Практическая работа № 2

Матрицы

Создание массивов и матриц данных

Понятия матрицы и массива данным в языке R, как и в других языках программирования, а также операции над ними не идентичны данным понятиям и действиям в математике.

В R матрица – это двухмерный массив однотипных данных – чисел, строк или логических значений. Массив данных – это структура однотипных данных с размерностью больше двух.

Подходы по созданию в R матриц и массивов данных на основе существующих векторов идентичны. Матрицы в R можно создавать разными способами. Выбор способа зависит от того, какую матрицу мы хотим создать: пустую матрицу (чтобы потом заполнять ее нужными значениями) или матрицу, составленную из уже имеющихся значений, например, из векторов.

```
 2  # Создание матриц и массивов
 3  z<-1:30
 4  dim(z)<-c(3,10)# преобразование вектора в 2-мерную матрицу
 5  z
 6  z[2,10]# вывод 10-го элемента 2-ой строки
 7
 8  z1<-1:60
 9  dim(z1)<-c(3,10,2)# преобразование вектора в 3-мерный массив
10  z1
11  z1[2,10,1]#вывод элемента массива данных
```

Рисунок 1 Пример создания матриц и массивов

Для того чтобы создать пустую матрицу, нужно использовать функцию matrix(). Размерность матрицы – число строк и число столбцов в ней.

matrix(data = NA, = 1, = 1, byrow = FALSE,
dimnames = NULL)

Параметр data может содержать значения создаваемой матрицы, заданные любым способом. По умолчанию (при значении NA) создается пустая матрица. Параметры nrow и ncol задают количество строк и столбцов в создаваемой матрице (соответственно).

Параметр byrow определяет порядок заполнения матрицы, при значении FALSE (по умолчанию) матрица заполняется по столбцам (сверху вниз), в противном случае – по строкам.

Параметр dimnames представляет собой Список, первый элемент которого задает имена строк, а второй – имена столбцов создаваемой матрицы. При значении NULL имена строк и столбцов нумеруются цифрами.

Аналогично, если список не полный, одно из измерений нумеруется цифрами.

Создадим матрицу 2×3, состоящую из нулей:

```
M <- matrix(0, nrow = 2, ncol = 3)
M
##      [,1] [,2] [,3]
## [1,]    0    0    0
## [2,]    0    0    0
```

Можем посмотреть на ее размерность:

```
dim(M)
## [1] 2 3
```

Рассмотрим, как собрать матрицу из «готовых» векторов. Пусть у нас есть три вектора

```
x <- c(1, 2, 3, 0)
y <- c(4, 5, 6, 0)
z <- c(7, 8, 9, 0)
```

и мы хотим объединить их в матрицу. Векторы будут столбцами матрицы:

```
m_cols <- cbind(x, y, z) # c – от columns
m_cols
##      x y z
## [1,] 1 4 7
## [2,] 2 5 8
## [3,] 3 6 9
## [4,] 0 0 0
```

А теперь векторы будут строками матрицы:

```
M_rows <- rbind(x, y, z) # r – от rows
M_rows
##   [,1] [,2] [,3] [,4]
## x    1    2    3    0
## y    4    5    6    0
## z    7    8    9    0
```

```
# Создание заполненной матрицы 3*2 с именованием строк
m1<-matrix(data=1:6,nrow = 3,ncol = 2,byrow = FALSE,
           dimnames = list(c("str1", "str2","str3")))
m1
# Создание заполненной матрицы 4*2 с именование строк и столбцов
m2<-matrix(data=1:8,nrow = 4,ncol = 2,byrow = TRUE,
           dimnames = list(c("str1", "str2","str3","str4"),c("col1","col2")))
m2
```

Рисунок 2 Создание матрицы с помощью функции matrix()

Еще одна функция – array() позволяет создавать как матрицы, так и массивы данных любой размерности:

array (data = NA, dim = length(dat a), dimnames = NULL)

Параметр dim определяет размерность создаваемого объекта.

Значение параметра dimnames аналогично, описанному для функции matrix()

Примеры использования функции array() для создания многомерных объектов с элементами одного и того же типа приведены на рисунке .

```
# Создание заполненной матрицы 2*3:
m3<-array(1:12,c(2,3))
m3
# Создание заполненного массива данных 3*4*2:
m4<-array(1:48,c(3,8,2))
m4
# Создание поименованного массива данных 3*4*2:
m5<-array(data=1:48,dim = c(3,6,2),dimnames = list(c("str1", "str2","str3"),
                          c("r1","r2","r3","r4","r5","r6"), c("z1","z2")))
m5
# Создание пустого массива данных 3*4*2:
m6<-array(dim=c(3,6,2))
m6
```

Рисунок 3 Создание объектов с помощью функции array()

Элементы матрицы

Для того чтобы обратиться к элементу матрицы, необходимо указать строку и столбец, на пересечении которых он находится:

m1[1, 3]

[1] 7

Если нам нужна отдельная строка (одна строка, все столбцы), то номер столбца нужно не указывать, просто оставить позицию пустой:

m1[1,] # вся первая строка

A B C D

1 4 7 0

Аналогично для столбцов:

m1[, 2] # весь второй столбец

r1 r2 r3

4 5 6

Операции над матрицами

Над матрицами и массивами данных с числовыми значениями можно выполнять те же операции, что и для векторов, причем с теми же уточнениями. Однако если планируется решение математических задач,

следует соблюдать требования, установленные в математике, в частности, по размерностям объектов и значениям их элементов.

Транспонирование матрицы

Транспонированной называется матрица $A^{T}_{(m\times n)}$, полученная из исходной матрицы $A_{(m\times n)}$.путем замены строк на столбцы, а столбцов на строки.

Для вычисления транспонированной матрицы используется функция t(), единственным аргументом которой является исходная матрица $A_{(m\times n)}$.

Пример использования функции t() приведен на рисунке 11.

```
# Транспонирование матрицы с помощью функции t()
a<-matrix(data=1:12,nrow = 3,ncol = 4,byrow = FALSE)
a
# Вычисление транспонированной матрицы
at<-t(a)
at
```

Рисунок 4 пример выполнения транспонирования матрицы

Обратная матрица

Обратной называется матрица $A^{-1}_{(m\times n)}$, при умножении на которую исходная квадратная матрица $A_{(m\times n)}$дает в результате единичную матрицу$E_{(n\times n)}$. Обратная матрица существует тогда и только тогда, когда определитель исходной матрицы не равен нулю

Для вычисления обратной матрицы применяют функцию solve() с единственным аргументом – исходной матрицей $A_{(n\times n)}$.

Пример вычисления обратной матрицы приведен на рисунке 12.

```
# Вычисление обратной матрицы с помощью функции solve()
a1<-matrix(data=c(1,6,2,7),nrow = 2,ncol = 2,byrow = FALSE)
a1
ao<-solve(a1)
ao
```

Рисунок 5 Пример вычисления обратной матрицы

Перемножение матриц

Для перемножения матриц по правилам математики используется операция %*%. Примеры перемножения матриц приведены на рисунке 13.

```
# Перемножение матриц с помощью операции %*%
a1<-matrix(data=c(1,6,2,7),nrow = 2,ncol = 2,byrow = FALSE)
a1
ao<-solve(a1)
ao
malt<-a1 %*% ao
malt
# Умножение матрицы на транспонированную матрицу
a<-matrix(data=1:12,nrow = 3,ncol = 4,byrow = FALSE)
a
# Вычисление транспонированной матрицы
at<-t(a)
at
rez<-a %*% at
rez
```

Рисунок 6 Пример выполнения операции перемножения матриц

Диагональная матрица

Диагональной называется квадратная матрица, все элементы которой, стоящие вне главной диагонали, равны нулю.

Для создания диагональной матрицы используется функция diag().

diag(x = 1, nrow, ncoi, names = TRUE)

Параметр x задает значения диагональных элементов (по умолчанию создается единичная диагональная матрица).

Параметры nrow и ncol определяют число строк и столбцов; значения данных параметров, как правило, должны быть равны, хотя это не обязательно.

Параметр names при значении TRUE предписывает, что в создаваемой диагональной матрице должны наследоваться имена исходной матрицы.

Пример создания диагональной матрицы приведен на рисунке 14.

```
# Создание диагональной матрицы
? diag
# 1 вариант
nrow<-5
ncol<-5
e_1<-diag(x=1,nrow,ncol,names = T)
e_1
# 2 вариант
# вывод диагональной матрицы
y=diag(1:4)
y
# 3 вариант
# количество строк не равно числу столбцов
z<-diag(nrow=4,ncol=5)
z
```

Рисунок 7 Создание диагональных матриц

Собственные векторы и собственные значения

Собственный вектор квадратной матрицы $A_{(n\times n)}$ представляет собой вектор X_n, умножение матрицы $A_{(n\times n)}$ на который дает коллинеарный вектор λX_n:

$$A, {}_{(n\times n)}\, X_{n=}\, \lambda\, X_n,$$

где λ – скалярная величина собственное значение вектора X_n относительно матрицы $A_{(n\times n)}$.

Для вычисления собственных значений и собственных векторов числовых (двойных, целочисленных, логических) или комплексных матриц используется функция eigen();

eigen(x, symmetric, only.values = FALSE, EISPACK = FALSE)

Раскроем назначение основных параметров функции.

Параметр x представляет собой матрицу с целочисленными, вещественными или комплексными значениями элементов, спектральное разложение которой должно быть вычислено. Логические матрицы приводятся к числовым (TRUE =1, FALSE=0).

Если параметр only.values равен TRUE, вычисляются и возвращаются только собственные значения, иначе возвращаются как собственные значения, так и собственные вектора.

```
2  # Вычисление собственных значений и собственных векторов
3  y<-diag(c(2,4,1,6))
4  y
5  e<-eigen(y)
6  e$values # вывод собственных значений
7  e$vectors # вывод собственных векторов
```

Рисунок 8 Вычисление собственных значений и собственных векторов матрицы

Приведем примеры обращений к элементам матриц и массивов данных.

По ходу изложения материала варианты обращений к элементам матриц и массивов данных приводились неоднократно. Еще некоторые возможности представлены на рисунке 17.

```
 9  # Обращение к элементам матриц и массивов данных
10  w<-array(1:24,dim = c(3,3,2))
11  w
12  w[1,,2]
13  m<-matrix(data = c(1:4,5,6,12,9,4),nrow = 3,ncol = 3)
14  m
15  m[1,]
16  m[,1]
17  m[1:2,1:2]
```

Рисунок 9 Примеры обращения к элементам матриц и массивов

Метаданные матриц

В R любая переменная является объектом какого-либо класса, например стандартные числа – это векторы длины 1, матрицы – это векторы векторов и т.д. Для матриц, как и для векторов, определены специальные функции, позволяющие работать со свойствами матриц как со свойствами объектов класса.

Имена строк и столбцов. Допустим, вы создали матрицу по специальной формуле или заданию и вам необходимо обозначить в данной матрице названия строк и столбцов. Для решения данной задачи используются функции просмотра свойств матрицы rownames() и colnames().

Данные функции при вызове их относительно какой-то заданной матрицы будут возвращать их действительные имена строк и столбцов, а также если при вызове функции, присваивать данному вызову вектора строк той же длины, что и данная размерность, то они присваиваются на соответствующие места в качестве имён строк или столбцов. Пример использования данных функций:

```
# инициализируем матрицу
matrix <- diag(1:4)
print(matrix)

# попробуем посмотреть на имена строк и столбцов
cat("\nСтроки:\n")
print(rownames(matrix))
cat("\nСтолбцы:\n")
print(colnames(matrix))

# присвоим имена
cat("\nПрисвоение названий\n")
rownames(matrix) <- paste0("row", 1:4)
colnames(matrix) <- paste0("col", 1:4)
cat("Присвоение завершено\n")

# попробуем посмотреть на имена строк и столбцов
cat("\nСтроки:\n")
print(rownames(matrix))
cat("\nСтолбцы:\n")
print(colnames(matrix))
```

Результат выполнения скрипта:

```
     [,1] [,2] [,3] [,4]
[1,]    1    0    0    0
```

```
[2,]    0    2    0    0
[3,]    0    0    3    0
[4,]    0    0    0    4

Строки:
NULL

Столбцы:
NULL

Присвоение названий
Присвоение завершено

Строки:
[1] "row1" "row2" "row3" "row4"

Столбцы:
[1] "col1" "col2" "col3" "col4"
```

В данном примере показано, как применение свойств переменных в разрезе изменения их метаданных или полей объектов, так и продемонстрирована векторизованная функция paste() и её применение в разрезе уменьшения размера кода.

Вычисления по матрице. Для специальных быстрых вычислений по матрице в R существуют функции «обхода» – векторизованные функции быстро вычисляющие агрегацию по столбцам или строкам. Самым часто используемым в таких вычислениях являются функции rowSums() и colSums():

```
> # Задание матрицы
> matrix <- cbind(1:4, 9:6, c(1.8, 9.1, -2.3, 3.4))

> print(matrix)
     [,1] [,2] [,3]
[1,]    1    9  1.8
[2,]    2    8  9.1
[3,]    3    7 -2.3
[4,]    4    6  3.4

> # Агрегации
> print(rowSums(matrix))     # Сумма элементов в строках
[1] 11.8 19.1  7.7 13.4


> print(colSums(matrix))     # Сумма элементов в столбцах
```

[1] 10 30 12

Помимо классических агрегаций по размерностям матрицы кроме суммы можно задать собственную агрегацию путём использования функции apply() – функция быстрого вычисления функции по матрице. Функция состоит в применении указанной функции, или созданной на ходу, к указанной размерности матрицы:

```
> print(apply(X = matrix, MARGIN = 1, FUN = sum))
[1] 11.8 19.1  7.7 13.4

> print(apply(matrix, 2, sum))
[1] 10 30 12

> # Другие агрегации
> print(apply(matrix, 1, prod))
[1]  16.2 145.6 -48.3  81.6

> print(apply(matrix, 2, mean))
[1] 2.5 7.5 3.0

> print(apply(matrix, 2, max))
[1] 4.0 9.0 9.1
```

Из примеров наглядно видно, что MARGIN отвечает за размерность к которой применяется функция, в FUN указывается имя применяемой функции или сама функция, X – это собственно матрица.

Самостоятельная работа №2

Часть 1

1. Создайте матрицу размерности 3 * 4, состоящую из 3, а затем измените некоторые ее элементы так, чтобы получить следующее:
2. [3 3 4 3]
3. [1 3 3 3]

[3 NA 3 1]

4. Создайте из следующих векторов матрицу, такую, что:

- векторы являются столбцами матрицы
- векторы являются строками матрицы

```
a <- c(1, 3, 4, 9, NA)
b <- c(5, 6, 7, 0, 2)
c <- c(9, 10, 13, 1, 20)
```

Дайте (новые) названия строкам и столбцам матрицы.

3. Может ли матрица состоять из элементов разных типов? Проверьте: составьте матрицу из следующих векторов (по столбцам):

```
names <- c("Jane", "Michael", "Mary", "George")
ages <- c(8, 6, 28, 45)
gender <- c(0, 1, 0, 1)
```

Если получилось не то, что хотелось, подумайте, как это можно исправить, не теряя информации, которая сохранена в векторах.

Добавьте в матрицу столбец age_sq – возраст в квадрате.

4. Создайте из векторов из задачи 3 список (list) и назовите его info.

- Выведите на экран имя Michael (обращаясь к элементам списка, конечно).
- Выведите на экран вектор gender.
- Назовите векторы в списке name, age, gender. Выведите на экран элементы вектора name.
- Добавьте в список вектор drinks, в котором сохранены значения: juice, tea, rum, coffee.
- Добавьте в список данные по еще одному человеку: John, 2 года, мужской пол, любит молоко.

5. В R есть функция strsplit(), которая позволяет разбивать строки (текстовые переменные) на части по определенным символам.

Пусть у нас есть строка s:

```
s <- "a,b,c,d"
```

Мы хотим получить из нее вектор из 6 букв. Применям функцию:

```
let <- strsplit(s, ",")
```

Получили почти то, что хотели. Почему почти? Потому что получили не вектор, а список!

```
class(let)
## [1] "list"
```

Превратим в вектор:

```
unlist(let)
## [1] "a" "b" "c" "d"
```

Теперь все в порядке, получили вектор из четырех элементов.

Дана строка index:

```
index <- "0,72;0,38;0,99;0,81;0,15;0,22;0,16;0,4;0,24"
```

Получите из этой строки числовой вектор I.

Часть 2

Итерационный метод

1. Создать диагональную матрицу A размерности 2x2, состоящую из элементов 4 и 9. Задать для данной матрицы названия строк как «eq1» и «eq2», а для столбцов «x1» и «x2».

2. Найти собственные значения матрицы A, вывести эти значения на экран. Объяснить, почему собственные значения матрицы A получились именно такими.

3. Найти матрицу B с помощью соотношения $B = I - A$, где I —единичная матрица. Вывести данную матрицу на экран.

4. Задать векторы u и f, равными:

$$f = (4 \quad 2), u = (0.2 \quad -0.3)$$

5. Для заданных A, u, f решить СЛАУ вида:

$$A * u = f$$

при помощи стандартного вида решения СЛАУ для невырожденной матрицы A с помощью стандартных функций R. Вывести итоговый вектор решения СЛАУ u_result на экран (вам может понадобиться функция solve()).

6. Произвести 7 итераций по следующей схеме:

$$u_{i+1} = Bu_i + f,$$

где результат каждой итерации будет записан в отдельную переменную и храниться в памяти сессии.

7. Сравните результаты u7 и u_result. Насколько векторы различны по каждой из координат?

8. Для матрицы A и вектора f произвести деление всех их элементов на максимальное значение элементов матрицы A.

9. Повторить пункты 2-3 и далее 5-7 для полученных в результате пункта 8 матрицы и вектора свободных членов.

10. Сравнить результаты, полученные в пункте 7 и в пункте 9.

Часть 3

Срезы матрицы

Введем матрицу с помощью следующего скрипта:

```
step <- 1              # Шаг сетки
dekart_begin <- -5     # Начало сетки
dekart_end <- 5        # Конец сетки

# Задание сеточной поверхности
x <- seq(from = dekart_begin, to = dekart_end, by = step)
```

```
y <- x

# Задание двумерной функции на координатной сетке
surface_matrix <- outer(X = x,
                        Y = y,
                        FUN = function(x,y) Re(exp(-1i * 0.5  * x * y)))
dimnames(surface_matrix) <- list(x, y)
```

Задание 1.

1. Вывести в файл “summary.txt” следующую информацию о созданной матрице с соответствующими подписями к ней:

- количество элементов матрицы (“number of matrix elements:”)
- размерность строк (“number of rows:”)
- размерность столбцов (“number of cols:”)
- сумма элементов главной диагонали (“sum of main diag elements:”)
- сумма элементов серединного среза матрицы по строкам (“sum of middle row elements: ”)
- сумма элементов серединного среза матрицы по столбцам (“sum of middle column elements:”)
- суммы строк матрицы (“row sums:”)
- суммы столбцов матрицы (“col sums:”)

2. Переписать скрипт пункта 1 так, чтобы данные о начале, конце и шаге координатной сетки вводились пользователем с помощью консольного ввода (берём в рассмотрение только квадратную сетку). Произвести вывод в файл “summary2.txt” той же информации о полученной матрице, как и в пункте 1, за исключением суммы элементов серединного столбца и строки (из-за неопределённости в четности или нечетности измерений матрицы)

3. Переписать скрипт пункта 2 так, чтобы данные о начале, конце и шаге координатной сетки вводились пользователем с помощью текстового файла “inputs.txt”. Выбор формата считывания данных остаётся за программистом. В данном задании необходимо переписать скрипт так, чтобы была возможность задавать неквадратную и неравномерную сетку для функции (6 параметров вместо 3). Произвести вывод в файл “summary3.txt” той же информации о полученной матрице, как и в пункте 2.

Часть 4

Машины

Для данного задания в матрицу собраны данные cars о максимальной скорости и тормозной дистанции машин 1920-х годов. На основе данной матрицы данных нам необходимо будет на практике научиться приложениям матричных вычислений в программировании на R.

Если производить специальную визуализацию предоставленных нам данных, то график разброса показаний дистанции торможения от максимальной скорости авто в милях в час выглядит следующим образом (рис.18):

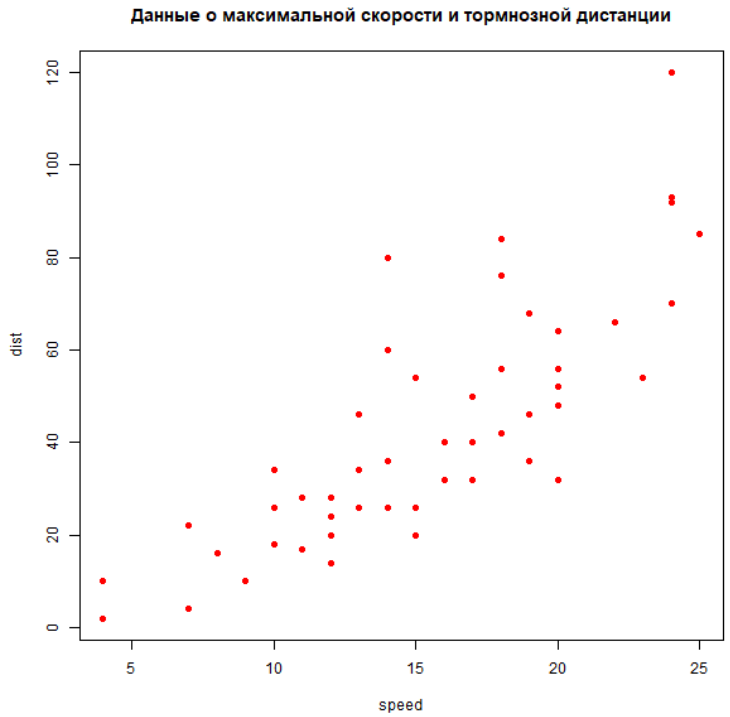

Рисунок 10. Максимальная скорость (x), тормозная дистанция (y) для машин 1920-х годов

Матричные вычисления в данной работе помогут нам построить модель линейной аппроксимации данных при помощи метода наименьших квадратов и нормального уравнения.

Данные для задания формируются в матрицу с помощью следующей команды:

```
cars_matrix <- as.matrix(cars)
```

Задание 1.

1. Создать новую матрицу cars_speed, состоящую из 2 столбцов: единичного вектора и первого столбца матрицы cars_matrix.

2. Создать новый вектор cars_dist, который является срезом матрицы cars_matrix по второму столбцу.

3. Рассчитать новый вектор alpha по следующему соотношению, называемому *нормальным уравнением модели регрессии*:

$$\vec{\alpha} = (A^T * A)^{-1} * A^T * y,$$

где A – матрица cars_speed, y – вектор cars_dist, умножение между членами подразумевается под матричным.

4. Проверить тип данных вычисленного вектора alpha, произвести преобразование данной переменной к структуре данных “vector”, в случае если получилось иное, без потери, содержащейся в нём информации.

5. Создать переменные alpha_c и alpha_x на основе первого и второго элемента вектора alpha соответственно. Вывести на экран данные вектора по следующему шаблону:

“alpha_c = …” – первый элемент вектора

“alpha_x = …” – второй элемент вектора

6. Создать новый вектор cars_speed_lm на основе матрицы cars_matrix с помощью взятия его первого столбца.

7. Создать новый вектор cars_dist_lm на основе следующего соотношения:

$$\text{cars\_dist\_lm} = \text{alpha\_c} + \text{cars\_speed\_lm} * \text{alpha\_x}$$

8. В вектор dist_residuals сохранить информацию о разности между векторами cars_dist_lm и cars_dist.

9. Вычислить среднее и стандартное отклонение вектора dist_residuals.

10. Вывести на экран значения вектора cars_dist_lm, убедиться в их отсортированности по возрастанию (потому что это прямая линия).

11. Вывести значения среднего и стандартного отклонений dist_residuals на экран.

Результат работы с матрицами:

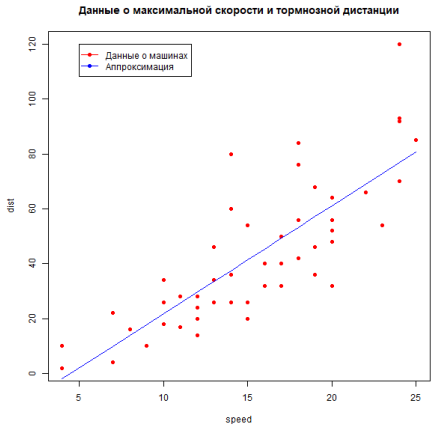

Рисунок 11. Аппроксимирующая прямая для зависимости дистанции торможения от тормозного пути автомобиля

Контрольные вопросы

1. Приведите способы создания матриц
2. Чем массивы данных отличаются от матриц
3. Приведите примеры операций над матрицами и массивами
4. Сравните возможности выполнения операций над матрицами и массивами в математике и в R