Лабораторная работа №4

ВЕКТОРНОЕ ПРОГРАММИРОВАНИЕ

Цель: познакомиться с особенностями векторного программирования в R. Решить задания в соответствующем стиле программирования. Составить отчет.

Теоретические сведения

R – язык программирования для научных вычислений и анализа данных с упором на визуализацию и воспроизводимость;

R – свободное кроссплатформенное программное обеспечение с открытым исходным кодом;

R – интерпретируемый язык с интерфейсом командной строки;

R – мультипарадигмальный, векторный язык, сочетающий в себе:

функциональное программирование;

процедурное программирование;

объектно-ориентированное программирование;

рефлексивное программирование.

Векторизация – поэлементное одновременное выполнение действий над всеми элементами.

Основные объекты языка:

Вектор – основной объект языка R. Вектор может содержать более одного значения, все объекты в векторе имеют одну природу.

Лист – элемент языка R который может содержать разные по размеру и типу данных векторы.

Дата фрейм – элемент языка R который может содержать одинаковые по размеру, но разные по типу данных векторы. Дата фрейм является разновидностью листа.

Примеры синтаксиса языка R

```
my_var1 <- 42
my_var2 <- 35.25
my_var1 + 100
my_var1 + my_var2 - 12
my_var3 <- my_var1^2 + my_var2^2
my_var3
```

Рисунок 1 – Создание объектов и работа с ними (выполните код)

```
my_var3 > 200
my_var3 > 5000
my_var1 == my_var2
a <- 34
a = 2
my_var1 != my_var2
my_new_var <- my_var1 == my_var2
my_new_var
```

Рисунок 2 – Получение логических переменных (выполните код)

```
1:67
my_vector1 <- 1:67
my_vector2 <- c(-32, 45, 67, 12.78, 129, 0, -65)
my_vector1[1]
my_vector1[3]
my_vector2[2]
my_vector2[c(1, 2, 3)]
my_vector2[1:3]
my_vector2[c(1, 5, 6, 7, 10)]
```

Рисунок 3 – Создание векторов и обращение к элементам вектора (выполните код)

```
my_vector1 + 10
my_vector2 + 56
my_vector2 == 0
my_vector1 > 30
x <- 23
my_vector1 > 23
my_vector1 > x
x == 23
```

Рисунок 4 – Арифметические и логические операции (выполните код)

```
my_vector2 > 0
my_vector2[my_vector2 > 0]
my_vector2[my_vector2 < 0]
my_vector2[my_vector2 == 0]
my_vector1[my_vector1 > 20 & my_vector1 < 30]
my_numbers <- my_vector1[my_vector1 > 20 & my_vector1 < 30]
positeve_numbers <- my_vector2[my_vector2 > 0]
```

Рисунок 5 – Логические выборки (выполните код)

```
age <- c(16, 18, 22, 27)
is_married <- c(F, F, T, T)
data <- list(age, is_married)
data
data[[1]][1]
data[[2]][3]
name <- c('Olga', 'Maria', 'Nastya', 'Polina')
data <- list(age, is_married, name)
data
```

Рисунок 6 – Создание листа (выполните код)

```
df <- data.frame(Name = name, Age = age, Status = is_married)
df
```

Рисунок 7 – Создание дата фрейма (выполните код)

Создание собственных функций в R:

Для создания собственных функций в R используется следующая конструкций:

```
function_name <- function(argument_1, argument_2){
  function_body
  ...
  return(value)
}
```

Рисунок 8 – Конструкция создания функций в R

```
area_circle <- function(r){
  area <- r^2*pi
  return(area)
}
# Вызов функции
area_circle(5)
```

Рисунок 9 – Пример создания и вызова функции «Площадь круга»

Некоторые объекты языка R:

Матрица (matrix) представляет собой двумерную совокупность числовых, логических или текстовых величин.

Массив (array) – это совокупность некоторых однотипных элементов, обладающая размерностью больше двух.

Векторизованные функции:

Характерной особенностью R является векторизация вычислений. Векторизация представляет собой один из способов выполнения параллельных вычислений, при котором

программа определенным образом модифицируется для выполнения нескольких однотипных операций одновременно.

Принцип векторизованных вычислений применим не только к векторам как таковым, но и к более сложным объектам R - матрицам, спискам и таблицам данных (для R разницы между последними двумя типами объектов не существует – фактически таблица данных является списком из нескольких компонентов-векторов одинакового размера). В базовой комплектации R имеется целое семейство функций, предназначенных для организации векторизованных вычислений над такими объектами. В названии всех этих функций имеется слово apply (англ. применить), которому предшествует буква, указывающая на принцип работы той или иной функции (см. подробнее в справочном файле - ?apply).

Функция apply() - используется в случаях, когда необходимо применить какую-либо функцию ко всем строкам или столбцам матрицы (или массивов большей размерности):

```
# Создадим обычную двумерную матрицу
M <- matrix(seq(1, 16), 4, 4)
# Найдем минимальные значения в каждой строке матрицы
apply(M, 1, min)
# Найдем минимальные значения в каждом столбце матрицы
apply(M, 2, max)
```

Рисунок 10 – Пример работы с функцией apply

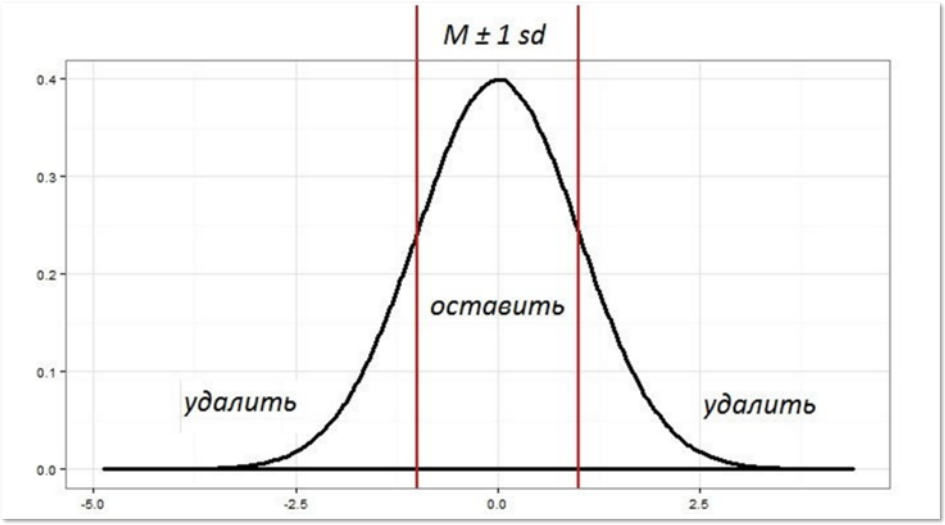
Практическая часть

Задание 1 – Предобработка данных. Создайте новый вектор my_vector, следующей строчкой:

```
my_vector <- c(21, 18, 21, 19, 25, 20, 17, 17, 18, 22, 17, 18, 18, 19, 19, 27, 21, 20,
24, 17, 15, 24, 24, 29, 19, 14, 21, 17, 19, 18, 18, 20, 21, 21, 19, 17, 21, 13, 17, 13,
23, 15, 23, 24, 16, 17, 25, 24, 22)
```

В векторе my_vector отберите только те наблюдения, которые отклоняются от среднего меньше, чем на одно стандартное отклонение. Сохраните эти наблюдения в новую переменную my_vector2. При этом исходный вектор оставьте без изменений.

Полезные функции: mean(x) – среднее значение вектора x; sd(x) – стандартное отклонение вектора x; abs(x) – абсолютное значение числа n



Задание 2. Напишите функцию get_negative_values, которая получает на вход dataframe произвольного размера. Функция должна для каждой переменной в данных проверять, есть ли в ней отрицательные значения. Если в переменной отрицательных значений нет, то эта

переменная нас не интересует, для всех переменных, в которых есть отрицательные значения мы сохраним их в виде списка или матрицы, если число элементов будет одинаковым в каждой переменной (смотри пример работы функции).

```
test_data <- as.data.frame(list(V1 = c(-9.7, -10, -10.5, -7.8, -8.9), V2 = c(NA, -10.2,
-10.1, -9.3, -12.2), V3 = c(NA, NA, -9.3, -10.9, -9.8)))
get_negative_values(test_data)

$V1
[1]  -9.7 -10.0 -10.5  -7.8  -8.9
$V2
[1] -10.2 -10.1  -9.3 -12.2
$V3
[1]  -9.3 -10.9  -9.8

test_data <- as.data.frame(list(V1 = c(NA, 0.5, 0.7, 8), V2 = c(-0.3, NA, 2, 1.2), V3 =
c(2, -1, -5, -1.2)))
get_negative_values(test_data)

$V2
[1] -0.3
$V3
[1] -1.0 -5.0 -1.2

test_data <- as.data.frame(list(V1 = c(NA, -0.5, -0.7, -8), V2 = c(-0.3, NA, -2, -1.2),
V3 = c(1, 2, 3, NA)))
get_negative_values(test_data)

      V1   V2
[1,] -0.5 -0.3
[2,] -0.7 -2.0
[3,] -8.0 -1.2
```

Вопросы для контроля из материалов лабораторного занятия

1. Векторизация;
2. Основные объекты языка R.
3. Создание собственных функций
4. Векторизованные функции семейства apply.

Вопросы для поиска и письменного ответа

1. Особенности языка программирования R;
2. Языки, поддерживающие парадигму векторизации;
3. CRAN;
4. Плюсы и минусы языка R.