

**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ФАКУЛЬТЕТ РАДИОФИЗИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ  
Кафедра системного анализа и компьютерного моделирования**

**Н. Н. Яцков, А. А. Юдин**

# **ВВЕДЕНИЕ В БИОИНФОРМАТИКУ**

**Методические указания  
к лабораторным работам**

**Для студентов специальности  
1-31 03 07-02 «Прикладная информатика»**

**МИНСК  
2018**

УДК 575.112:004 (075.8)

ББК 28.04я73+32.97я73

Я93

Рекомендовано советом  
факультета радиофизики и компьютерных технологий  
26 декабря 2017 г., протокол № 5

Рецензент  
кандидат технических наук, доцент *В. И. Микулович*

**Яцков, Н. Н.**

Я93 Введение в биоинформатику : метод. указания к лаб. работам /  
Н. Н. Яцков, А. А. Юдин. – Минск : БГУ, 2018. – 51 с.

Методические указания предназначены для проведения лабораторного практикума по курсу «Введение в биоинформатику» для студентов факультета радиофизики и компьютерных технологий. Содержит учебный материал по методам обработки и анализа многомерных наборов биоданных.

**УДК 575.112:004 (075.8)**

**ББК 28.04я73+32.97я73**

© Яцков Н. Н., Юдин А. А., 2018

© БГУ, 2018

## **ВВЕДЕНИЕ**

Настоящее издание представляет собой руководство к лабораторному практикуму по курсу «Введение в биоинформатику». Лабораторные работы выполняются в интегрированной вычислительной среде статистического программирования R. Цель настоящего издания – научить студента применять алгоритмы биоинформатики и интеллектуального анализа данных для решения ряда прикладных задач обработки и анализа многомерных наборов биоданных на примере использования среды R. Тематика практикума охватывает алгоритмы предварительного анализа больших наборов данных, метода главных компонент, кластерного анализа, нейронных сетей, метода опорных векторов и глобального выравнивания пар нуклеотидных последовательностей. Приведенные справочные теоретические сведения содержат достаточный объем информации, исключающий необходимость прибегать другим источникам литературы в ходе выполнения практических работ.

Прилагается компакт-диск, содержащий файлы заданий, данных и приложений к лабораторному практикуму.

# ЛАБОРАТОРНАЯ РАБОТА № 1

## ОСНОВЫ РАБОТЫ В R

**Цель работы.** Изучение элементов языка, основных функций и среды R. Решение систем линейных уравнений. Создание пользовательских функций и построение их графиков в среде R.

### КРАТКИЕ СВЕДЕНИЯ

Система R предназначена для выполнения статистических расчетов на компьютере. С ее помощью эффективно решаются задачи математической статистики, математического моделирования и биоинформатики.

R – это одновременно и операционная среда и язык программирования. Язык R прост и легко поддается изучению. Из командной строки можно вызвать отдельную команду или программу (последовательность команд), оформленную в виде r-файла. Для удобства работы со средой R рекомендуется загрузить и установить RStudio.

#### 1. Загрузка системы RStudio и работа в главном окне

Загрузить RStudio: кнопка **Пуск / Программы / RStudio**. Создать на доступном диске собственную папку: *имя папки набирайте латинскими буквами*. Текущей папкой среды RStudio назначьте свою созданную папку с использованием команды `setwd("C:/...")` (рис. 1.1).



Рис. 1.1. Окно текущей папки среды RStudio

Вычисления можно производить прямо в командном окне. Для этого в главном окне после символа `>` наберите какие-либо арифметические действия.

При выполнении большого количества вычислений (например, при реализации некоторого алгоритма) работать в подобном режиме не всегда удобно. В таких случаях следует оформить вычисления в виде r-файлов.

**Задание 1.** Выполните произвольные вычисления в командном окне, попробуйте использовать различные арифметические операции и задайте с помощью круглых скобок приоритет их выполнения.

## 2. Создание г-файла

г-файлы, как правило, содержат последовательность команд. Фактически в г-файле содержится код пользовательской программы, который удобно редактировать и дополнять.

Для создания г-файла следует зайти в меню **File\ New File\ R Script**. В результате открывается текстовый редактор RStudio, предназначенный для создания и редактирования подобных файлов.

Теперь следует сохранить г-файл: в меню редактора **File\ Save as...\**, в диалоговом окне задать **имя файла\ ОК**.

В созданном г-файле можно запрограммировать любые арифметические вычисления. Они будут сохранены в этом файле, и их можно выполнить на другом компьютере, на котором установлен R или RStudio.

**ВНИМАНИЕ!** Имена объектов должны начинаться с буквы или точки (“.”) и состоять из букв, цифр, знаков точки и подчеркивания. R чувствителен к регистру!

**Задание 2.** Создайте г-файл с произвольным именем. Сохраните его в своей рабочей папке.

## 3. Использование справочной системы

R содержит большое число встроенных математических функций. Для того, чтобы пользоваться встроенной функцией необходимо знать, какие параметры и в какой последовательности следует в неё передавать. Справку о встроенной функции можно получить набрав в командном окне команду *help* и далее имя используемой функции в качестве аргумента. Например, *sin: help(sin)* или *?sin*. Вызов расширенной справки производится из главного меню RStudio: **Help\ R Help** (или нажать клавишу **F1**). Поиск нужной функции осуществляется в строке поиска по некоторому ключевому слову.

**Задание 3.** Получите справку по следующим операторам и функциям: *t*, *asin*, *plot*.

## 4. Вспомогательные функции

**Очистка экрана с помощью клавиш <CTRL>+<L>.** С помощью сочетания клавиш <CTRL>+<L> можно очистить видимое содержимое командного окна системы R. Для этого в главном окне необходимо просто ввести сочетание клавиш <CTRL>+<L> – экран очистится от всех введенных записей.

**Очистка рабочего пространства.** Рабочее пространство системы R (Workspace / Global Environment) – специально зарезервированная область оперативной памяти, в которой хранятся значения переменных, вычисляемых в рамках текущего сеанса.

С помощью команды `rm(list=ls( ))` произведите очистку рабочего пространства системы R – сотрите из оперативной памяти все вычисленные значения переменных.

**Задание 4.** Произведите очистку экрана и рабочего пространства.

## 5. Работа с векторами и матрицами

**Объявление векторов и матриц.** Объявление векторов и матриц производится следующим образом:

```
V <- c(1,2,3,4,5)           # Создание вектора
M <- matrix(1:15,nrow=3,ncol=5) # Создание матрицы
```

**Доступ к отдельным элементам вектора и матрицы.** Доступ к отдельному элементу вектора: `V[3]` – третий элемент вектора **V**.

Доступ к отдельному элементу матрицы: `M[2,3]` – элемент матрицы из второй строки и третьего столбца матрицы **M**.

Доступ ко всему третьему столбцу матрицы **M**: `M[,3]`.

Доступ ко всей второй строке матрицы **M**: `M[2,]`.

**Добавление и удаление столбцов матрицы.** Добавление строки к матрице:

```
M <- matrix(1:15,nrow=3,ncol=5)
S <- 1:5
M <- rbind(M,S)
```

Добавление столбца к матрице:

```
M <- matrix(1:15,nrow=3,ncol=5)
S <- 1:3
M <- cbind(M,S)
```

Удаление всего третьего столбца матрицы **M**: `M <- M[, -3]`.

Удаление всей второй строки матрицы **M**: `M <- M[-2, ]`.

## 6. Нахождение максимального и минимального элементов матрицы

Пусть дана матрица **C**. С помощью функций `min`, `max` и `apply` можно находить минимальные и максимальные элементы по строкам, по столбцам и по всей матрице:

`apply(C,2,min)` – вектор минимальных элементов, найденных по столбцам,

`apply(C,1,min)` – вектор минимальных элементов, найденных по строкам (аналогично для функции *max*).

Для поиска минимального и максимального элемента по всей матрице применяются функции *min* и *max* соответственно.

**Задание 5.** Введите в **test\_matrix.r** описанные выше процедуры объявления вектора и матрицы, процедуры доступа к элементам вектора и матрицы, поиск минимального и максимального элемента в матрице. Просмотрите результат в командном окне.

Добавьте в **test\_matrix.r** код, добавляющий и удаляющий вектор к матрице.

**Задание 6.** Объявите матрицу **C** размерности 5 строк на 5 столбцов с произвольными элементами. С помощью функций *min*, *max* найдите минимальные и максимальные элементы по строкам, по столбцам и по всей матрице.

Добавьте соответствующий код в файл **test\_matrix.r**.

## 7. Многомерные массивы

Многомерные массивы – массивы с размерностью больше двух (рис. 1.2). Для доступа к элементу такого массива необходимо задавать три и более индекса.

Для объявления, например, трехмерного массива состоящего из нулей следует вызвать функцию *array()*:

```
A <- array(0,dim = c(2,3,4))
```

В результате получится трехмерный массив, состоящий из двумерных матриц.

Доступ к элементу осуществляется путем указания индексов для трехмерного массива: `A[номер строки, номер столбца, номер матрицы]`: `A[1,2,1]`.

**Размер массива.** Размер вектора, матрицы или многомерного массива можно узнать, вызвав функцию *dim()*:

```
dim(A)
```

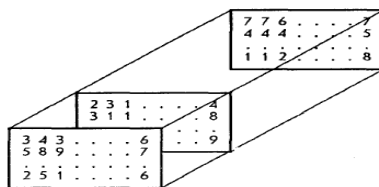


Рис. 1.2. Схематическое представление трехмерного массива

Транспонирование матрицы. Транспонирование матрицы осуществляется помощью функции *t*:

```
b <- matrix(1:4,2,2)
s <- t(b)
```

## 8. Арифметические операции с векторами и матрицами

По умолчанию все действия (+, −, \*, /, ^ – возведение в степень) выполняются над элементами векторов или матриц

```
a <- 2:5
b <- 1:4
c <- a + b
```

Для поэлементного перемножения двух векторов или матриц

```
d <- a * b
```

В результате перемножения вектора на вектор по правилам линейной алгебры получаем число (вектор *b* при этом надо транспонировать):

```
a <- 2:5
b <- 1:4
c <- a %*% b
```

## 9. Массивы строковых переменных

Создание строковой переменной производится с использованием одинарных или двойных кавычек

```
Distance <- 'euclidean'
```

Создание вектора строковых переменных производится с помощью функции *c*:

```
Distances <- c('euclidean','cityblock','minkowski')
```

**Задание 7.** Получить решение *x* заданной системы линейных уравнений вида  $\mathbf{A} \cdot \mathbf{x} = \mathbf{B}$ , где *A* – квадратная матрица  $n \times n$ , *B* – вектор размера *n*. Решение данной системы уравнений можно получить, набрав строку кода

```
x <- solve(A,B)
```

Функция *solve()* вычисляет решение *x* системы линейных уравнений вида  $\mathbf{A} \cdot \mathbf{x} = \mathbf{B}$ .

Решите уравнение  $\mathbf{A} \cdot \mathbf{x} = \mathbf{B}$ ,  $\mathbf{A} = \begin{bmatrix} 2 & 1 \\ 3 & 4 \end{bmatrix}$  и  $\mathbf{B} = \begin{bmatrix} 4 \\ 11 \end{bmatrix}$ .

Проверьте найденное решение, подставив его в исходное уравнение.

Добавьте в **test\_matrix.m** соответствующий код.

Запустите программу на выполнение (клавиши <Ctrl>+<Shft>+<S>).



## 10. Работа с графикой

Пример программы, реализующей построение графика функции *sin*.

```
x <- seq(0,6.28,0.01) # Создание вектора аргумента 0<x<6.28 с
шагом 0.1
y <- sin(x)           # Вычисление значений функции sin в заданных точках
plot(x,y,xlab = 'Argument x',ylab = 'Function y') # Построение
графика функции
grid() # Отображение сетки на рисунке
title(main='Function sin(x)')# Заголовок графика
```

## 11. Сохранение фигуры с графиком в заданном графическом формате (tiff)

Для сохранения текущего окна с графиком в заданном графическом формате необходимо, не закрывая это окно, набрать в командном окне следующий код (его можно также вставлять в код г-файла)

```
tiff(filename = "plot.tiff",res = 100)
plot(x,y)
dev.off()
```

где *tiff* указывает формат файла (*tiff*), *res = 100* – разрешение файла, *filename* – имя файла с рисунком.

**Задание 8.** Создайте файл **test\_graphic.r**, наберите приведенный код п. 10. Сохраните полученный график в формате *tiff* с разрешением *res = 100* (имя файла произвольное).

## 12. Построение трехмерных графиков

Рассмотрим построение трехмерных графиков функций на примере функции Розенброка. В отдельном г-файле введите код:

```
Lx <- -5 # Левая граница для x
Rx <- 5  # Правая граница для x
stepx <- 0.05 # Шаг по оси x

Ly <- -5 # Левая граница для y
Ry <- 5  # Правая граница для y
stepy <- 0.05 # Шаг по оси y

# Создание сетки координат
xs <- seq(Lx,Rx,stepx)
ys <- seq(Ly,Ry,stepy)

z <- outer(xs, ys, function(x, y) 100*(y - x^2)^2 + (1-x)^2)
persp(xs, ys, z, phi = 30, theta = -30,col = "lightblue")
```

**Задание 9.** Создайте файл **test\_3Dgraphic.r**, в котором наберите приведенный выше код п. 12.

### 13. Создание пользовательских функций

Рассмотрим задачу реализации функции, возвращающей значения двумерной случайной величины  $\xi=(X,Y)$ , равномерно распределённой на плоскости с заданными границами (границы области изменения должны задаваться в программе).

Требуется построить:

- а) график, отражающий значения реализации случайной величины  $\xi$  на плоскости в виде точек;
- б) гистограммы распределения компонент двумерной случайной величины.

**Создание функции.** Введите в новый г-файл следующий код функции, а затем сохраните файл:

```
my_func <- function(left, right, N){  
  # Генерация N значений случайной величины X в области задания X  
  runif(n = N, min = left, max = right)  
}
```

Функция *runif()* – генератор значений случайной величины, равномерно распределённой на интервале  $[0;1]$ . Вызов *runif()* – возвращает вектор из  $N$  элементов. Результат функции *my\_func* возвращается в векторах  $X$  и  $Y$ .

**Задание 10.** Создайте новый г-файл и сохраните его под именем **my\_func**. В файле поместите размещенное выше описание функции **my\_func**.

**Вызов собственной функции.** Создайте файл **test\_my\_func.r**, в котором введите следующий код:

```
# Задание диапазона изменения X  
X_left <- -2  
X_right <- 2  
  
# Задание диапазона изменения Y  
Y_left <- -3  
Y_right <- 3  
  
N <- 1000 # Задание количества сгенерированных точек  
source("myfunc.R") # Вызов функции  
  
X <- my_func(X_left, X_right, N)  
Y <- my_func(Y_left, Y_right, N)
```

```
plot(X,Y) # Построение графика функции
```

**Построение гистограммы с помощью *hist*.** В файле **test\_my\_func.r** добавьте следующий код:

```
# Инициализация параметров для построения гистограммы
BinNumber <- 20
k <- 0:BinNumber

# Вычисление границ карманов на оси X
X_bins <- X_left + k*(X_right - X_left)/BinNumber

# Вычисление границ карманов на оси Y
Y_bins <- Y_left + k*(Y_right - Y_left)/BinNumber

# Построение гистограмм для X и Y
hist(X,X_bins)
hist(Y,Y_bins)
```

**Задание 11.** Создайте файл **test\_my\_func.r**, в котором наберите код, демонстрирующий работу функции **my\_func**. Запустите программу и посмотрите результат ее работы.

**Задание 12.** Создайте функцию **my\_gauss\_gen**, которая генерирует заданное количество случайных величин  $N$ , имеющих нормальное распределение с заданным математическим ожиданием  $m$  и дисперсией  $D$ . Постройте гистограмму (рис. 1.3).

Для этого воспользуйтесь встроенной функцией **rnorm()**, которая позволяет получать реализацию нормальной случайной величины  $a$ .

Код, реализующий созданную функцию (построение гистограммы), сохраните в новом файле **test\_my\_gauss\_gen.r**.

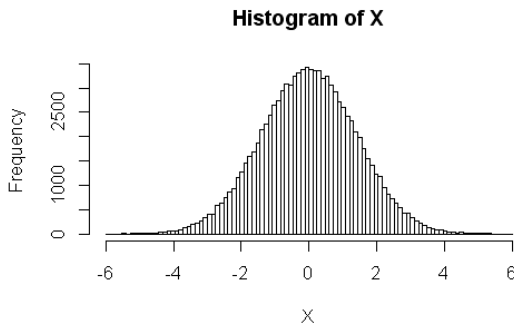


Рис. 1.3. Гистограмма нормальной случайной величины с параметрами  $m=0$  и  $D=2$ ,  $N=10000$ , число каналов гистограммы равно 40

## 14. Циклы

Формирование матрицы с помощью цикла **for**. Пример:

```
n <- 5; # Количество строк
```

```

m <- 10; # Количество столбцов
# Формируем нулевую матрицу Q размером n x m
Q <- matrix(0,nrow = n, ncol = m);
# В цикле заполняем матрицу Q новыми значениями
for (k in 1:n){
  for (j in 1:m){
    Q[k,j] <- round(10*runif(1));
  }
}
print(Q)

```

Цикл *while*. Пример:

```

k <- 0
while(k < 20){
  print(k)
  k <- k+1
}

```

Просмотрите результаты выполнения программы в главном окне RStudio.

**Задание 13.** Создайте файл **test\_cycle.r**, в котором сформируйте матрицу **Q** произвольного размера и заполните ее случайными целыми числами. Выведите в главном окне RStudio матрицу **Q** (добавьте в код команду `print(Q)`).

**Задание 14.** В файле **test\_cycle.r** реализуйте цикл, позволяющий вычислять сумму элементов матрицы **Q**. Необходимый код напишите в файле **test\_cycle.r**.

## 15. Работа с текстовыми файлами

Запись данных в файл с помощью функции *write.table*. Пример:

```

x <- seq(from=0,to=5,by=0.1)
y <- 2*x^2 + x - 1
M <- cbind(x,y)
write.table(M,"MyFile.txt")

```

Чтение данных из файла с помощью функции *read.table*. Пример:

```

A = read.table("MyFile.txt")
plot(A[,1],A[,2], xlab='Argument x', ylab='Function y')
title(main='My polynom function')

```

**Задание 15.** Создайте новый r-файл с именем **write\_txt**, в котором наберите код записи данных с помощью функции *write.table*. Запустите код

на выполнение. Откройте созданный текстовый файл и убедитесь, что запись прошла корректно (в файле есть данные).

**Задание 16.** Создайте новый г-файл с именем **read\_txt**, в котором наберите код считывания из файла с помощью *read.table*. Запустите программу на выполнение. Выведите на экран матрицу данных из файла.

**Форма отчёта:** работающие программные модули, реализующие задания, листинг программ.

### Контрольные вопросы

1. Каким образом производятся вычисления в строке интерпретатора команд в командном окне R или RStudio?
2. Как получить информацию о требуемой функции R?
3. Как образом удалить из оперативной памяти переменные, резервированные во время рабочей сессии?
4. Каким образом производится очистка рабочего окна?
5. Как создать в командной строке вектор, матрицу, трехмерный массив? Выведите в рабочем окне элемент вектора, матрицы, массива.
6. Как добавить заданный столбец/строку к матрице?
7. Что произойдет в результате применения операции  $t(\mathbf{b})$ , где  $\mathbf{b}$  – матрица?
8. Как определить размер вектора, матрицы, массива?
9. С помощью какой функции R производится построение двумерного графика функции? Трехмерного графика функции?
10. С помощью каких функций R производится запись/чтение в/из файл(a)?

## ЛАБОРАТОРНАЯ РАБОТА № 2

### ПРЕДВАРИТЕЛЬНЫЙ АНАЛИЗ БОЛЬШИХ НАБОРОВ БИОЛОГИЧЕСКИХ ДАННЫХ

**Цель работы.** Практическое освоение базовых элементов предварительного анализа данных – очистка данных, вычисление среднего значения и линейного коэффициента корреляции Пирсона.

#### КРАТКИЕ СВЕДЕНИЯ

Методы предварительного анализа данных нацелены на очистку данных, получение информации о центральной тенденции и вариации в данных, оценке степени статистической взаимосвязи в исходных данных.

**Пропущенные значения.** После того как получен набор данных, необходимо оценить качество данных и, если потребуется, выполнить процедуру очистки данных. Под данными низкого качества или грязными данными подразумеваются отсутствующие, неточные или бесполезные данные с точки зрения практического применения. Наиболее распространенный вид грязных данных – пропущенные значения (missing values). Очистка данных направлена на устранение пропущенных значений в данных. Пропущенные значения в экспериментальных данных могут быть обусловлены различными причинами, как правило, экспериментального характера. Пример.  $n_1 = (1, \text{NA}, 10, \text{NA}, 3, 5)$ , где NA (от англ. Not Available) – пропущенное значение. Наиболее простой способ устранения влияния пропущенных значений на результаты работы алгоритмов анализа – исключить объекты с пропущенными значениями из дальнейшей обработки.

*Выборочное среднее значение* – приближение теоретического среднего распределения выборки  $x_{1j}, x_{2j}, \dots, x_{Nj}$  объема  $N$ , основанное на вычислении арифметического среднего значения по выборке из него

$$\bar{X}_j = \frac{1}{N} \sum_{i=1}^N x_{ij}. \quad (2.1)$$

*Выборочный коэффициент корреляции Пирсона* для двух признаков  $X_i$  и  $X_j$ , представляет собой отношение их ковариации к произведению среднеквадратических отклонений этих величин:

$$r_{x_i x_j} = \frac{\text{cov}_{x_i x_j}}{\sigma_{x_i} \sigma_{x_j}}, \quad (2.2)$$

или

$$r_{X_i X_j} = \frac{\sum_{l=1}^N (x_{li} - \bar{X}_i)(x_{lj} - \bar{X}_j)}{\sqrt{\sum_{l=1}^N (x_{li} - \bar{X}_i)^2 \sum_{l=1}^N (x_{lj} - \bar{X}_j)^2}} \quad (2.3)$$

Основные свойства коэффициента корреляции:

1.  $-1 \leq r_{X_i X_j} \leq 1$ .
2.  $r_{X_i X_j} = \pm 1$  – для линейно связанных признаков.
3.  $r_{X_i X_j} = 0$  – для не связанных линейной связью признаков.
4. Коэффициент корреляции – безразмерная величина.

**Данные.** Необходимо выполнить предварительный анализ данных, размещенных в архиве *specdata.zip*. Работа состоит из трех частей, в каждой из которых требуется запрограммировать функцию, позволяющую выполнить заданные расчеты для указанного набора файлов.

В архиве *specdata.zip* размещены 332 файла формата CSV (comma-separated-value), содержащие данные о мониторинге загрязнений воздуха твердыми частицами в различных районах США. Определенный файл содержит данные из одной отдельной лаборатории (далее – источник данных). Идентификационный номер (ID number) источника данных указан в названии файла. Например, данные из источника 200 расположены в файле 200.csv. В файле размещены четыре переменные:

«Date» (Дата) – дата регистрации наблюдения в формате YYYY-MM-DD (year-month-day).

«sulfate» (Сульфаты) – уровень загрязнения воздуха сульфатами в указанный день (мкг/см<sup>3</sup>).

«nitrate» (Нитраты) – уровень загрязнения воздуха нитратами в указанный день (мкг/см<sup>3</sup>).

«ID» – идентификационный номер.

Для выполнения заданий требуется разархивировать файл и создать в рабочем каталоге папку *specdata*.

**Замечание!** Нельзя модифицировать разархивированные файлы. В файлах могут отсутствовать данные (закодированы константой NA).

**Задание 1. Вычисление среднего значения.** Запрограммируйте функцию *pollutantmean*, которая должна вычислить среднее значение уровня загрязнения сульфатами или нитратами для заданного количества источников (задается номерами файлов). Аргументы функции *pollutantmean*: *directory* – путь и название папки с данными; *pollutant* – тип загрязнителя ("sulfate" или "nitrate") и *id* – вектор цифровых значений источников данных. Принимая на входе список аргументов, функция *pollutantmean* счи-

тывает данные о заданном загрязнителе от указанных источников (из папки 'directory'), вычисляет среднее значение по всем наблюдениям, игнорируя пропущенные значения, и возвращает вычисленное среднее значение. Образец прототипа функции:

```
pollutantmean <- function(directory, pollutant, id = 1:332) {  
  ## directory - вектор типа character, указывающий путь к папке,  
  содержащей CSV файлы.  
  ## pollutant - вектор типа character, указывающий имя загрязни-  
  теля ("sulfate" or "nitrate"), для которого необходимо вычислить  
  среднее значение.  
  ## id - вектор типа integer, идентификационные номера файлов  
  данных.  
  ## На выходе функции возвращается среднее значение по всем на-  
  блюдам указанных источников, игнорируя пропущенные значения NA.  
}
```

### Примеры вызова разработанной функции pollutantmean.R.

```
> source("pollutantmean.R")  
> pollutantmean("specdata", "sulfate", 1:10)  
[1] 4.064  
> pollutantmean("specdata", "nitrate", 70:72)  
[1] 1.706  
> pollutantmean("specdata", "nitrate", 23)  
[1] 1.281
```

**Задание 2. Очистка данных от пропусков.** Запрограммируйте функцию, которая считывает указанные в вашем варианте файлы и возвращает количество наблюдений, не содержащих пропущенные значения, для каждого из файлов. Функция должна вернуть объект типа Data Frame, в котором первая колонка – имя файла ('id'), вторая колонка ('nobs') – количество наблюдений, не содержащих пропущенные значения в данном файле. Образец прототипа функции:

```
complete <- function(directory, id = 1:332) {  
  ## directory - вектор типа character, указывающий путь к папке,  
  содержащей CSV файлы.  
  ## id - вектор типа integer, идентификационные номера файлов  
  данных.  
  ## На выходе функции возвращается объект типа Data Frame:  
  ## id nobs  
  ## 1 117  
  ## 2 1041  
  ## ...  
  ## где id - номер источника данных,  
  ## nobs - количество наблюдений, не содержащих пропущен-  
  ные значения в данном файле.  
}
```



**Задание 3. Определение линейной статистической связи.** Запрограммируйте функцию, которая считывает все файлы данных (322 файла) и вычисляет коэффициент корреляции Пирсона между переменными "sulfate" или "nitrate", для которых число не пропущенных значений выше, чем заданный порог threshold. Функция должна вернуть вектор коэффициентов корреляции для источников данных, удовлетворяющих пороговому значению. Если не наблюдается источников данных удовлетворяющих требуемому условию, то возвращается вектор типа numeric длины 0. Образец прототипа функции:

```
corr <- function(directory, threshold = 0) {  
  ## directory - вектор типа character, указывающий путь к папке,  
  ## содержащей CSV файлы.  
  ## threshold - вектор типа numeric, содержащий пороговое значение,  
  ## при превышении которого вычисляется коэффициент корреляции  
  ## Пирсона между переменными "sulfate" или "nitrate" для каждого из  
  ## источников данных  
  ## На выходе функции возвращается вектор типа numeric, содержащий  
  ## корреляционные коэффициенты для источников, удовлетворяющих  
  ## пороговому условию.  
}
```

### Порядок выполнения

1. С помощью *help* изучить функции *dir*, *getwd*, *setwd*, *read.csv*, *length*, *as.character*, *paste*, *mean*, *complete.cases*, *cor*, *data.frame*.
2. Загрузить данные к **заданию 1** согласно вашему варианту. Выполнить **задание 1** (табл. 2.1).
3. Загрузить данные к **заданию 2** согласно вашему варианту. Выполнить **задание 2**.
4. Загрузить все исходные данные. Выполнить **задание 3**, используя значение порога threshold вашего варианта.

**Форма отчёта:** работающие функции, реализующие задания 1-3, тексты программ.

Таблица 2.1

## Варианты заданий

| Вариант | Задание 1.<br>ID, pollutant | Задание 2.<br>ID   | Задание 3.<br>Threshold |
|---------|-----------------------------|--------------------|-------------------------|
| 1       | 1:49, "nitrate"             | 2, 4, 8, 10, 12    | 1000                    |
| 2       | 50:99, "sulfate"            | 3, 5, 9, 11, 13    | 950                     |
| 3       | 99:149, "nitrate"           | 4, 6, 10, 13, 14   | 970                     |
| 4       | 150:199, "sulfate"          | 5, 7, 11, 13, 15   | 980                     |
| 5       | 200:249, "nitrate"          | 6, 8, 12, 14, 16   | 930                     |
| 6       | 250:299, "sulfate"          | 7, 9, 13, 15, 17   | 920                     |
| 7       | 300:322, "nitrate"          | 8, 10, 14, 16, 18  | 935                     |
| 8       | 20:70, "sulfate"            | 9, 11, 15, 17, 19  | 945                     |
| 9       | 120:170, "nitrate"          | 10, 12, 16, 18, 20 | 800                     |
| 10      | 140:190, "sulfate"          | 11, 13, 17, 19, 21 | 850                     |
| 11      | 220:270, "nitrate"          | 12, 14, 18, 20, 22 | 900                     |
| 12      | 240:290, "sulfate"          | 13, 14, 18, 20, 22 | 830                     |

## Контрольные вопросы

1. Перечислите базовые элементы предварительного анализа данных.
2. Для каких целей используется константа NA?
3. Каким образом вычислить среднее значение для набора данных?
4. Как оценить линейную статистическую связь между двумя наборами данных?
5. Для каких целей используются функции R: *dir*, *getwd*, *setwd*, *read.csv*, *length*, *as.character*, *paste*, *mean*, *complete.cases*, *cor*, *data.frame*?

# ЛАБОРАТОРНАЯ РАБОТА № 3

## МЕТОД ГЛАВНЫХ КОМПОНЕНТ

**Цель работы.** Практическое освоение метода главных компонент для решения задач снижения размерности и визуализации многомерных данных.

### КРАТКИЕ СВЕДЕНИЯ

Во многих экспериментах по обработке многомерных данных приходится сталкиваться с задачами связанными с наглядным представлением данных (визуализация данных), снижением размерности данных без существенной потери информативности данных или (сжатием данных), стремлением к лаконизму исследуемых данных (упрощение данных). Для решения подобных задач используются методы, в которых снижение размерности пространства происходит одновременно с его преобразованием. Это – метод главных компонент, факторный анализ, канонический анализ. Характерной особенностью данных методов является то, что происходит выбор и оценка значимости не отдельных переменных, а информативных по совокупности групп переменных. В данной работе рассматривается метод главных компонент.

В общем виде задача снижения размерности признакового пространства может быть сформулирована следующим образом. Пусть многомерные данные представлены  $N$  - числом исследуемых объектов  $n_1, n_2, \dots, n_N$  (табл. 3.1), каждый из которых  $n_i = (x_{i1}, x_{i2}, \dots, x_{iK})$ ,  $i = 1, 2, \dots, K$ , характеризуется набором из  $K$  – признаков  $X_1, X_2, \dots, X_K$  (измеряемых характеристик объектов).

*Таблица 3.1*

**Набор многомерных данных**

|       | $X_1$    | $X_2$    | ... | $X_K$    |
|-------|----------|----------|-----|----------|
| $n_1$ | $x_{11}$ | $x_{12}$ | ... | $x_{1K}$ |
| $n_2$ | $x_{21}$ | $x_{22}$ | ... | $x_{2K}$ |
| ...   | ...      | ...      | ... | ...      |
| $n_N$ | $x_{N1}$ | $x_{N2}$ |     | $x_{NK}$ |

Необходимо определить такое линейное преобразование, задаваемое матрицей  $A$ , в результате действия которого исходные данные выражаются набором (матрицей) главных компонент  $Z = (Z_1, Z_2, \dots, Z_K)$ , где первые  $M$  – главных компонент ( $M \ll K$ ), обеспечивают требуемую долю дисперсии  $\gamma$  групп признаков (как правило,  $\gamma \geq 0.95$ ).

Матрица главных компонент строится на основе матрицы весовых коэффициентов  $A = \{a_{ij}\}$ ,  $i=1, 2, \dots, K$  и  $j=1, 2, \dots, K$ , учитывающих тесноту связи между исходными признаками и главными компонентами:

$$Z = XA, \quad (3.1)$$

где  $Z = (Z_1, Z_2, \dots, Z_K)$  – матрица всех  $K$  главных компонент и  $X = (X_1, X_2, \dots, X_K)$  – матрица исходных признаков. В развернутом виде  $j$ -ая главная компонента матрицы  $Z$  выражается через векторы признаков следующим образом:

$$Z_j = a_{1j} X_1 + a_{2j} X_2 + \dots + a_{Kj} X_K, \quad (3.2)$$

где элементы  $a_{ij}$ ,  $i=1, 2, \dots, K$  и  $j=1, 2, \dots, K$ , – называются параметрами загрузки главных компонент.

Задача сводится к определению матрицы  $A$  (нахождению неизвестных параметров  $a_{ij}$ , где  $i=1, 2, \dots, K$  и  $j=1, 2, \dots, K$ ). Для этого используется аппарат матричной алгебры. Элементы матрицы  $A$  рассчитываются на основе корреляционной матрицы  $R$  (построенной по входным данным  $n_i$ ,  $i=1, 2, \dots, N$ ), решением систем линейных уравнений:

$$RA = A\Lambda, \quad (3.3)$$

где матрица

$$A = \begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \lambda_K \end{pmatrix} \quad (3.4)$$

содержит на главной диагонали собственные значения корреляционной матрицы  $R$ , такие, что  $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_K$ , а координаты собственных векторов являются искомыми параметрами  $a_{ij}$   $i=1, 2, \dots, K$  и  $j=1, 2, \dots, K$ . При этом выполняется равенство

$$\lambda_i = \sigma^2(Z_i), \quad i=1, 2, \dots, K. \quad (3.5)$$

Сумма выборочных дисперсий исходных признаков равна сумме выборочных дисперсий проекций объектов на главные компоненты

$$\sigma^2(Z_1) + \sigma^2(Z_2) + \dots + \sigma^2(Z_K) = \sigma^2(X_1) + \sigma^2(X_2) + \dots + \sigma^2(X_K). \quad (3.6)$$

Несмотря на то, что вместо  $K$  признаков получается такое же количество главных компонент, вклад большей части главных компонент в объясняемую дисперсию оказывается небольшим. Исключают из рассмотрения те главные компоненты, вклад которых мал. При помощи  $M$  первых (наиболее весомых) главных компонент можно объяснить основную часть суммарной дисперсии в данных.

Относительная доля разброса, приходящаяся на  $j$ -ую главную компоненту

$$\alpha_j = \frac{\sigma^2(Z_j)}{\sigma^2(Z_1) + \sigma^2(Z_2) + \dots + \sigma^2(Z_K)} = \frac{\lambda_j}{\lambda_1 + \lambda_2 + \dots + \lambda_K}. \quad (3.7)$$

Относительная доля разброса, приходящаяся на  $i$  первых компонент

$$\gamma_i = \frac{\sigma^2(Z_1) + \sigma^2(Z_2) + \dots + \sigma^2(Z_i)}{\sigma^2(Z_1) + \sigma^2(Z_2) + \dots + \sigma^2(Z_K)} = \frac{\lambda_1 + \lambda_2 + \dots + \lambda_i}{\lambda_1 + \lambda_2 + \dots + \lambda_K}. \quad (3.8)$$

Таким образом, метод главных компонент позволяет описать большой набор признаков  $K$  небольшим числом главных компонент  $M$ ,  $M \ll K$ , при этом различия между объектами зависят от доли изменчивости, связанной с данной главной компонентой. Связи между признаками и главными компонентами – линейные.

На основе выявленных наиболее весомых главных компонент, обычно проводится компонентный анализ для объяснения индивидуальных значений данных главных компонент для рассматриваемых объектов, ранжирования объектов по весу этих значений, выбора наиболее оптимальных объектов.

### ***Алгоритм метода главных компонент***

*Шаг 1.* Сформировать матрицу входных данных на основе объектов  $n_i$ ,  $i=1, 2, \dots, N$

$$X = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1K} \\ x_{21} & x_{22} & \dots & x_{2K} \\ \dots & \dots & \dots & \dots \\ x_{N1} & x_{N2} & \dots & x_{NK} \end{pmatrix}. \quad (3.9)$$

*Шаг 2.* Для устранения неоднородности в исходных данных выполнить нормировку (стандартизацию) данных по колонкам

$$x'_{ij} = \frac{x_{ij} - \bar{X}_j}{\sigma(X_j)}, \quad (3.10)$$

где  $\bar{X}_j$  и  $\sigma(X_j)$  – математическое ожидание и среднее квадратическое отклонение по  $j$ -ой колонке, матрица  $X' = \{x'_{ij}\}$ ,  $i = 1, 2, \dots, N$  и  $j = 1, 2, \dots, K$ .

*Шаг 3.* Построить матрицу ковариации

$$Cov = R = (X'^T X') / (N-1) \quad (3.11)$$

Ввиду произведенной нормализации данных матрица ковариаций будет корреляционной матрицей исходных данных  $R$  порядка  $K \times K$ .

*Шаг 4.* Вычислить матрицы  $A$  и  $\Lambda$  путем решения систем линейных уравнений (3.3).

*Шаг 5.* Рассчитать проекции объектов на главные компоненты

$$Z = X^*A. \quad (3.12)$$

**Замечание.** В заключение отметим, прежде чем начинать анализ главных компонент целесообразно проверить значимо ли отличается от единичной матрицы корреляционная матрица исходных нормированных данных. В предположении, что исходные данные подчиняются многомерному нормальному распределению, можно воспользоваться статистикой

$$d = N \sum_{i=1}^K \sum_{j=i+1}^K r_{ij}^2 \quad (3.13)$$

где  $r_{ij}$  – наддиагональные элементы корреляционной матрицы  $R$ . Статистика  $d$  подчиняется  $\chi^2$ -распределению с  $K(K-1)/2$  степенями свободы. Если корреляционная матрица исходных данных не отличается от единичной матрицы, т. е.  $d \leq \chi^2$ , вычисленное при заданном уровне доверительной вероятности и заданном числе степеней свободы, то применение метода главных компонент нецелесообразно.

### Порядок выполнения

1. С помощью *help* изучить функции *read.table*, *dim*, *as.matrix*, *colMeans*, *apply*, *sd*, *t*, *print*, *sum*, *qchisq*, *eigen*, *%\*%*, *var*.

2. Разработать алгоритм метода главных компонент и программно реализовать его в среде R.

3. Выполнить анализ экспериментальных данных методом главных компонент.

- Загрузить данные согласно вашему варианту (табл. 3.2).
  - Нормировать (стандартизировать) исходные экспериментальные данные. Построить корреляционную матрицу.
  - Удостовериться, что корреляционная матрица значимо отличается от единичной матрицы.
  - Рассчитать проекции объектов на главные компоненты.
4. Произвести анализ результатов работы метода главных компонент.
- Проверить равенство сумм выборочных дисперсий исходных признаков и выборочных дисперсий проекций объектов на главные компоненты.
  - Определить относительную долю разброса, приходящуюся на главные компоненты. Построить матрицу ковариации для проекций объектов на главные компоненты.

- На основе первых  $M = 2$  главных компонент построить диаграмму рассеяния.

**Форма отчета:** работающий программный модуль, реализующий задание, тексты программ. Результаты метода главных компонент. Выводы по результатам обработки экспериментальных данных.

Таблица 3.2

Варианты заданий

| Вариант | Данные    | Вариант | Данные     |
|---------|-----------|---------|------------|
| 1       | data1.txt | 7       | data7.txt  |
| 2       | data2.txt | 8       | data8.txt  |
| 3       | data3.txt | 9       | data9.txt  |
| 4       | data4.txt | 10      | data10.txt |
| 5       | data5.txt | 11      | data11.txt |
| 6       | data6.txt | 12      | data12.txt |

### Контрольные вопросы

1. Для каких задач обработки экспериментальных данных используется метод главных компонент?
2. В чем состоит суть метода главных компонент?
3. Чему равно математическое ожидание и дисперсия стандартизованной переменной?
4. Какова дисперсия  $j$ -й главной компоненты?
5. Чему равна сумма выборочных дисперсий проекций объектов на главные компоненты?
6. Какова относительная доля разброса, приходящаяся на  $j$ -ую главную компоненту?
7. Какова относительная доля разброса, приходящаяся на  $i$  первых главных компонент?
8. Целесообразно ли использование метода главных компонент, если ковариационная матрица исходных признаков диагональная?
9. Как проверить значимость корреляционной матрицы исходных данных?
10. Какой вид связи между признаками и главными компонентами?
11. Для каких целей используются функции R: *read.table*, *dim*, *as.matrix*, *colMeans*, *apply*, *sd*, *t*, *print*, *sum*, *qchisq*, *eigen*, *%\*%*, *var*?

## ЛАБОРАТОРНАЯ РАБОТА № 4

### ИЕРАРХИЧЕСКИЕ МЕТОДЫ КЛАСТЕРНОГО АНАЛИЗА

**Цель работы.** Практическое освоение методов иерархического кластерного анализа данных.

#### КРАТКИЕ СВЕДЕНИЯ

Решение задачи разбиения множества элементов без обучения называется кластерным анализом. Кластерный анализ не требует априорной информации о данных и позволяет разделить множество исследуемых объектов на группы похожих объектов – кластеры. Это дает возможность резко сокращать большие объемы данных, делать их компактными и наглядными.

Задачу кластеризации можно сформулировать следующим образом.

Имеется некоторое конечное множество объектов произвольной природы  $C = \{n_1, n_2, \dots, n_N\}$ , каждый из которых  $n_i = (x_{i1}, x_{i2}, \dots, x_{iK})$ ,  $i = 1, 2, \dots, N$ , характеризуется набором из  $K$  – признаков (измеряемых характеристик объектов). Необходимо кластеризовать эти объекты, т.е. разбить их множество на заданное или произвольное количество групп (кластеров или классов) таким образом, чтобы в каждую группу оказались включенными объекты, близкие между собой в том или ином смысле. Априорная информация о классификации объектов при этом отсутствует. Таким образом, необходимо разбить множество векторов  $C$  на  $k$  попарно непересекающихся классов  $C_1, C_2, \dots, C_k$  так, чтобы  $\bigcup_{i=1}^k C_i = C$ , где  $1 < k < N$ .

Для нахождения определенного решения данной задачи необходимо задать:

- способ сравнения объектов между собой (меру сходства);
- способ кластеризации;
- разбиение данных по кластерам (установление числа кластеров).

Для сравнения двух объектов  $n_i$  и  $n_j$  могут быть использованы следующие меры:

1. Евклидово расстояние



$$d_2(n_i, n_j) = \left[ \sum_{l=1}^K (x_{il} - x_{jl})^2 \right]^{\frac{1}{2}}; \quad (4.1)$$

2. стандартизированное Евклидово расстояние

$$d(n_i, n_j) = \left[ \sum_{l=1}^K \frac{(x_{il} - x_{jl})^2}{\sigma_l^2} \right]^{\frac{1}{2}}, \quad (4.2)$$

где  $\sigma_l^2$  – дисперсия по  $l$ -му признаку;

3. метрика города

$$d_H(n_i, n_j) = \sum_{l=1}^K |x_{il} - x_{jl}|; \quad (4.3)$$

4. расстояние Минковского

$$d_{Mnkw}(n_i, n_j) = \left[ \sum_{l=1}^K |x_{il} - x_{jl}|^p \right]^{\frac{1}{p}}, \quad (4.4)$$

где  $p = 1, 2, \dots, \infty$ ;

5. расстояние Канберры (Canberra)

$$d_{Can}(n_i, n_j) = \sum_{l=1}^K \frac{|x_{il} - x_{jl}|}{|x_{il} + x_{jl}|}; \quad (4.5)$$

6. метрика Чебышева

$$d_C(n_i, n_j) = \max_{1 \leq l \leq K} |x_{il} - x_{jl}|. \quad (4.6)$$

Иерархические методы используют последовательное объединение объектов в кластеры, малые кластеры в большие, которое может быть визуально отображено в виде дерева вложенных кластеров – дендрограммы (рис. 4.1), при этом число кластеров скрыто или условно. Как правило, на графе дендрограммы вдоль оси  $x$  располагают номера объектов, а вдоль оси  $y$  – значения меры сходства. Иерархические методы делятся на:

- *агломеративные*, характеризующиеся последовательным объединением исходных объектов и соответствующим уменьшением числа кластеров (построение кластеров снизу вверх);
- *дивизимные* (делимые), в которых число кластеров возрастает начиная с одного, в результате чего образуется последовательность расщепляющихся групп (построение кластеров сверху вниз);
- *гибридные*, сочетающие положительные возможности механизмов кластеризации как агломеративных, так и дивизимных алгоритмов.

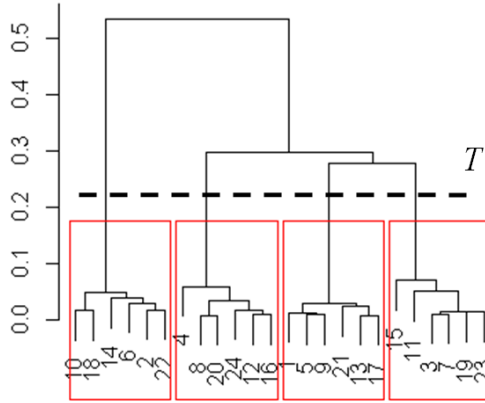


Рис. 4.1. Дендрограмма

Алгоритмы иерархического кластерного анализа различаются по способу или методу связывания объектов в кластеры. Если кластеры  $i$  и  $j$  объединяются в кластер  $r$  и требуется рассчитать расстояние до кластера  $s$ , то наиболее часто используются следующие методы связывания объектов/кластеров  $r$  и  $s$ :

1. метод ближнего соседа (расстояние между ближайшими соседями-ближайшими объектами кластеров)

$$d_{rs} = \min(d_{is}, d_{js}); \quad (4.7)$$

2. метод дальнего соседа (расстояние между самыми дальними соседями)

$$d_{rs} = \max(d_{is}, d_{js}); \quad (4.8)$$

3. метод средней связи (среднее расстояние между всеми объектами пары кластеров с учетом расстояний внутри кластеров)

$$d_{rs} = \frac{m_i d_{is} + m_j d_{js}}{m_i + m_j}, \quad (4.9)$$

где  $m_i$  и  $m_j$  – число объектов в  $i$ -ом и  $j$ -ом кластерах соответственно;

4. центроидный метод

$$d_{rs} = \frac{m_i d_{is} + m_j d_{js}}{m_i + m_j} - \frac{m_i m_j d_{ij}}{(m_i + m_j)^2}; \quad (4.10)$$

5. метод медианной связи

$$d_{rs} = (d_{is} + d_{js}) / 2 - d_{ij} / 4. \quad (4.11)$$

Таким образом, в большой кластер объединяются те малые кластеры, расстояние между которыми минимально.

**Оценка различия.** Используется для определения наиболее оптимального выбора метрического расстояния и метода связывания объектов. Два наугад выбранных объекта на иерархическом дереве связаны между собой так называемым кофенетическим расстоянием, величина которого определяется расстоянием между двумя кластерами, в которых находятся данные объекты. Мерой линейной связи между кофенетическими и метрическими расстояниями является кофенетический корреляционный коэффициент  $k$ . Построение иерархического дерева считается успешным, если кофенетический корреляционный коэффициент близок к 1. Для наиболее успешной кластеризации строится дендрограмма иерархического дерева.

**Выделение значимых кластеров.** Для выделения значимых кластеров можно задать некоторое пороговое значение  $T$  меры расстояний сходства (горизонтальная ось  $T$  на дендрограмме рис. 4.1). Число значимых кластеров определяется количеством пересечений линии порога  $T$  и связей иерархического дерева. Причем каждая из отсекаемых линией порога ветвей дерева будет формировать отдельный кластер. На практике часто выбирают пороговое значение  $T$  на основе визуального анализа плотности ветвей построенной дендрограммы.

Альтернативный способ выделения значимых кластеров – метод задания фиксированного числа кластеров. Пороговое значение меры сходства  $T$  устанавливается в корне иерархического дерева. Затем значение порога  $T$  постепенно снижается до тех пор, пока не будет установлено число пересечений линии порога  $T$  и связей иерархического дерева равное заданному количеству кластеров.

## **Основные этапы иерархического кластерного анализа в R**

*Этап 1.* Вычисление матрицы расстояния между объектами  $D$  (функция *dist*).

*Этап 2.* Связывание или группировка объектов в иерархические деревья (дендрограммы) (функция *hclust*).

*Этап 3.* Оценка качества кластеризации (функции *cophenetic* и *cor*).

*Этап 4.* Выделение значимых кластеров (функция *cutree*).

*Этап 5.* Визуализация и анализ значимых кластеров (функции *plot* и *rect.hclust*).

## Порядок выполнения

1. С помощью *help* изучить функции *dist*, *hclust*, *cophenetic*, *cutree*, *rect.hclust*, *points*.
2. Загрузить данные согласно вашему варианту (табл. 4.1). Построить графическое изображение экспериментальных данных.
3. Вычислить расстояния между объектами. Использовать меры для расчета расстояний согласно варианту (табл. 4.1).
4. Выполнить кластерный анализ исходных данных методом иерархической кластеризации по методам связывания согласно варианту.
5. Выполнить анализ качества кластеризации с помощью вычисления кофенетического корреляционного коэффициента. Заполнить таблицу для кофенетического корреляционного коэффициента по расстояниям и методам связывания согласно варианту.
6. Определить наиболее и наименее эффективные способы иерархической кластеризации для анализа исходного набора данных (максимальные и минимальные коэффициенты и соответствующие им способы кластеризации). Для наиболее эффективного способа иерархической кластеризации построить дендрограмму результатов кластерного анализа.
7. Определить количество достоверных кластеров. Для выделения значимых кластеров использовать пороговое значение, рассчитанное по метрике расстояний или методом задания фиксированного числа кластеров.
8. Рассчитать центры и внутрикластерную дисперсию полученных кластеров

$$D_{C_k} = \frac{1}{N_{C_k}} \sum_{i=1}^{N_{C_k}} d^2(n_i, c), \quad (4.12)$$

где  $N_{C_k}$  – число объектов в рассматриваемом кластере, геометрические расстояния от элементов до центров кластеров, расстояния между центрами кластеров. Отобразить графически найденные кластеры и их центры (использовать диаграмму рассеяния в цвете).

**Форма отчета:** работающий программный модуль, реализующий задание, тексты программ. Результаты иерархической кластеризации. Выводы по результатам обработки экспериментальных данных.

Таблица 4.1

## Варианты заданий

| Вариант | Метрики (расстояния) | Методы связывания | Данные     |
|---------|----------------------|-------------------|------------|
| 1       | 1, 2, 3              | a, b, c           | data1.txt  |
| 2       | 1, 3, 4              | a, b, d,          | data2.txt  |
| 3       | 1, 4, 5              | a, b, e           | data3.txt  |
| 4       | 1, 5, 6              | a, c, e           | data4.txt  |
| 5       | 1, 2, 4              | a, c, d           | data5.txt  |
| 6       | 1, 3, 5              | a, d, e           | data6.txt  |
| 7       | 1, 4, 6              | b, c, d           | data7.txt  |
| 8       | 1, 2, 5              | b, c, e           | data8.txt  |
| 9       | 1, 3, 6              | b, d, e           | data9.txt  |
| 10      | 1, 2, 6              | c, d, e           | data10.txt |
| 11      | 2, 4, 6              | a, b, c           | data11.txt |
| 12      | 2, 3, 6              | a, b, d,          | data12.txt |

\* Метрики расстояния: 1 – Евклидово, 2 – стандартизированное Евклидово, 3 – города, 4 – Канберра, 5 – Минковского ( $p = 4$ ), 6 – Чебышева (method = 'maximum').

\*\* Методы связывания: a – ближнего соседа (method = 'single'), b – дальнего соседа (method = 'complete'), c – средней связи (method = 'average'), d – центроидный (method = 'centroid'), e – медианной связи (method = 'median').

## Контрольные вопросы

1. В чем заключается задача кластерного анализа?
2. Для каких задач обработки экспериментальных данных используются методы иерархического кластерного анализа?
3. Перечислите основные меры сравнения объектов между собой.
4. Что такое дендрограмма?
5. Что представляют собой иерархические агломеративные, дивизимные и гибридные алгоритмы кластерного анализа?
6. Перечислите основные способы связывания объектов в кластеры.
7. Что такое кофенетический корреляционный коэффициент?
8. Перечислите этапы иерархического кластерного анализа.
9. Каким образом определить значимое число кластеров?
10. Для каких целей используются функции R: *dist*, *hclust*, *cophenetic*, *cutree*, *rect.hclust*, *points*?

## ЛАБОРАТОРНАЯ РАБОТА № 5

### НЕИЕРАРХИЧЕСКИЕ МЕТОДЫ КЛАСТЕРНОГО АНАЛИЗА

**Цель работы.** Практическое освоение неиерархического кластерного анализа многомерных данных на примере метода  $k$ -средних.

#### КРАТКИЕ СВЕДЕНИЯ

Задачей кластерного анализа является организация наблюдаемых данных в наглядные структуры – кластеры. Исходные данные могут быть представлены в виде матрицы  $X$ , строки которой соответствуют объектам (наблюдениям), а столбцы их признакам (табл. 5.1). Требуется разбить заданное множество объектов на кластеры.

Таблица 5.1

#### Многомерные данные

|       | $X_1$    | $X_2$    | ... | $X_K$    |
|-------|----------|----------|-----|----------|
| $n_1$ | $x_{11}$ | $x_{12}$ | ... | $x_{1K}$ |
| $n_2$ | $x_{21}$ | $x_{22}$ | ... | $x_{2K}$ |
| ...   | ...      | ...      | ... | ...      |
| $n_N$ | $x_{N1}$ | $x_{N2}$ |     | $x_{NK}$ |

При большом количестве объектов ( $N > 200$ ) иерархические методы кластерного анализа непригодны, ввиду больших вычислительных затрат и сложности интерпретации дерева кластеров. В таких случаях могут использоваться неиерархические методы кластерного анализа, одним из которых является метод  $k$ -средних. Суть этого метода заключается в следующем. Предположим, уже имеются гипотезы относительно числа кластеров – т.е. заранее определено количество кластеров  $k$ , на которые необходимо разбить имеющиеся объекты. Среди множества объектов выбираются  $k$  объектов в качестве начальных центров кластеров. Для каждого объекта рассчитываются расстояния до центров кластеров, и данный объект относится к тому кластеру, расстояние до которого оказалось минимальным. После чего, для этого кластера (в котором изменилось количество объектов) рассчитывается новое положение центра кластера (как среднее по каждому признаку  $X_j$ ) по всем включенным в кластер объектам. В общем случае, в результате применения метода  $k$ -средних исходное множество объектов разделяется ровно на  $k$  различных кластеров, расположенных на возможно больших расстояниях друг от друга.

Расстояние между объектами  $n_i$  и  $n_j$  может быть вычислено, например, по следующим формулам:

1. Евклидово расстояние

$$d_2(n_i, n_j) = \left[ \sum_{l=1}^K (x_{il} - x_{jl})^2 \right]^{\frac{1}{2}}; \quad (5.1)$$

2. расстояние Минковского

$$d_{Minkw}(n_i, n_j) = \left[ \sum_{l=1}^K |x_{il} - x_{jl}|^p \right]^{\frac{1}{p}}. \quad (5.2)$$

Для хранения информации о принадлежности объекта к некоторому кластеру в методе  $k$ -средних вводится матрица  $U = \{u_{ij}\}$ ,  $i = 1, 2, \dots, N$  и  $j = 1, 2$ . В первом столбце матрицы  $U$  содержатся индексы кластеров, к которым относятся объекты данных, во втором столбце – расстояния от объектов до соответственных центров кластеров.

Алгоритм  $k$ -средних является итерационным. Блок-схема алгоритма неиерархической кластеризации по методу  $k$ -средних представлена на рис. 5.1.

### *Алгоритм метода $k$ -средних*

*Шаг 1.* Инициализация начальных параметров метода (блок 1, рис. 5.1).

Задать:  $k$  - количество предполагаемых кластеров, матрицу координат центров кластеров  $C^{(0)} = \{c_l^{(0)}\}$ ,  $l = 1, 2, \dots, k$  (например, выбрать  $k$  объектов из файла исходных данных), матрицу  $U^{(0)}$ , начальное значение функционала качества кластеризации  $Q^{(0)}$  (некоторое большое число) и точность его вычисления  $\varepsilon$  (для остановки алгоритма). Установить номер итерации  $m = 1$ .

*Шаг 2.* Рассчитать расстояния от объектов  $n_1, n_2, \dots, n_N$  до центров кластеров  $C^{(m-1)}$ , определенных на предыдущей итерации. Заполнить матрицу  $U^{(m)}$ , исходя из расположения центров  $C^{(m-1)}$  вычисленных на предыдущей итерации (блок 2). Рассчитать значение функционала качества кластеризации  $Q^{(m)}$  (с учетом  $C^{(m-1)}$ ).

*Шаг 3.* Проверить условие остановки алгоритма  $|Q^{(m)} - Q^{(m-1)}| \leq \varepsilon$  (блок 3). При этом оценивается, привело ли новое объединение объектов в кластеры к существенному улучшению качества кластеризации. Если условие выполняется, то завершить процесс кластеризации (блок 6). Иначе перейти к шагу 4.

*Шаг 4.* Рассчитать новое положение центров кластеров  $C^{(m)}$  как среднее арифметическое по координатам объектов, входящих в соответствующие кластеры (блок 4).

Шаг 5. Установить  $Q^{(m-1)} = Q^{(m)}$  и перейти к шагу 2 (новой итерации) с  $m = m + 1$  (блок 5).

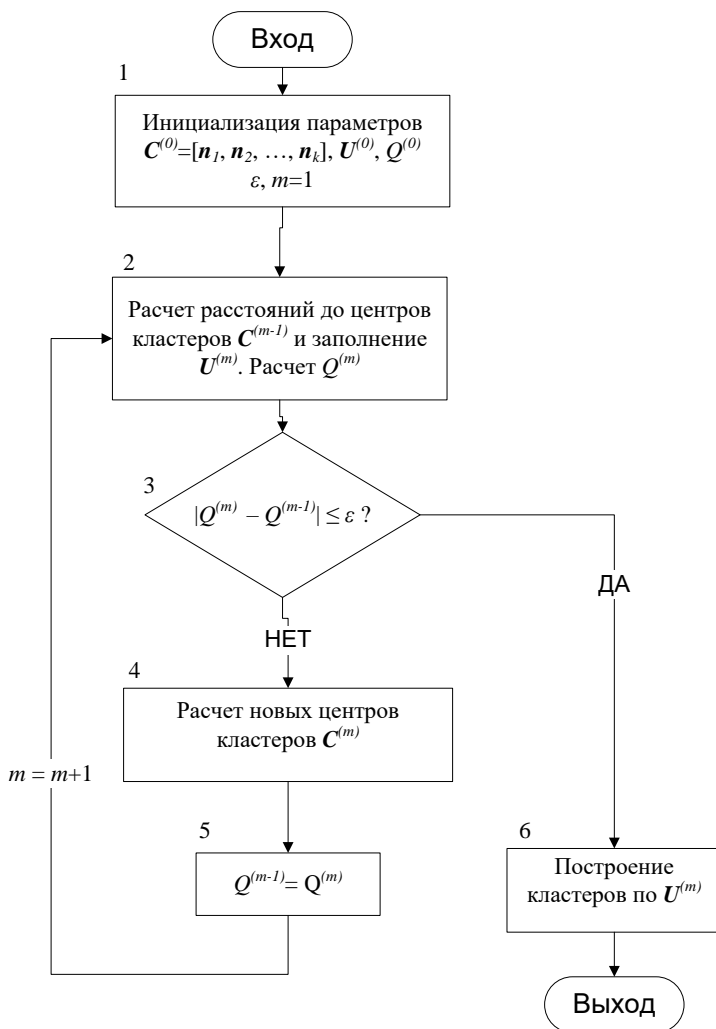


Рис. 5.1. Блок-схема алгоритма неиерархической кластеризации по методу  $k$ -средних

Автоматическая группировка объектов в кластеры прекращается при выполнении одного из критериев остановки автоматического группирова-



ния. На практике используются следующие виды критериев качества автоматического группирования (функционалы качества кластеризации):

1. Сумма расстояний до центров кластеров:

$$Q_1 = \sum_{l=1}^k \sum_{i \in S_l} d(n_i, c_l) \rightarrow \min, \quad (5.3)$$

$l$  – номер кластера,  $l = 1, 2, \dots, k$ ,  $n_i$  – вектор признаков  $i$ -го объекта в  $l$ -ом кластере,  $S_l$  – множество объектов в  $l$ -ом кластере.

2. Сумма квадратов расстояний до центров кластеров:

$$Q_2 = \sum_{l=1}^k \sum_{i \in S_l} d^2(n_i, c_l) \rightarrow \min, \quad (5.4)$$

$l$  – номер кластера,  $l = 1, 2, \dots, k$ ,  $n_i$  – вектор признаков  $i$ -го объекта в  $l$ -ом кластере,  $S_l$  – множество объектов в  $l$ -ом кластере.

3. Сумма внутрикластерных расстояний между объектами:

$$Q_3 = \sum_{l=1}^k \sum_{i, j \in S_l} d(n_i, n_j) \rightarrow \min. \quad (5.5)$$

Основным недостатком алгоритма  $k$ -средних является то, что осуществляется локальная, а не глобальная минимизация функционала  $Q$ .

### Порядок выполнения

1. Изучить функции R: *source*, *abs*, *min*, *which.min*, *rbind*, *for*, *while*, *is.null*, *sum*, *list*.
2. Загрузить данные согласно вашему варианту (табл. 5.2). Построить графическое изображение экспериментальных данных (диаграмму рассеяния). Визуально оценить число кластеров  $k$  по построенному изображению.
3. Разработать алгоритм кластеризации  $k$ -средних и программно его реализовать в среде R.
4. Выполнить кластерный анализ исходных данных методом  $k$ -средних (параметры метода см. в табл. 5.2).
5. Рассчитать центры полученных кластеров. Отобразить графически найденные кластеры (использовать диаграмму рассеяния в цвете).

**Форма отчета:** работающий программный модуль, реализующий задание, тексты программ. Результаты неиерархической кластеризации алгоритмом  $k$ -средних. Выводы по результатам обработки экспериментальных данных.

Таблица 5.2

## Варианты заданий

| Вариант | Метрика (расстояние)    | Функционал качества кластеризации | Данные     |
|---------|-------------------------|-----------------------------------|------------|
| 1       | Евклидово               | $Q_1$                             | data1.txt  |
| 2       | Евклидово               | $Q_2$                             | data2.txt  |
| 3       | Евклидово               | $Q_3$                             | data3.txt  |
| 4       | Евклидово               | $Q_1$                             | data4.txt  |
| 5       | Евклидово               | $Q_2$                             | data5.txt  |
| 6       | Евклидово               | $Q_3$                             | data6.txt  |
| 7       | Минковского ( $P = 4$ ) | $Q_1$                             | data7.txt  |
| 8       | Минковского ( $P = 4$ ) | $Q_2$                             | data8.txt  |
| 9       | Минковского ( $P = 4$ ) | $Q_3$                             | data9.txt  |
| 10      | Минковского ( $P = 4$ ) | $Q_1$                             | data10.txt |
| 11      | Минковского ( $P = 4$ ) | $Q_2$                             | data11.txt |
| 12      | Минковского ( $P = 4$ ) | $Q_3$                             | data12.txt |

## Контрольные вопросы

1. В чем заключается задача неиерархического кластерного анализа?
2. Для каких задач обработки экспериментальных данных используются методы неиерархического кластерного анализа?
3. В чем суть алгоритма  $k$ -средних?
4. Перечислите основные этапы неиерархического кластерного анализа по методу  $k$ -средних.
5. Для каких целей в алгоритме  $k$ -средних вводится матрица разбиений  $U$ ?
6. Какие критерии остановки автоматической кластеризации используются на практике?
7. Каким образом оценить число кластеров в алгоритме  $k$ -средних?
8. Для каких целей используются функции R: *source*, *abs*, *min*, *which.min*, *rbind*, *for*, *while*, *is.null*, *sum*, *list*?

## ЛАБОРАТОРНАЯ РАБОТА № 6

### НЕЙРОННЫЕ СЕТИ. СЛОЙ КОХОНЕНА

**Цель работы.** Изучение алгоритма нейронной сети слой Кохонена для решения задачи кластерного анализа.

#### КРАТКИЕ СВЕДЕНИЯ

**Нейронные сети.** Нейронные сети представляют собой широкий круг алгоритмов машинного обучения и используются для решения задач классификации, кластеризации, сглаживания функций, прогнозирования, оптимизации, управления и адресации памяти.

Основой нейронной сети является искусственный нейрон – математическая модель нейрона мозга живых организмов. Искусственный нейрон состоит из синапсов (умножителей), ядра и аксона. Нейрон воспринимает входные сигналы через синапсы. Каждый синапс имеет вес, который определяет, насколько соответствующий вход нейрона влияет на его состояние. Ядро нейрона состоит из сумматора и нелинейного преобразователя. Сумматор выполняет сложение сигналов, поступающих по синапсам от других нейронов, и внешних входных сигналов. Нелинейный преобразователь реализует нелинейную функцию одного аргумента – функцию активации (передаточную функцию). Выходной сигнал формируется на аксоне. Через аксон отдельный нейрон может связываться с другими нейронами.

Для формирования искусственной нейронной сети отдельные нейроны объединяются в сети с определенной топологией. Функции активации нейронов выбираются и фиксируются на этапе проектирования сети, веса сети являются её параметрами и могут изменять свои значения. Работа нейронной заключается в преобразовании независимых переменных объекта  $n_i = (x_{i1}, x_{i2}, \dots, x_{iK})$  в выходную зависимую переменную  $y_i$ .

Общий принцип разработки нейронной сети состоит в создании модели нейронов сети и поиска неизвестных весов синапсов нейронов. Неизвестные веса определяются таким образом, чтобы минимизировать ошибку обучения – некоторую функцию разности между выходными и желаемыми значениями зависимой переменной.

**Слой Кохонена (слой «состязующихся» нейронов).** Нейронные сети Кохонена представляют собой класс нейронных сетей, основным элементом которых является слой Кохонена. Алгоритм функционирования самообучающегося слоя Кохонена является одним из вариантов кластеризации

многомерных данных. По аналогии с алгоритмом кластеризации  $k$ -средних, нейроны слоя Кохонена представляют собой узлы или центры кластеров.

Пусть выполняется анализ некоторого набора данных  $n_1, n_2, \dots, n_N$ . Метки классов  $y_1, y_2, \dots, y_N$  объектов не известны. Требуется разбить набор данных на  $m$  классов, выделив  $m$  эталонных образцов (центров кластеров или классов).

Нейроны слоя Кохонена представляют собой узлы или центры кластеров, синаптические веса которых  $W_1, W_2, \dots, W_m$  должны быть определены. Нейронная сеть анализирует выходные значения нейронов слоя и в качестве результата выдает номер наиболее близко расположенного нейрона сети (нейрона-«победителя») к классифицируемому объекту (рис. 6.1). Номер активного нейрона-«победителя» определяет ту группу (кластер), к которой наиболее близок входной вектор-объект. В результате победивший нейрон, вероятно, выиграет конкуренцию и в том случае, когда будет представлен новый входной вектор, близкий к предыдущему, и его победа менее вероятна, когда будет представлен вектор, существенно отличающийся от предыдущего. Когда на вход сети поступает всё большее и большее число векторов, нейрон, являющийся ближайшим, снова корректирует свой весовой вектор  $W_j$ . В конечном счёте, если в слое имеется достаточное количество нейронов, то каждая группа близких векторов окажется связанной с одним из нейронов слоя. В этом и заключается свойство самоорганизации слоя Кохонена.

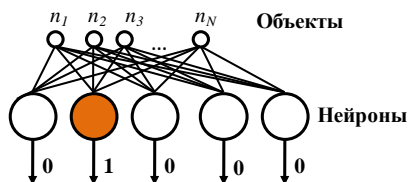


Рис. 6.1. Слой Кохонена. Положительный сигнал (единица) формируется только у одного нейрона – «нейрона-победителя», расположенного наиболее близко к входному объекту  $n_i$ .

$K$ -входов на каждый из нейронов представлены в виде сплошных линий

В общем случае задача сводится к поиску минимума некоторой функции ошибок, например – суммы квадратов расстояний до центров кластеров или нейронов слоя:

$$F = \sum_{l=1}^m \sum_{i \in C_l} d^2(n_i, W_l) \rightarrow \min, \quad (6.1)$$

$l$  – номер кластера/нейрона,  $l = 1, 2, \dots, m$ ,  $n_i$  – вектор признаков  $i$ -го объекта в  $l$ -ом кластере,  $C_l$  – множество объектов в  $l$ -ом кластере.

Минимизацию функции  $F$  относительно ядра  $W_j$  можно осуществить методом стохастического градиентного спуска по итерационной формуле

$$W_j^{(k)} = W_j^{(k-1)} + h \cdot (n_k - W_j^{(k-1)}), \quad (6.2)$$

где  $h$  – параметр скорости обучения.

Если вектор  $n_k$  относится к кластеру  $C_j$ , то нейрон этого кластера  $W_j$  смещается в сторону объекта  $n_k$  на расстояние  $h \cdot (n_k - W_j^{(k-1)})$ .

### *Алгоритм нейронной сети слой Кохонена*

1. Задать размер слоя  $m$ . Инициализировать начальные значения нейронов  $W_1^{(0)}, W_2^{(0)}, \dots, W_m^{(0)}$ . Установить номер итерации  $k = 1$  и максимальное число итераций  $k_{max}$ .

2. Слою сети случайным образом предъявляется объект  $n_k$ . Вычислить расстояния  $d_{kj}$  между вектором  $n_k$  и всеми нейронами слоя  $W_j, j = 1, 2, \dots, m$ , например, Евклидово расстояние

$$d_{kj} = \left[ \sum_{i=1}^K (x_{ki} - w_{ji})^2 \right]^{\frac{1}{2}}. \quad (6.3)$$

3. Определить нейрон, для которого расстояние  $d_{kj}$  минимально. Выполнить коррекцию его весового вектора на величину  $h \cdot (n_k - W_j^{(k-1)})$ .

4. Проверить условие остановки. Если превышено максимальное количество итераций  $k_{max}$ , то алгоритм завершает работу.

Иначе  $k = k + 1$  и перейти к п. 2.

### **Порядок выполнения**

1. Загрузить в память компьютера исходные данные вашего варианта (табл. 6.1) и отобразить их на экране. Построить графическое изображение экспериментальных данных (диаграмму рассеяния). Визуально оценить число кластеров  $m$  по построенному изображению.

2. Разработать и программно реализовать алгоритм слоя Кохонена, задав количество нейронов равным предполагаемому количеству кластеров.

3. Выполнить кластерный анализ исходных данных с использованием разработанного алгоритма. Определить наиболее оптимальное число нейронов  $m$ . Исследовать влияние параметра обучения  $h$  на точность кластерного анализа.

4. Классифицировать исходные объекты в кластеры, исходя из принципа: объект относится к тому кластеру, расстояние до нейрона которого оказалось минимальным.

5. Отобразить графически найденные кластеры и их центры.

**Форма отчета:** работающий программный модуль, реализующий задания, тексты программ. Результаты кластеризации алгоритмом нейронная сеть слой Кохонена.

*Таблица 6.1*

**Варианты заданий**

| Вариант | Данные     |
|---------|------------|
| 1       | data1.txt  |
| 2       | data2.txt  |
| 3       | data3.txt  |
| 4       | data4.txt  |
| 5       | data5.txt  |
| 6       | data6.txt  |
| 7       | data7.txt  |
| 8       | data8.txt  |
| 9       | data9.txt  |
| 10      | data10.txt |
| 11      | data11.txt |
| 12      | data12.txt |

### **Контрольные вопросы**

1. Для каких задач обработки экспериментальных данных используются нейронные сети?

2. Что такое искусственный нейрон?

3. Объясните понятия: синапсы, ядро, аксон нейрона.

4. В чем суть алгоритма слоя Кохонена?

5. Опишите этапы алгоритма слоя Кохонена.

6. Каким образом выбрать размер слоя Кохонена?

7. Какие критерии остановки алгоритма можно использовать?

8. Влияет ли выбор начальных приближений для параметров нейронов на точность кластеризации данных?

9. Влияет ли выбор параметра обучения  $h$  на точность кластерного анализа?

# ЛАБОРАТОРНАЯ РАБОТА № 7

## МЕТОД ОПОРНЫХ ВЕКТОРОВ

**Цель работы.** Решение задачи классификации данных на основе метода (машины) опорных векторов и практическое использование R функций сторонних пакетов.

### КРАТКИЕ СВЕДЕНИЯ

Метод опорных векторов (*support vector machines – SVM*) относится к методам обучения с учителем и предназначен для нахождения оптимальных в некотором смысле функций классификации данных (решающих функций). Метод наиболее эффективен при разделении двух классов.

**Линейно разделимый случай.** Рассмотрим задачу нахождения наилучшего в некотором смысле линейного разделения множества объектов  $\{n_1, n_2, \dots, n_N\}$  на два класса  $C_1$  и  $C_2$ . Каждый объект  $n_i = (x_{i1}, x_{i2}, \dots, x_{iK})$ ,  $i = 1, 2, \dots, K$ , характеризуется набором из  $K$  признаков. Есть обучающая выборка  $A = \{n_1, n_2, \dots, n_L\}$ , где  $L < N$ , и  $Y = \{y_1, y_2, \dots, y_L\}$  – множество меток классов  $C_1$  и  $C_2$  для объектов обучающей выборки  $A$ . Требуется по обучающей выборке  $A$  построить линейную функцию  $f(n_i)$ , которая удовлетворяла бы условию

$$\begin{aligned} f(n_i) &> 0 \quad \text{для } n_i \in C_1, \\ f(n_i) &< 0 \quad \text{для } n_i \in C_2. \end{aligned} \quad (7.1)$$

Можно полагать, что метки равны

$$y_i = \begin{cases} 1, & x_i \in C_1 \\ -1, & x_i \in C_2 \end{cases}, \quad (7.2)$$

тогда поставленную задачу можно переформулировать

$$y_i f(n_i) > 0. \quad (7.3)$$

Умножая, если нужно функцию  $f$  на некоторое положительное число, нетрудно видеть, что система неравенств (7.3) равносильна системе

$$y_i f(n_i) > 1, \quad \text{для всех } n_i \in A. \quad (7.4)$$

С учетом того, что  $f(n_i)$  – линейная функция, то система неравенств примет вид

$$y_i((w \cdot n_i) + b) \geq 1, \quad i = 1, 2, \dots, L, \quad (7.5)$$

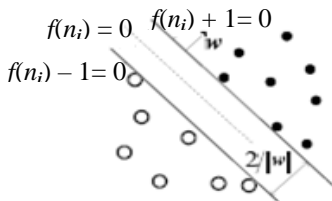


Рис. 7.1. Разделяющая полоса

где  $\mathbf{w}$  (или  $\vec{w}$ ) – вектор весовых коэффициентов,  $b$  – некоторое число. Тогда разделяющей два класса гиперплоскостью будет  $(\mathbf{w} \cdot \mathbf{n}_i) + b = 0$ .

Нетрудно видеть, что и все гиперплоскости вида  $(\mathbf{w} \cdot \mathbf{n}_i) + b' = 0$ ,  $b' \in (b-1, b+1)$  также будут разделяющими (см. рис. 7.1). Расстояние между граничными гиперплоскостями  $(\mathbf{w} \cdot \mathbf{n}_i) + b - 1 = 0$  и  $(\mathbf{w} \cdot \mathbf{n}_i) + b + 1 = 0$  равно  $2 / \|\mathbf{w}\|$ , где  $\|\mathbf{w}\| = (w_1^2 + w_2^2 + \dots + w_K^2)^{1/2}$  – Евклидова норма вектора  $\mathbf{w}$ .

*Опорными векторами* (не менее двух) называются обучающие вектора, находящиеся на граничных гиперплоскостях.

Для надежного разделения объектов классов  $C_1$  и  $C_2$  необходимо чтобы расстояние между разделяющимися гиперплоскостями было как можно больше, т.е.  $\|\mathbf{w}\|$  как можно меньше. Тогда задача нахождения классифицирующей функции  $f(\mathbf{n}_i)$  по параметрам  $\mathbf{w}$  и  $b$  сводится к нахождению минимума квадратичного функционала  $(\mathbf{w} \cdot \mathbf{w})/2$  в выпуклом многограннике, задаваемом системой неравенств (7.5). Данная задача является задачей квадратичной оптимизации, решается математически и равносильна поиску седловой точки лагранжиана

$$\Phi(\vec{w}, b, \Lambda) = \frac{\vec{w} \cdot \vec{w}}{2} - \sum_{i=1}^L \lambda_i (y_i (\vec{w} \cdot \mathbf{n}_i + b) - 1) \rightarrow \min_{\vec{w}, b} \max_{\Lambda} \quad (7.6)$$

по множителям Лагранжа  $\Lambda = \{\lambda_1, \lambda_2, \dots, \lambda_L\}$ ,  $0 \leq \lambda_i \leq C$ , при условии, что  $\lambda_i (y_i (\mathbf{w} \cdot \mathbf{n}_i + b) - 1) = 0$ . Константа  $C$  задает соотношение между плоскостью функции  $f(\mathbf{n}_i)$  и допустимым значением нарушения границы  $\varepsilon$ . Параметр  $C$  выбирается вручную для каждой ситуации отдельно.

Из необходимых условий существования седловой точки следует, что  $\mathbf{w}$  следует искать в виде

$$\vec{w} = \sum_{i=1}^L \lambda_i y_i \mathbf{n}_i, \quad \text{причем} \quad \sum_{i=1}^L \lambda_i y_i = 0. \quad (7.7)$$

Параметр  $b$  равен

$$b = y_s^{-1} - \mathbf{w} \cdot \mathbf{n}_s \quad (7.8)$$

где  $\mathbf{n}_s$  – любой из опорных векторов. С учетом подстановки выражений (7.7) и (7.8) в (7.6) лагранжиан примет вид

$$\Phi(\Lambda) = \sum_{i=1}^L \lambda_i - \frac{1}{2} \sum_{i=1}^L \sum_{j=1}^L \lambda_i \lambda_j y_i y_j (\mathbf{n}_i \cdot \mathbf{n}_j) \quad (7.9)$$

Для нахождения множителей Лагранжа  $\{\lambda_1^*, \lambda_2^*, \dots, \lambda_L^*\}$  решается задача поиска максимума лагранжиана при условии  $\sum_{i=1}^L \lambda_i y_i = 0$  и  $0 \leq \lambda_i \leq C$

с использованием методов оптимизации.

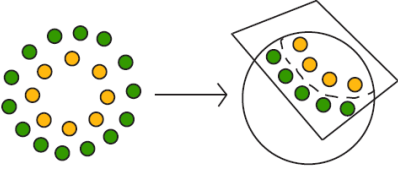
Таким образом, решением задачи (7.6) является функция вида



$$f(n_i) = \sum_{j=1}^L \lambda_j^* y_j (n_j \cdot n_i) + y_s^{-1} - \sum_{j=1}^L \lambda_j^* y_j (n_j \cdot n_s), \quad (7.10)$$

где  $(n_s, y_s)$  – один из опорных векторов.

**Линейно неразделимый случай.** Метод опорных векторов также используется для нелинейного разделения классов. В этом случае необходимо вложить пространство  $A$  в пространство  $H$  большей размерности с помощью отображения  $\varphi: A \rightarrow H$ .



Для этого необходимо скалярное произведение двух векторов  $n_j \cdot n_i$  заменить на скалярное произведение двух преобразованных векторов

$$K(n_j, n_i) = \varphi(n_j) \cdot \varphi(n_i). \quad (7.11)$$

Рис. 7.2. Пример перехода к расширенному пространству

Функция  $K(n_j, n_i)$  – называется ядром. Ядро гарантирует разделение  $L$  векторов на любые два класса. В данном случае решением задачи (7.6) является функция вида

$$f(n_i) = \sum_{j=1}^L \lambda_j^* y_j K(n_j, n_i) + y_s^{-1} - \sum_{j=1}^L \lambda_j^* y_j K(n_j, n_s), \quad (7.12)$$

где  $(n_s, y_s)$  – один из опорных векторов. Основные виды функций классификации, применяемые в методе опорных векторов следующие:

1. линейная

$$K(n_j, n_i) = n_j \cdot n_i; \quad (7.13)$$

2. полиномиальная степени  $d$

$$K(n_j, n_i) = (\gamma(n_j \cdot n_i) + \beta)^d; \quad (7.14)$$

где  $\gamma$  и  $\beta$  – некоторые константы;

3. радиальная базовая функция

$$K(n_j, n_i) = e^{-\gamma \|n_j - n_i\|^2}. \quad (7.15)$$

Стоит отметить, что не существует общего подхода к автоматическому выбору ядра в случае линейной неразделимости классов.

## Программная реализация алгоритма в R проектах

Различные варианты алгоритма метода опорных векторов реализованы в R пакетах `e1071`, `kernlab`, `klaR`, `svmpath`, `shogun`. Первая и наиболее интуитивная версия алгоритма была реализована в пакете `e1071` в виде функции `svm`. Вызов функции:

```
svm(formula,data,scale,type,kernel,degree,gamma,coef0, cost,...)
```

где `formula` – выражение, слева от знака “~” указывается имя колонки данных обучающей выборки, содержащей метки классов, справа – колонки признаков объектов, по которым будет проводиться обучение; `data` – набор данных обучающей выборки (Data Frame), для решения задачи классификации колонка данных, содержащая метки классов, должна иметь тип данных – `factor` (табл.7.1); `scale` – логический вектор, указывает набор признаков для стандартизации; `type` – указывает тип анализа, для решения задачи классификации необходимо использовать “C-classification”; `kernel` – указывает вид функции ядра (“linear”, “radial”, “polynomial”); `degree` – степень полинома  $d$  в формуле (7.14); `gamma` – параметр  $\gamma$  в формулах (7.14) и (7.15); `coef0` – параметр  $\beta$  в формуле (7.14); `cost` – константа  $C$  (см. пояснение к формуле (7.6)).

Функция `svm` возвращает объект класса `svm`, содержащий: `sv` – координаты опорных векторов; `index` – индексы опорных векторов в исходном наборе данных; `coefs` – оцененные значения множителей Лангранжа  $\{\lambda_1^*, \lambda_2^*, \dots, \lambda_L^*\}$ ; `rho` – отрицательное значение оцененного параметра  $b$  (см. формулу (7.5)).

Для визуализации результатов функции `svm` (в примере ниже – объект `clt`) используются функции `predict`, `table`, `plot`. Пример.

```
>> print(table(predict(clt), Training_data$V3)) # the confusion matrix
>> plot(clt, Training_data) # the palette scatter plot
```

### Порядок выполнения

1. С помощью функций `install.packages` и `library` установить и подключить пакет `e1071`. Для проверки правильности установки пакета в командной строке RStudio/R набрать `help(svm)` и нажать **Enter**.

2. С помощью команды `help` изучить основные функции пакета `e1071` для решения задач классификации: `svm`, `predict`, `plot`, `summary`.

3. Загрузить данные согласно вашему варианту (табл. 7.2). Построить графическое изображение экспериментальных данных (диаграмму рассеяния).

4. С помощью линейного метода опорных векторов разделить исходные данные на два класса. Определить опорные вектора, классифицирующую функцию (параметры  $w$  и  $b$ ), ошибку классификации (матрицу не-точностей – confusion matrix). Отобразить графически классифицированные объекты и классифицирующую функцию  $(w_1 \cdot x + w_2 \cdot y) + b = 0$  (использовать диаграмму рассеяния в цвете. См. пример на рис. 7.3).

5. С использованием нелинейного метода опорных векторов разделить исходные данные на два класса с учетом вида функции классификации согласно вашему варианту. Отобразить графически классифицированные объекты с помощью функции *plot* (см. пример на рис. 7.3).

**Форма отчета:** работающий программный модуль, реализующий задание, тексты программ. Результаты классификации данных методом опорных векторов. Выводы по результатам обработки экспериментальных данных.

Таблица 7.1

Пример варианта исходных данных для классификации

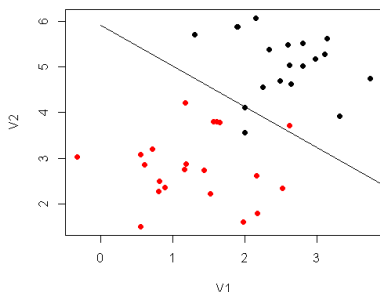
| Объект | V1<br>(Признак 1) | V2<br>(Признак 2) | V3<br>(Класс) |
|--------|-------------------|-------------------|---------------|
| 1      | 1                 | 1                 | -1            |
| 2      | 3                 | 3                 | 1             |
| 3      | 1                 | 3                 | 1             |

Таблица 7.2

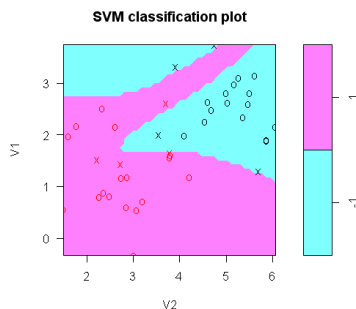
Варианты заданий

Ядра:  $1 - K(u, v) = (\gamma(u \cdot v) + \beta)^d$ ,  $2 - K(u, v) = e^{-\gamma \|u - v\|^2}$ . Параметр  $C = 10$

| Вариант | Ядро | Параметры ядра                 | Данные     |
|---------|------|--------------------------------|------------|
| 1       | 1    | $\gamma = 1, \beta = 1, d = 2$ | data1.txt  |
| 2       | 2    | $\gamma = 1$                   | data2.txt  |
| 3       | 1    | $\gamma = 1, \beta = 1, d = 3$ | data3.txt  |
| 4       | 2    | $\gamma = 2$                   | data4.txt  |
| 5       | 1    | $\gamma = 1, \beta = 1, d = 3$ | data5.txt  |
| 6       | 2    | $\gamma = 2$                   | data6.txt  |
| 7       | 1    | $\gamma = 1, \beta = 1, d = 2$ | data7.txt  |
| 8       | 2    | $\gamma = 4$                   | data8.txt  |
| 9       | 1    | $\gamma = 1, \beta = 1, d = 4$ | data9.txt  |
| 10      | 2    | $\gamma = 1$                   | data10.txt |
| 11      | 1    | $\gamma = 1, \beta = 1, d = 5$ | data11.txt |
| 12      | 2    | $\gamma = 2$                   | data12.txt |



Линейно разделимый случай



Линейно неразделимый случай

Рис. 7.3. Примеры графического отображения классифицированных объектов

### Контрольные вопросы

1. Для каких задач обработки экспериментальных данных используется метод опорных векторов?
2. Для решения какой задачи метод опорных векторов наиболее эффективен?
3. В чем состоит суть метода опорных векторов?
4. Что представляют собой опорные векторы?
5. Запишите выражение классифицирующей функции для линейно разделимого случая.
6. Запишите выражение классифицирующей функции для линейно неразделимого случая.
7. В каких R пакетах реализованы алгоритмы метода опорных векторов?
8. Укажите список аргументов функции *svm* пакета *e1071*.
9. Перечислите основные атрибуты объекта класса *svm*, возвращаемого функцией *svm*.
10. Улучшилось ли разделение исходных данных на два класса в результате применения нелинейной классификации?

## ЛАБОРАТОРНАЯ РАБОТА № 8

### ГЛОБАЛЬНОЕ ВЫРАВНИВАНИЕ ПАР НУКЛЕОТИДНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ. АЛГОРИТМ НИДЛМАНА-ВУНША

**Цель работы.** Изучение алгоритма Нидлмана-Вунша для выполнения глобального выравнивания двух нуклеотидных последовательностей с целью установления их сходства.

#### КРАТКИЕ СВЕДЕНИЯ

В биоинформатике часто исследуется задача сравнения двух нуклеотидных или аминокислотных последовательностей (строк символов). Сравнение (и последующее выравнивание) двух строк в контексте биоинформатики осложняется тем, что практически никогда не бывает полного совпадения (или вхождения) одной строки с другой – строки почти всегда различаются. Причины различны: экспериментальные ошибки, генные мутации (не существует двух одинаковых геномов – человек имеет множество коротких синонимных мутаций и небольшое количество несинонимных мутаций), действие эволюции (если сравниваются последовательности разных организмов). Сравнение двух строк получило название *выравнивание* (alignment), так как строки как бы прикладываются друг к другу максимально правдоподобным способом, при этом появляются промежутки (gaps), вставки (insertions) и удаления (deletions).

*Глобальное выравнивание* – это полное выравнивание одной последовательности относительно другой последовательности.

*Локальное выравнивание* – это поиск части одной последовательности, которая совпадает с частью другой последовательности.

Решение задачи оптимального глобального выравнивания прямым перебором, т.е. всевозможным прикладыванием одной строки к другой, – трудновыполнимо, из-за огромного количества вариантов. Используется специальная техника программирования – *динамическое программирование*. Идея динамического программирования состоит в разбиении сложной задачи сравнения двух строк на простые подзадачи. Динамическое программирование позволяет сократить время выполнения вычислений с экспоненциального времени  $O(e^n)$  до  $O(n^2)/O(n^3)$  ( $n$  – средняя длина сравниваемых строк последовательностей). Примером приложения динамиче-

ского программирования к сравнению биологических последовательностей является алгоритм Нидлмана-Вунша.

В алгоритме Нидлмана-Вунша используются матрица *замен* (нуклеотидов)  $S$  и *точечная матрица сходства*  $F$ .

Матрица *замен*  $S$  содержит счета (scores), характеризующие похожесть символов сравниваемых последовательностей (табл. 8.1). Элементы матрицы определяют счета или штрафы, назначаемые заменам символов в выравниваемых последовательностях.

Таблица 8.1

**Пример матрицы замен  $S$**

За совпадение символов назначается 1, за несовпадение – -1

|   | A  | G  | C  | T  |
|---|----|----|----|----|
| A | 1  | -1 | -1 | -1 |
| G | -1 | 1  | -1 | -1 |
| C | -1 | -1 | 1  | -1 |
| T | -1 | -1 | -1 | 1  |

Точечная матрица сходства  $F$  – это простейшее изображение, дающее представление о сходстве между двумя символьными последовательностями. Строки соответствуют символам одной последовательности, колонки – символам другой последовательности. В простейшем случае позиции в матрице оставляются пустыми, если символы различны, и заполняются, если они совпадают. Совпадающие фрагменты последовательностей отобразятся в виде диагоналей, идущих из верхнего левого угла в нижний правый.

Алгоритм Нидлмана-Вунша реализуется в два этапа. На первом этапе выполняется создание и заполнение точечной матрицы сходства  $F$ . На втором этапе осуществляется нахождение наилучшего или оптимального выравнивания путем обратного прохода из правого нижнего угла в левый верхний угол точечной матрицы.

**Алгоритм**

*Шаг 1.* Инициализация. Установить штраф за удаление  $d$ , создать и заполнить матрицу *замен*  $S$ , создать точечную матрицу  $F$ .

*Шаг 2.* Заполнить матрицу сходства  $F$ .

Заполнить первые колонку и столбец матрицы увеличивающимися штрафами за удаление с шагом  $i \cdot d$ , где  $i = 0, 1, \dots, n$ ,  $n$  – число символов в

первой или второй строках последовательностей. Например, для  $d = -1$  фрагмент матрицы  $F$  примет вид

|          |           |           |           |           |           |           |
|----------|-----------|-----------|-----------|-----------|-----------|-----------|
|          |           | <b>G</b>  | <b>C</b>  | <b>A</b>  | <b>T</b>  | <b>G</b>  |
|          | <b>0</b>  | <b>-1</b> | <b>-2</b> | <b>-3</b> | <b>-4</b> | <b>-5</b> |
| <b>G</b> | <b>-1</b> |           |           |           |           |           |
| <b>A</b> | <b>-2</b> |           |           |           |           |           |
| <b>T</b> | <b>-3</b> |           |           |           |           |           |

Заполнить ячейки матрицы  $F$  по рекуррентной формуле

$$F_{ij} = \max(F_{i-1,j-1} + S(A_i, B_j), F_{i,j-1} + d, F_{i-1,j} + d), \quad (8.1)$$

где  $A_i$  и  $B_j$  – символы в  $i$ -ой строке и  $j$ -ой колонках матрицы  $F$  соответственно. В результате заполнения матрицы  $F$  нижняя правая ячейка содержит максимальный *счет* среди всех возможных выравниваний и является оценкой качества выравнивания.

*Шаг 3.* Определить наилучшее выравнивание последовательностей путем обратного прохода из правого нижнего угла в левый верхний угол точечной матрицы  $F$ . Для построения оптимального выравнивания необходимо восстановить кратчайший путь к максимальному счету в правом нижнем углу матрицы.

Проход производится из правой нижней ячейки матрицы  $F$ .

Проверить, из какой предыдущей ячейки таблицы вычислено текущее значение (слева, сверху или назад по диагонали). Перейти к той ячейке, значение счета которой являлось максимальным до перехода в текущую ячейку. В случае наличия нескольких ячеек с одинаковыми значениями счетов, одна из которых диагональная, – перейти в ячейку назад по диагонали. При переходе в ячейку по диагонали – в выражения сравниваемых последовательностей записываются текущие обозначения символов строки и колонки, при переходе в ячейку слева – записывается символ вставки «–» во вторую последовательность, при переходе в ячейку сверху – записывается символ вставки «–» в первую последовательность,

Последовательно осуществить проход по ячейкам таблицы путем изменения индексов  $i$  и  $j$ , добавляя очередную букву или символ «–» в новые элементы выровненных строк.

Если одна из строк короче другой, то добавить символ(ы) «–» до приведения в соответствие длин двух строк.

Пример глобального выравнивания двух последовательностей GAATTC и GATTA представлен на рис. 8.1.

|   |     | G  | A  | A  | T  | T   | C   |
|---|-----|----|----|----|----|-----|-----|
|   | 0   | -2 | -4 | -6 | -8 | -10 | -12 |
| G | -2  | 1  | -1 | -3 | -5 | -7  | -9  |
| A | -4  | -1 | 2  | 0  | -2 | -4  | -6  |
| T | -6  | -3 | 0  | 1  | 1  | -1  | -3  |
| T | -8  | -5 | -2 | -1 | 2  | 2   | 0   |
| A | -10 | -7 | -4 | -1 | 0  | 1   | 1   |

Рис. 8.1. Точечная матрица выравнивания последовательностей GAATTC и GATTA. Стрелками обозначен маршрут обратного прохода. Штраф за удаление – 2, за совпадение символов – 1, за несовпадение – -1.

В результате применения алгоритма Нидлмана-Вунша оптимальное выравнивание последовательностей имеет вид

GAATTC  
G –ATTA

### Порядок выполнения

1. Изучить функции R: *diag*, *rownames*, *colnames*, *strsplit*, *unlist*, *length*, *paste*.
2. Разработать алгоритм глобального выравнивания Нидлмана-Вунша.
3. Загрузить в RStudio файл *GlobalAlignm\_Lab.R* и изучить содержимое файла. Файл содержит шаблон программного кода для выполнения лабораторной работы.
4. Программно реализовать алгоритм Нидлмана-Вунша.
5. Выполнить выравнивание строк нуклеотидных последовательностей согласно вашему варианту (табл. 8.2), назначая штраф за удаление – -1, за совпадение символов – 1, за несовпадение – -1.
6. Исследуйте влияние штрафа за удаление, счетов за совпадение и несовпадение символов на точность глобального выравнивания последовательностей.

**Форма отчета:** работающий программный модуль, реализующий задание, тексты программ. Результаты работы алгоритма Нидлмана-Вунша. Выводы по результатам выполнения лабораторной работы.



Таблица 8.2

## Варианты заданий

| Вариант | ДНК<br>последовательность 1 | ДНК<br>последовательность 2 |
|---------|-----------------------------|-----------------------------|
| 1       | AAATGACC                    | AAAAAATGACCA                |
| 2       | CCCTATATA                   | TACCCTATATAACC              |
| 3       | TAAATCTGCC                  | TATAAATCTGCCCC              |
| 4       | TTATATAT                    | CTTAATATA                   |
| 5       | CACTGAA                     | CCACTTTGAAA                 |
| 6       | CTGAATAA                    | CCTGAAAATAAC                |
| 7       | TTACGCGT                    | TTAGCGTA                    |
| 8       | ATCGCGTTT                   | CATCGCCCCGTTTA              |
| 9       | CCGTTATT                    | CGGCCTCAAT                  |
| 10      | GCGTTGTT                    | CGGCGTGGATT                 |
| 11      | TTATTAAAT                   | CTGCCATAT                   |
| 12      | TACCATACC                   | CCCTATATA                   |

## Контрольные вопросы

1. Что называется выравниванием биологических последовательностей?
2. Что такое глобальное выравнивание последовательностей?
3. Что такое локальное выравнивание последовательностей?
4. В чем заключается задача выравнивания последовательностей?
5. Что такое матрица замен и для чего её используют?
6. Что такое точечная матрица сходства и для чего её используют?
7. Запишите однобуквенные обозначения нуклеотидов.
8. Что такое оптимальное выравнивание и чем оно отличается от неоптимального?
9. В чем суть алгоритма Нидлмана-Вунша?
10. Причислите основные этапы работы алгоритма Нидлмана-Вунша?
11. Для каких целей используются функции R: *diag*, *rownames*, *colnames*, *strsplit*, *unlist*, *length*, *paste*?

## СПИСОК ЛИТЕРАТУРЫ

1. Введение в биоинформатику / А. Леск; пер. с англ. – М.: БИНОМ. Лаборатория знаний, 2009. – 318 с.
2. Основы биоинформатики: учеб. пособие / А.Н. Огурцов. - Х.: НТУ «ХПИ», 2013. – 400 с.
3. Технологии анализа данных: Data Mining, Visual Mining, Text Mining, OLAP / А. А. Барсегян, М. С. Куприянов, В. В. Степаненко, И. И. Холод. – 2-е изд., перераб. и доп. – СПб: БХВ-Санкт-Петербург, 2007. – 384 с.
4. Наглядная математическая статистика: учеб. пособие / Б. М. Лагутин. – М.: БИНОМ. Лаборатория знаний, 2007. – 472 с.
5. Кабаков, Р. И. R в действии. Анализ и визуализация данных в программе R / Р. И. Кабаков; пер. с англ. – М.: ДМК Пресс, 2014. – 588 с.
6. Яцков, Н. Н. Интеллектуальный анализ данных : пособие / Н. Н. Яцков. – Минск : БГУ, 2014. – 151 с.

## СОДЕРЖАНИЕ

|  |    |
|--|----|
| ВВЕДЕНИЕ .....   | 3  |
| ЛАБОРАТОРНАЯ РАБОТА № 1. ОСНОВЫ РАБОТЫ В R .....   | 4  |
| ЛАБОРАТОРНАЯ РАБОТА № 2. ПРЕДВАРИТЕЛЬНЫЙ АНАЛИЗ<br>БОЛЬШИХ НАБОРОВ БИОЛОГИЧЕСКИХ ДАННЫХ .....                              | 14 |
| ЛАБОРАТОРНАЯ РАБОТА № 3. МЕТОД ГЛАВНЫХ КОМПОНЕНТ   | 19 |
| ЛАБОРАТОРНАЯ РАБОТА № 4. ИЕРАРХИЧЕСКИЕ МЕТОДЫ<br>КЛАСТЕРНОГО АНАЛИЗА .....   | 24 |
| ЛАБОРАТОРНАЯ РАБОТА № 5. НЕИЕРАРХИЧЕСКИЕ МЕТОДЫ<br>КЛАСТЕРНОГО АНАЛИЗА .....   | 30 |
| ЛАБОРАТОРНАЯ РАБОТА № 6. НЕЙРОННЫЕ СЕТИ. СЛОЙ<br>КОХОНЕНА.....   | 35 |
| ЛАБОРАТОРНАЯ РАБОТА № 7. МЕТОД ОПОРНЫХ ВЕКТОРОВ .....  | 39 |
| ЛАБОРАТОРНАЯ РАБОТА № 8. ГЛОБАЛЬНОЕ ВЫРАВНИВАНИЕ<br>ПАР НУКЛЕОТИДНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ. АЛГОРИТМ<br>НИДЛМАНА-ВУНША ..... | 45 |
| СПИСОК ЛИТЕРАТУРЫ .....  | 50 |

Учебное издание

**Яцков Николай Николаевич**  
**Юдин Александр Александрович**

## **ВВЕДЕНИЕ В БИОИНФОРМАТИКУ**

**Методические указания  
к лабораторным работам**

**Для студентов специальности  
1-31 03 07-02 «Прикладная информатика»**

В авторской редакции

Ответственные за выпуск *Н. Н. Яцков, А. А. Юдин*

Подписано в печать 16.01.2018. Формат 60×84/16. Бумага офсетная.  
Усл. печ. л. 3,02. Уч.-изд. л. 2,36. Тираж 50 экз. Заказ

Белорусский государственный университет.  
Свидетельство о государственной регистрации издателя, изготовителя,  
распространителя печатных изданий № 1/270 от 03.04.2014.  
Пр. Независимости, 4, 220030, Минск.

Отпечатано с оригинал-макета заказчика  
на копировально-множительной технике  
факультета радиофизики и компьютерных технологий  
Белорусского государственного университета.  
Ул. Курчатова, 5, 220064, Минск.