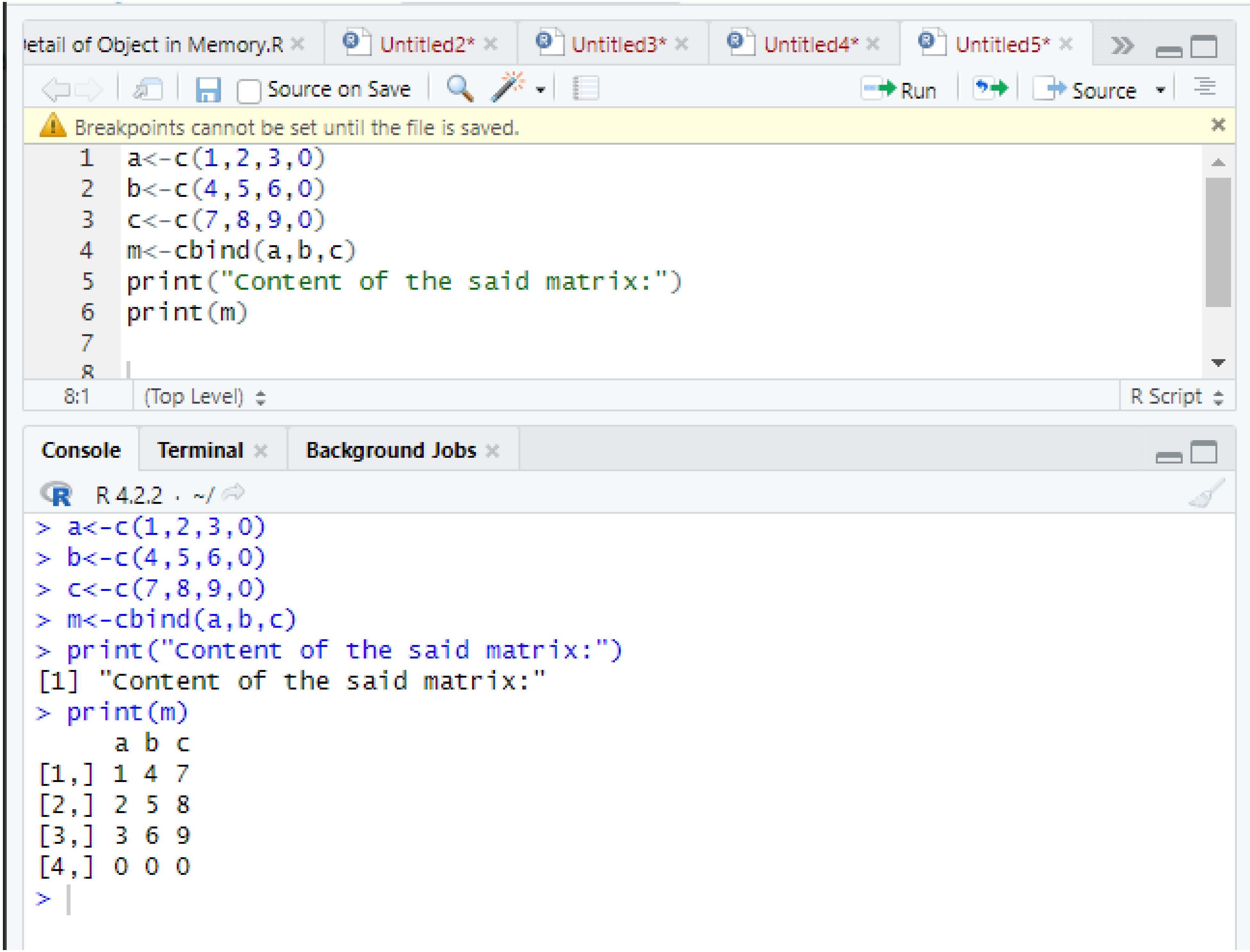
```
nums = c(10, 20, 30, 40, 50, 60)
print('Original vector:')
print(nums)
print(paste("Maximum value of the said vector:",max(nums)))
print(paste("Minimum value of the said vector:",min(nums)))
```

```
> nums = c(10, 20, 30, 40, 50, 60)
> print('Original vector:')
[1] "Original vector:"
> print(nums)
[1] 10 20 30 40 50 60
> print(paste("Maximum value of the said vector:",max(nums)))
[1] "Maximum value of the said vector: 60"
> print(paste("Minimum value of the said vector:",min(nums)))
[1] "Minimum value of the said vector: 10"
> 
```



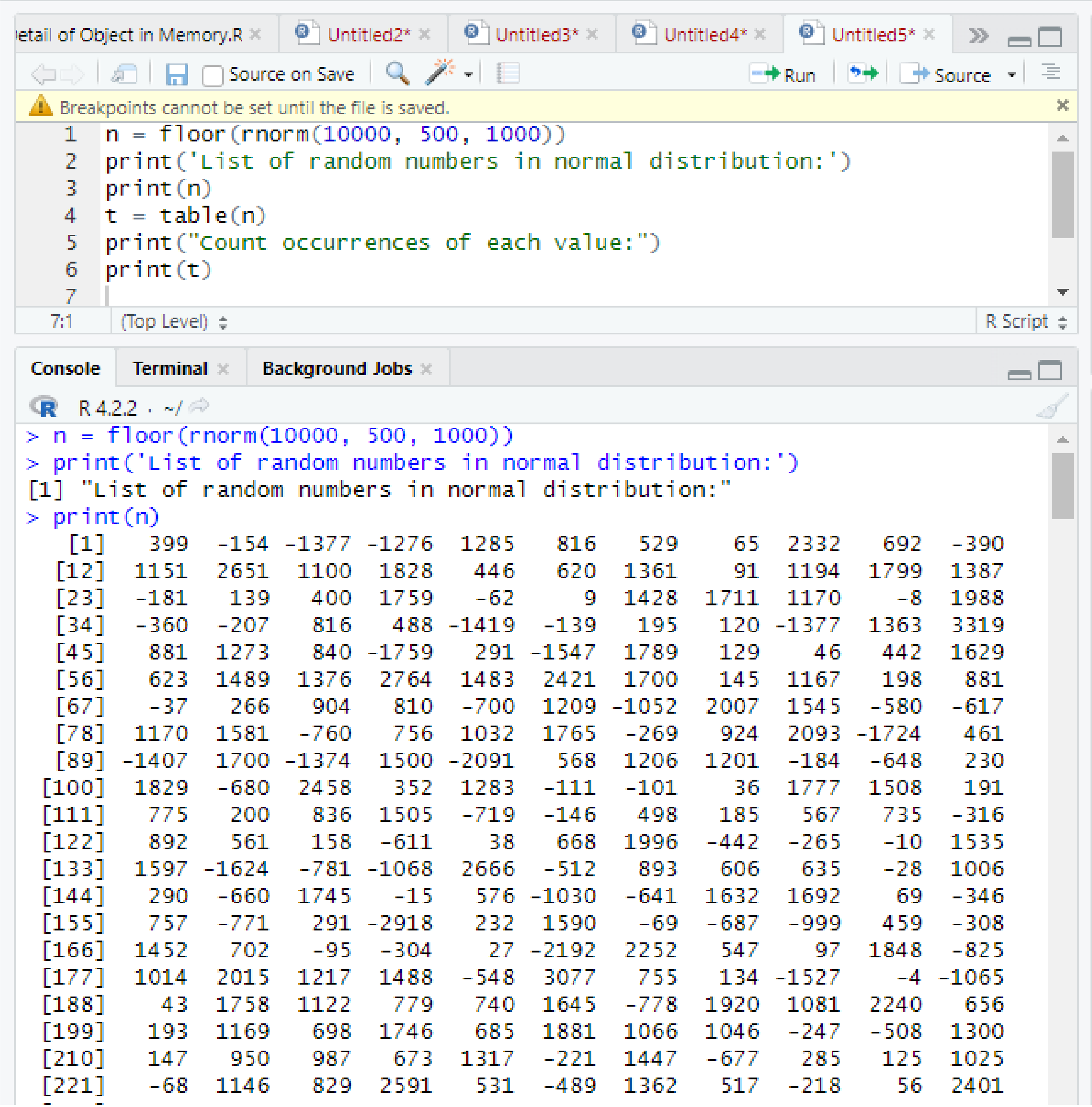
```r
str1 = "The quick brown fox jumps over the lazy dog."
print("Original vector(string)")
print(str1)
print("Unique elements of the said vector:")
print(unique(tolower(str1)))
nums = c(1, 2, 2, 3, 4, 4, 5, 6, 6, 7, 7)
print("Original vector(number)")
print(nums)
print("Unique elements of the said vector:")
print(unique(nums))


```

```
> str1 = "The quick brown fox jumps over the lazy dog."
> print("Original vector(string)")
[1] "Original vector(string)"
> print(str1)
[1] "The quick brown fox jumps over the lazy dog."
> print("Unique elements of the said vector:")
[1] "Unique elements of the said vector:"
> print(unique(tolower(str1)))
[1] "the quick brown fox jumps over the lazy dog."
> nums = c(1, 2, 2, 3, 4, 4, 5, 6, 6, 7, 7)
> print("Original vector(number)")
[1] "Original vector(number)"
> print(nums)
 [1] 1 2 2 3 4 4 5 6 6 7 7
> print("Unique elements of the said vector:")
[1] "Unique elements of the said vector:"
> print(unique(nums))
[1] 1 2 3 4 5 6 7
> 
```



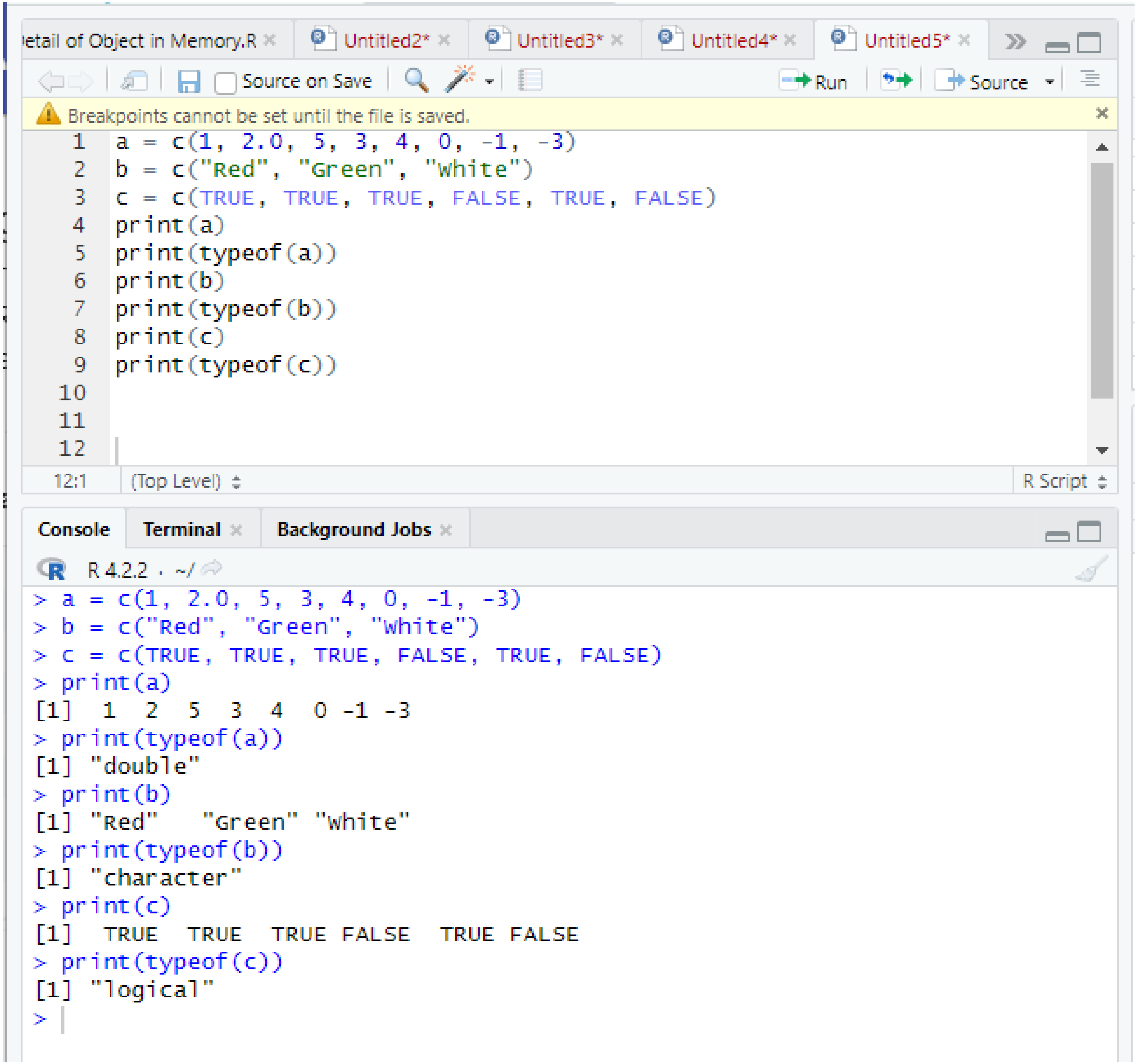
```
a<-c(1,2,3,0)
b<-c(4,5,6,0)
c<-c(7,8,9,0)
m<-cbind(a,b,c)
print("Content of the said matrix:")
print(m)
```

```
> a<-c(1,2,3,0)
> b<-c(4,5,6,0)
> c<-c(7,8,9,0)
> m<-cbind(a,b,c)
> print("Content of the said matrix:")
[1] "Content of the said matrix:"
> print(m)
     a b c
[1,] 1 4 7
[2,] 2 5 8
[3,] 3 6 9
[4,] 0 0 0
> 
```



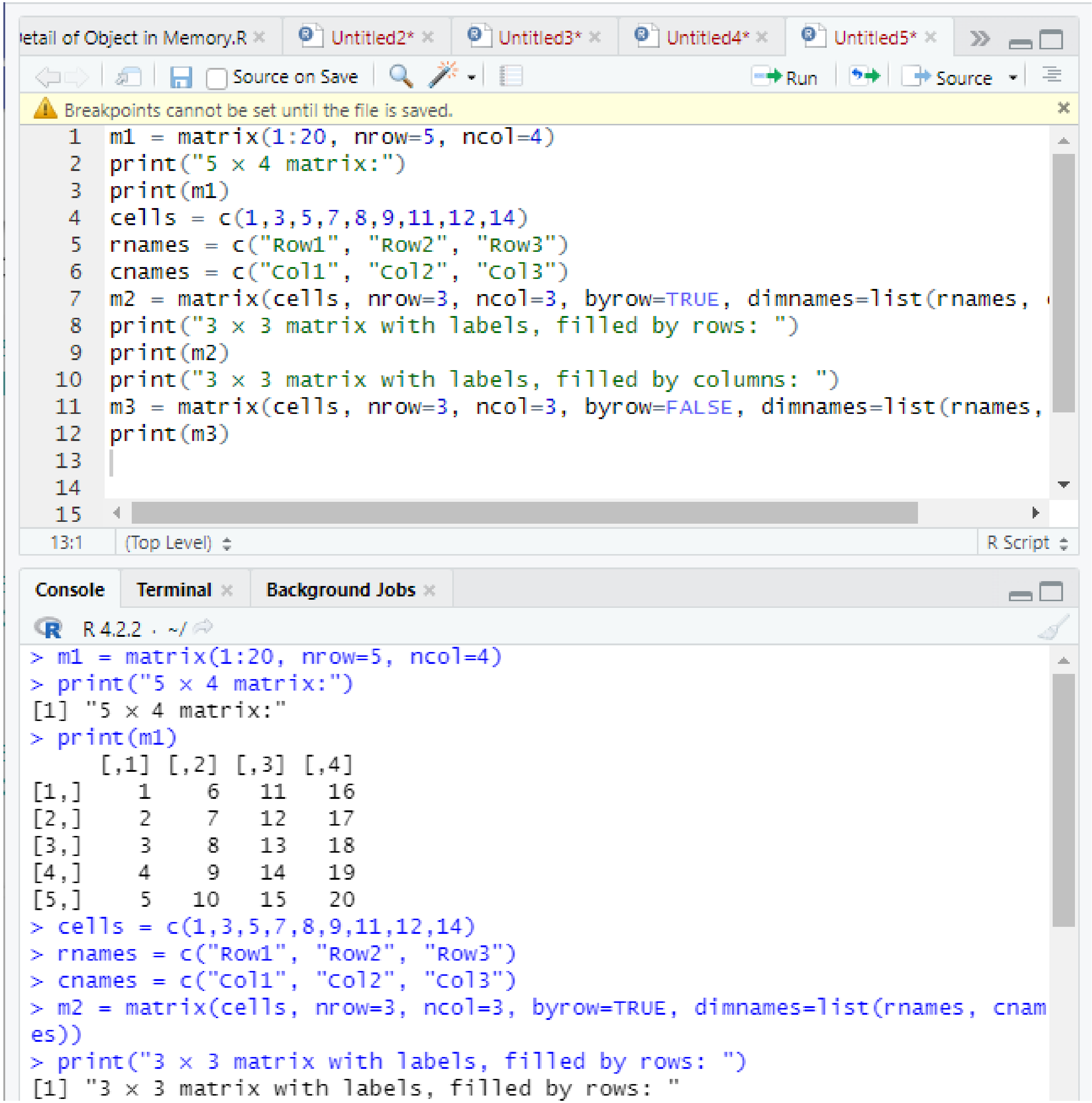
```
n = floor(rnorm(10000, 500, 1000))
print('List of random numbers in normal distribution:')
print(n)
t = table(n)
print("Count occurrences of each value:")
print(t)
```

```
> n = floor(rnorm(10000, 500, 1000))
> print('List of random numbers in normal distribution:')
[1] "List of random numbers in normal distribution:"
> print(n)
  [1]   399  -154 -1377 -1276  1285   816   529    65  2332   692  -390
 [12]  1151  2651  1100  1828   446   620  1361    91  1194  1799  1387
 [23]  -181   139   400  1759   -62     9  1428  1711  1170    -8  1988
 [34]  -360  -207   816   488 -1419  -139   195   120 -1377  1363  3319
 [45]   881  1273   840 -1759   291 -1547  1789   129    46   442  1629
 [56]   623  1489  1376  2764  1483  2421  1700   145  1167   198   881
 [67]   -37   266   904   810  -700  1209 -1052  2007  1545  -580  -617
 [78]  1170  1581  -760   756  1032  1765  -269   924  2093 -1724   461
 [89] -1407  1700 -1374  1500 -2091   568  1206  1201  -184  -648   230
[100]  1829  -680  2458   352  1283  -111  -101    36  1777  1508   191
[111]   775   200   836  1505  -719  -146   498   185   567   735  -316
[122]   892   561   158  -611    38   668  1996  -442  -265   -10  1535
[133]  1597 -1624  -781 -1068  2666  -512   893   606   635   -28  1006
[144]   290  -660  1745   -15   576 -1030  -641  1632  1692    69  -346
[155]   757  -771   291 -2918   232  1590   -69  -687  -999   459  -308
[166]  1452   702   -95  -304    27 -2192  2252   547    97  1848  -825
[177]  1014  2015  1217  1488  -548  3077   755   134 -1527    -4 -1065
[188]    43  1758  1122   779   740  1645  -778  1920  1081  2240   656
[199]   193  1169   698  1746   685  1881  1066  1046  -247  -508  1300
[210]   147   950   987   673  1317  -221  1447  -677   285   125  1025
[221]   -68  1146   829  2591   531  -489  1362   517  -218    56  2401
```



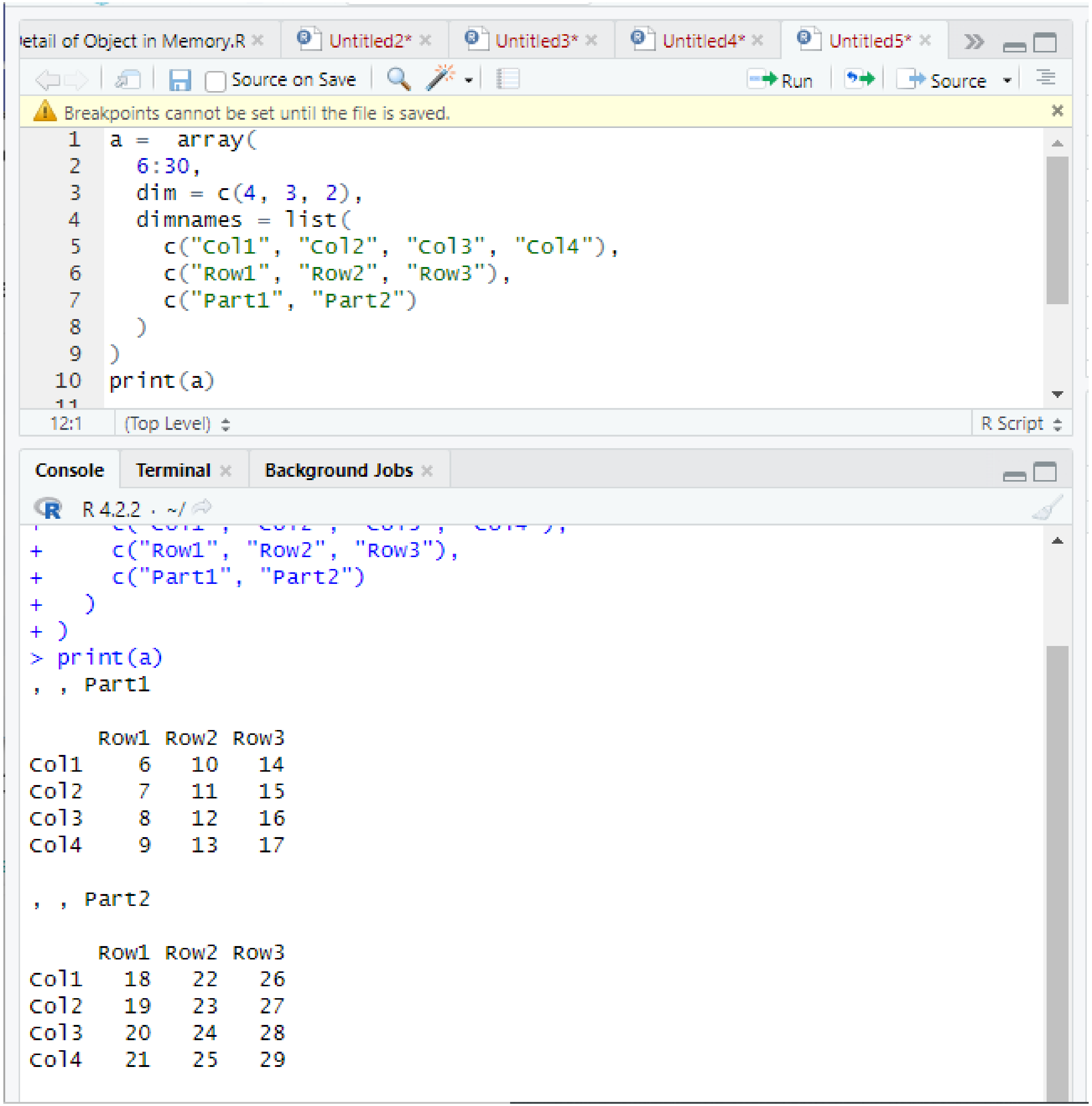
```
a = c(1, 2.0, 5, 3, 4, 0, -1, -3)
b = c("Red", "Green", "white")
c = c(TRUE, TRUE, TRUE, FALSE, TRUE, FALSE)
print(a)
print(typeof(a))
print(b)
print(typeof(b))
print(c)
print(typeof(c))
```

```
> a = c(1, 2.0, 5, 3, 4, 0, -1, -3)
> b = c("Red", "Green", "white")
> c = c(TRUE, TRUE, TRUE, FALSE, TRUE, FALSE)
> print(a)
[1]  1  2  5  3  4  0 -1 -3
> print(typeof(a))
[1] "double"
> print(b)
[1] "Red"   "Green" "white"
> print(typeof(b))
[1] "character"
> print(c)
[1]  TRUE  TRUE  TRUE FALSE  TRUE FALSE
> print(typeof(c))
[1] "logical"
> 
```



```
m1 = matrix(1:20, nrow=5, ncol=4)
print("5 x 4 matrix:")
print(m1)
cells = c(1,3,5,7,8,9,11,12,14)
rnames = c("Row1", "Row2", "Row3")
cnames = c("Col1", "Col2", "Col3")
m2 = matrix(cells, nrow=3, ncol=3, byrow=TRUE, dimnames=list(rnames,
print("3 x 3 matrix with labels, filled by rows: ")
print(m2)
print("3 x 3 matrix with labels, filled by columns: ")
m3 = matrix(cells, nrow=3, ncol=3, byrow=FALSE, dimnames=list(rnames,
print(m3)
```

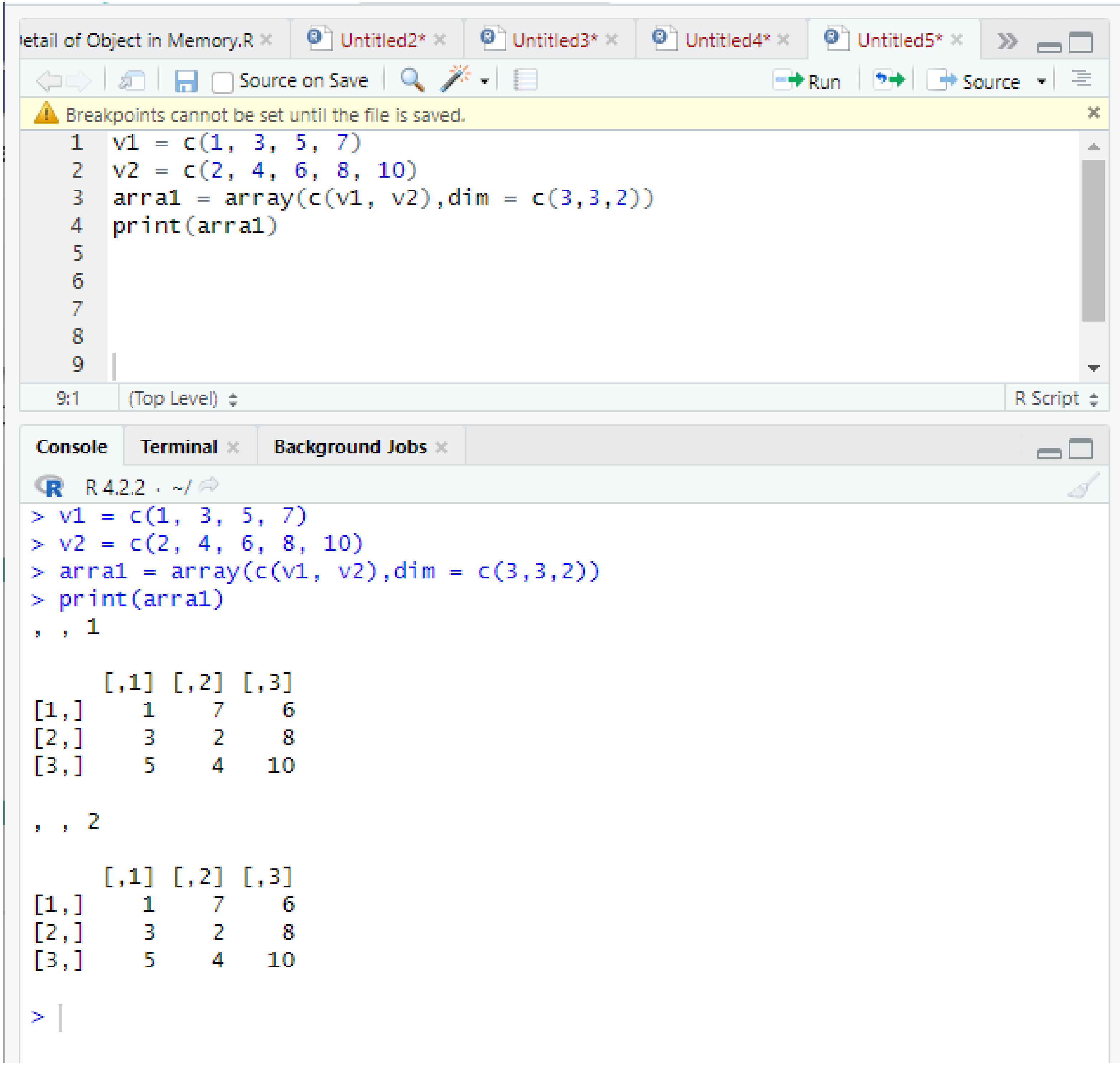
```
> m1 = matrix(1:20, nrow=5, ncol=4)
> print("5 x 4 matrix:")
[1] "5 x 4 matrix:"
> print(m1)
     [,1] [,2] [,3] [,4]
[1,]    1    6   11   16
[2,]    2    7   12   17
[3,]    3    8   13   18
[4,]    4    9   14   19
[5,]    5   10   15   20
> cells = c(1,3,5,7,8,9,11,12,14)
> rnames = c("Row1", "Row2", "Row3")
> cnames = c("Col1", "Col2", "Col3")
> m2 = matrix(cells, nrow=3, ncol=3, byrow=TRUE, dimnames=list(rnames, cnam
es))
> print("3 x 3 matrix with labels, filled by rows: ")
[1] "3 x 3 matrix with labels, filled by rows: "
```



Breakpoints cannot be set until the file is saved.

```
a =  array(
  6:30,
  dim = c(4, 3, 2),
  dimnames = list(
    c("Col1", "Col2", "Col3", "Col4"),
    c("Row1", "Row2", "Row3"),
    c("Part1", "Part2")
  )
)
print(a)
```

```
+     c("Col1", "Col2", "Col3", "Col4"),
+     c("Row1", "Row2", "Row3"),
+     c("Part1", "Part2")
+   )
+ )
> print(a)
, , Part1

     Row1 Row2 Row3
Col1    6   10   14
Col2    7   11   15
Col3    8   12   16
Col4    9   13   17

, , Part2

     Row1 Row2 Row3
Col1   18   22   26
Col2   19   23   27
Col3   20   24   28
Col4   21   25   29
```

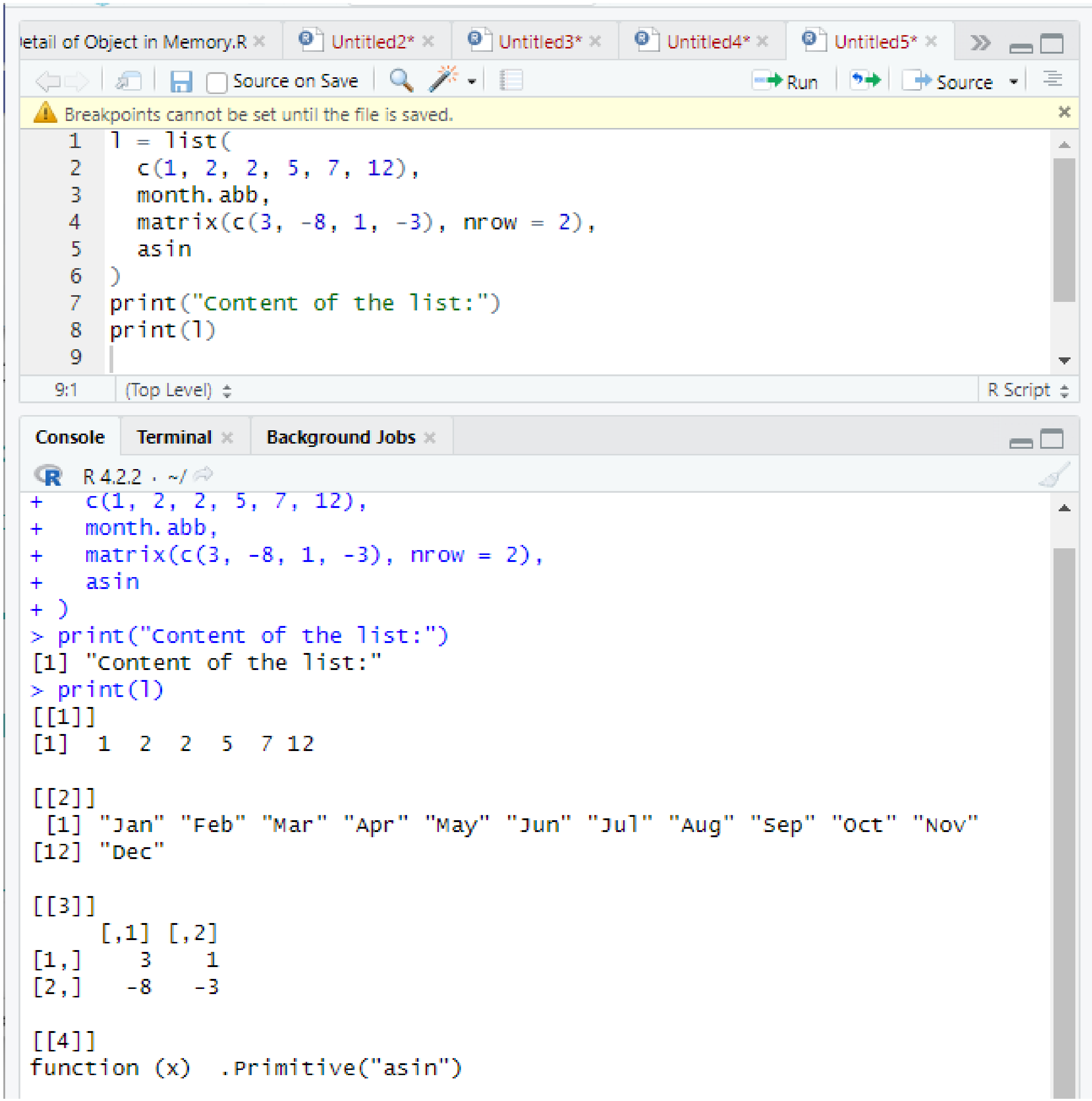


Breakpoints cannot be set until the file is saved.

```
v1 = c(1, 3, 5, 7)
v2 = c(2, 4, 6, 8, 10)
arra1 = array(c(v1, v2),dim = c(3,3,2))
print(arra1)
```

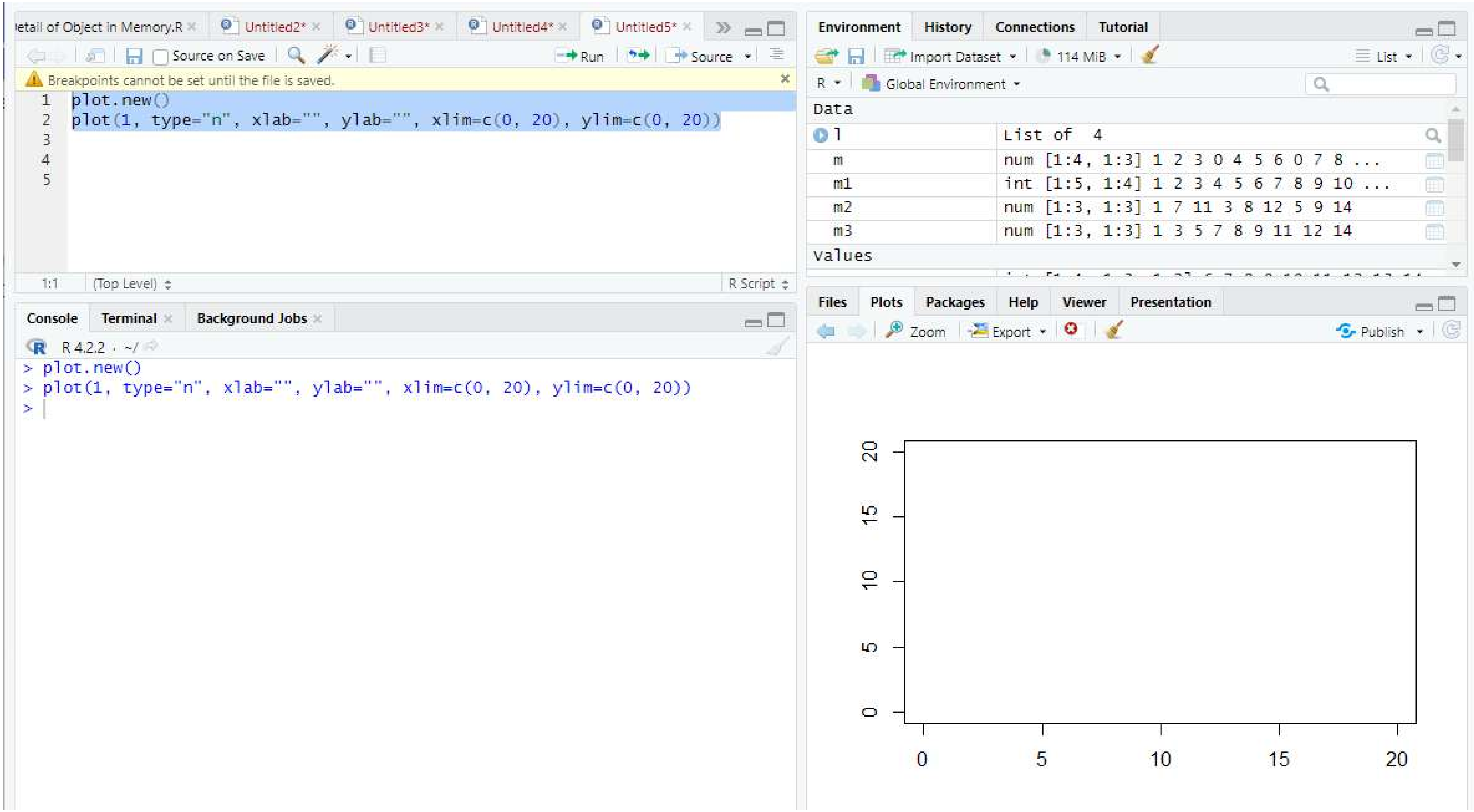
9:1 (Top Level) R Script

```
> v1 = c(1, 3, 5, 7)
> v2 = c(2, 4, 6, 8, 10)
> arra1 = array(c(v1, v2),dim = c(3,3,2))
> print(arra1)
, , 1

     [,1] [,2] [,3]
[1,]    1    7    6
[2,]    3    2    8
[3,]    5    4   10

, , 2

     [,1] [,2] [,3]
[1,]    1    7    6
[2,]    3    2    8
[3,]    5    4   10

>
```



```r
l = list(
  c(1, 2, 2, 5, 7, 12),
  month.abb,
  matrix(c(3, -8, 1, -3), nrow = 2),
  asin
)
print("Content of the list:")
print(l)
```

```
+   c(1, 2, 2, 5, 7, 12),
+   month.abb,
+   matrix(c(3, -8, 1, -3), nrow = 2),
+   asin
+ )
> print("Content of the list:")
[1] "Content of the list:"
> print(l)
[[1]]
[1]  1  2  2  5  7 12

[[2]]
 [1] "Jan" "Feb" "Mar" "Apr" "May" "Jun" "Jul" "Aug" "Sep" "Oct" "Nov"
[12] "Dec"

[[3]]
     [,1] [,2]
[1,]    3    1
[2,]   -8   -3

[[4]]
function (x)  .Primitive("asin")
```



```r
plot.new()
plot(1, type="n", xlab="", ylab="", xlim=c(0, 20), ylim=c(0, 20))
```

```
> plot.new()
> plot(1, type="n", xlab="", ylab="", xlim=c(0, 20), ylim=c(0, 20))
>
```

```
Data
  l          List of  4
  m          num [1:4, 1:3] 1 2 3 0 4 5 6 0 7 8 ...
  m1         int [1:5, 1:4] 1 2 3 4 5 6 7 8 9 10 ...
  m2         num [1:3, 1:3] 1 7 11 3 8 12 5 9 14
  m3         num [1:3, 1:3] 1 3 5 7 8 9 11 12 14
Values
```