

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

Н. Н. Яцков, М. А. Сиколенко, М. К. Чепелева

ВВЕДЕНИЕ В БИОИНФОРМАТИКУ

**Методические указания
к лабораторным работам**

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

**МИНСК
2021**

УДК 575.112:004 (075.8)

ББК 28.04я73+32.97я73

Я93

Рекомендовано советом
факультета радиофизики и компьютерных технологий
29 декабря 2020 г., протокол № 4

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

Яцков, Н. Н.

Я93 Введение в биоинформатику : метод. указания к лаб. работам /
Н. Н. Яцков, М. А. Сиколенко, М.К. Чепелева. – Минск : БГУ, 2021.
– 51 с.

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

УДК 575.112:004 (075.8)

ББК 28.04я73+32.97я73

© Яцков Н. Н., М. А. Сиколенко, М.К. Чепелева, 2021

© БГУ, 2021

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 наборов биологических данных на примере использования среды R. Тематика практикума охватывает алгоритмы предварительного анализа больших наборов данных, графов Де Брюйна для сборки генома *de novo*, глобального и локального выравнивания пар нуклеотидных последовательностей, кластерного анализа, векторизации и визуализации молекул ДНК. Приведенные справочные теоретические сведения содержат достаточный объем информации, исключая необходимость прибегать к другим источникам литературы в ходе выполнения практических работ.

Прилагается компакт-диск, содержащий файлы заданий и данных 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 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 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

R-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 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 задать 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)
```

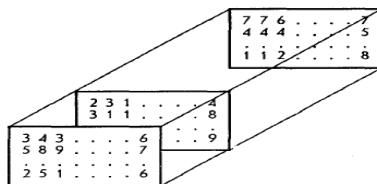


Рис. 1.2. Схематическое представление трехмерного массива

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

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

Функция *solve()* вычисляет решение \mathbf{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 ξ 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**.

Histogram of X

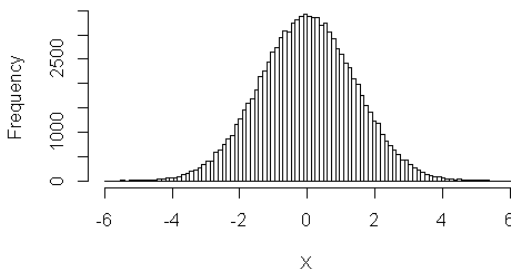


Рис. 1.3. Гистограмма нормальной случайной величины с параметрами $m=0$ и $D=2$, $N=10000$, число каналов гистограммы равно 40

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 г-файл с именем **write_txt.r**, 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.r`, 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. Каким образом удалить из оперативной памяти переменные, зарезервированные 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/см³).

«nitrate» (Нитраты) – уровень загрязнения воздуха нитратами в указанный день (мкг/см³).

«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  
  наблюдениям 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 (332 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. ID, pollutant	Задание 2. ID	Задание 3. 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

СБОРКА ГЕНОМА *DE NOVO*. МЕТОД ГРАФОВ ДЕ БРЮЙНА

Цель работы. Практическое освоение метода графов де Брюйна для сборки генома *de novo* по экспериментальным данным, полученным в результате геномного секвенирования.

КРАТКИЕ СВЕДЕНИЯ

Определение нуклеотидного состава молекул ДНК и РНК – ключевой вопрос для фундаментальных и прикладных исследований в медицине, биологии и биоинформатике. Экспериментальный метод определения последовательностей биополимеров нуклеиновых кислот, молекул ДНК или РНК, получил название *секвенирование* (от англ. sequencing). Прибор для выполнения геномного секвенирования называется *секвенатором*. Геномный секвенатор позволяет регистрировать фрагменты нуклеотидных последовательностей – прочтения или риды (от англ. read). Признанным стандартом в области секвенирования ДНК, для последующего получения полной последовательности генома, является технология секвенирования следующего поколения Illumina, основанная на секвенировании путем синтеза флуоресцентно окрашенных нуклеотидов и позволяющая получать последовательности прочтений (ридов) длиной не более 300 пар нуклеотидов. Качество регистрируемых прочтений достаточно высоко, что позволяет восстановить исходный геном целиком. При этом наименьшая известная длина генома организма бактерий обычно составляет 10^6 пар нуклеотидных оснований. Поэтому сборка генома весьма ресурсозатратная процедура.

Сборка генома — объединение большого числа ридов в одну или несколько длинных последовательностей, в результате чего должна быть сконструирована исходная последовательность генома. Рассматривают сборку генома на референс или *de novo*.

Сборка на референс — риды выравниваются на уже собранный геном того же или близкого вида.

Сборка de novo – на основе коротких прочтений осуществляется сборка полного генома. Возможности по изучению свойств геномов существенно ограничены, если полная последовательность генома неизвестна. Для сборки генома *de novo* наиболее распространенными подходами являются методы *Overlap layout consensus* и *графы де Брюйна*. В основе метода гра-

фов де Брюйна лежит поиск эйлера пути в графе, построенном по нуклеотидным прочтениям.

Эйлеров путь в графе – путь, проходящий через все рёбра графа, причём только один раз. В 1946 г. датский математик Н. де Брюйн предложил применение алгоритма нахождения эйлера цикла в решении задачи о сверхстроке – такой циклической строке¹ наименьшей длины, которая содержит всевозможные подстроки длины k (так называемые k -меры), сгенерированные из символов заданного алфавита. Например, для бинарного алфавита $\{0, 1\}$ при $k = 3$ множество всех возможных k -меров – $\{000, 001, 010, 011, 100, 101, 110, 111\}$, а искомая циклическая сверхстрока – 00011101.

Задача о нахождении *эйлерова пути в графе* схожа с задачей сборки генома (алфавит – множество $\{A, G, C, T\}$) с условием того, что искомая сверхстрока (собираемый геном) может быть, как циклической, так и линейной, и должна содержать не всевозможные k -меры, а лишь те, которые можно получить из имеющихся прочтений.

Введём определения префикса и суффикса, используемые в методе графов де Брюйна.

Префикс строки S – это её подстрока $S[1, n - 1]$, где n – длина строки S или количество символов в ней, [...] – оператор доступа к символам строки с 1 по $n - 1$ позиции.

Суффикс строки S – есть её подстрока $S[2, n]$. Например, префиксом строки АТТГС является строка АТТГ, а суффиксом – ТТГС.

Искомый геном будем считать циклическим, если найденный эйлеров путь является циклическим (т.е. начальная вершина на графе является также и конечной) и линейным в других случаях.

Алгоритм де Брюйна

Шаг 1. Разбить все имеющиеся прочтения на k -меры.

Шаг 2. Построить граф, такой, что его вершинами являются всевозможные уникальные префиксы и суффиксы имеющихся k -меров (далее ($k-1$)-меры).

Шаг 3. Соединить все вершины v_i и v_j направленными (от v_i к v_j) рёбрами в тех случаях, когда существует такой k -мер, для которого v_i является его префиксом, а v_j – суффиксом. Полученные рёбра будут соответствовать k -мерам.

Шаг 4. В построенном графе определить эйлеров путь.

¹ *Циклическая строка* длины n – строка, в которой n -й символ предшествует первому.

Шаг 5. Инициализировать искомым геном ($k-1$ -мером, соответствующим первой вершине найденного эйлерова пути.

Шаг 6. Проследовав по эйлерову пути, выполнить сборку генома, при посещении очередной вершины присоединяя к нему последний символ соответствующего ($k-1$ -мера.

Пример сборки генома по 6 прочтениям с использованием алгоритма графов де Брюйна представлен на рисунке 3.1.

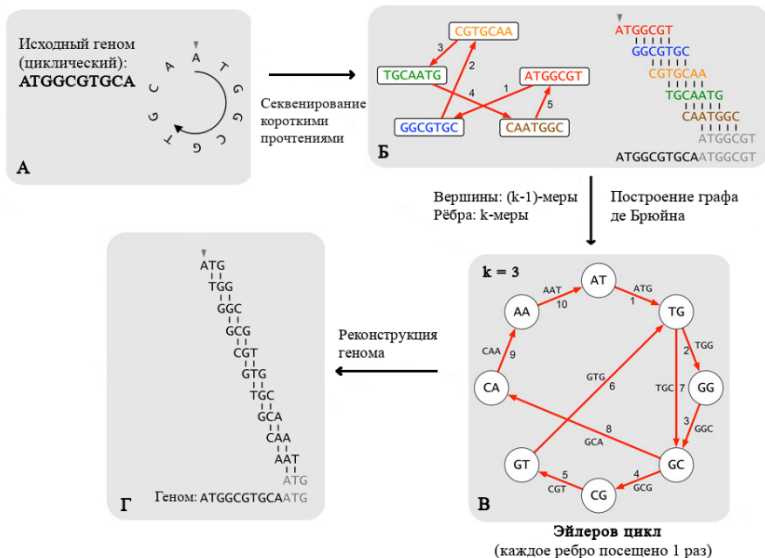


Рис. 3.1. Пример сборки циклического генома с использованием алгоритма графов де Брюйна (размер k -мера равен 3)

Следует отметить, что проход по ребрам является более предпочтительным, нежели алгоритм, в котором k -меры являются вершинами, поскольку в последнем случае требуется определить гамильтонов путь в графе (путь, проходящий через все вершины единожды), что является более трудоемкой вычислительно задачей.

Программная реализация графов в R

Для создания графов с помощью R можно воспользоваться функциями `graphNEL()` и `addEdge()` пакета `graph`.

```
# Создание объекта орграфа с вершинами {"A", "B", "C", "D"}:
> gr <- graphNEL(nodes=LETTERS[1:4], edgemode="directed")
# Соединение вершин рёбрами:
```

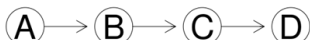
```
> gr <- addEdge(graph=gr, from=LETTERS[1:3], to=LETTERS[2:4])
```

Для нахождения эйлерова пути в графе используется функция *eulerian()* пакета *eulerian*:

```
> eul.path <- eulerian(gr)
> eul.path
[1] "A" "B" "C" "D"
```

Визуализация графов производится с помощью функции *plot* пакета *Rgraphviz*:

```
> plot(gr, 'circo')
```



Для установки вышеуказанных пакетов необходимо в среде R выполнить следующие команды:

```
> if (!requireNamespace("BiocManager", quietly = TRUE))
  install.packages("BiocManager")
> BiocManager::install("graph")
> install.packages("eulerian")
> BiocManager::install("Rgraphviz") # для визуализации графов
```

Порядок выполнения

1. Установить и подключить пакеты *graph* и *eulerian*. Для проверки правильности установки пакетов в командной строке RStudio/R выполнить коды рассмотренных примеров.
2. С помощью *help* изучить функции *paste*, *substr*, *nchar*, *unique*, *graphNEL*, *addEdge*, *eulerian*.
3. Загрузить данные согласно варианту (таблица 3.1)
4. Разработать алгоритм метода графов де Брюйна для сборки генома *de novo* и программно его реализовать в среде R, используя вспомогательные функции п. 2.
5. Выполнить сборку генома из заданных прочтений, установив значение *k* согласно вашему варианту.

Форма отчета: работающий программный модуль, реализующий задание, тексты программ. Результаты работы алгоритма графов де Брюйна сборки генома. Выводы по результатам обработки экспериментальных данных.

Таблица 3.1.

Варианты заданий

Вариант	k	Набор прочтений в виде вектора среды R
1	5	c("GCCACCG", "TGAGGCC", "AGGCCAC", "CCACCGC", "GAGGCCA", "GGCCACC", "ACCGCGA")
2	7	c("CTACCGTCT", "ACCGTCTGA", "CGCTACCGT", "GCTACCGTC", "GTCTGAAGA", "TACCGTCTG", "TCTGAAGAA")
3	5	c("AATGCACC", "GCACCCGC", "GAATGCAC", "ATGCACCC", "TGCACCCG", "CCGCGAAT", "GCGAATGC", "CCCGCGAA")
4	5	c("CTCAACG", "AACTCAA", "TCAACGT", "ACTCAAC", "AACGTAA", "CGTAACT")
5	7	c("CTACCGGC", "ACCGGCGC", "CGACTACC", "TACCGGCG", "GACTACCG", "ACTACCGG")
6	7	c("AGAAGGCGC", "TGAGAAGGC", "AGGCGCATG", "GTGAGAAGG", "GAGAAGGCG", "GAAGGCGCA")
7	6	c("TATCCGC", "ATATCCG", "TCCCAA", "ATCCGA", "CAAATAT", "CGCAAAT", "AAATATC")
8	5	c("GAGCGCC", "CGACGCC", "ACGCCCT", "CGCCCTT", "CCCTTCG")
9	7	c("TGCTCATAG", "GCTCATAGA", "CCTGCTCAT", "GACCTGCTC", "ACCTGCTCA", "TAGAGACCT", "ATAGAGACC", "GAGACCTGC", "TCATAGAGA")
10	5	c("CCAAGGT", "CCCAAGG", "CAAGGTG", "AAGGTGA", "GTGACCC", "GGTGACC", "GACCCAA")
11	7	c("CCCGAGGGGC", "AGGGGCCGAT", "GCCCGAGGGG", "GAGGGGCCGA")
12	5	c("CCAGAGG", "CAGAGGC", "GGCATGC", "AGGCATG")

Контрольные вопросы

1. Объясните термины *секвенирование*, *секвенатор* и *прочтение*.
2. Что такое *сборка генома*, *сборка генома на референс* и *de novo*.
3. Для какой задачи обработки экспериментальных данных используется метод графов де Брюйна?
4. Объясните понятие *эйлеров путь в графе*.
5. Что такое *префикс* и *суффикс* строки S ?
6. Объясните понятие циклической *сверхстроки*.
7. В чем состоит суть метода графов де Брюйна?
8. Перечислите основные шаги алгоритма метода графов де Брюйна.
9. Исследуйте влияние параметра k на точность и скорость сборки генома.
10. Для каких целей используются функции R и специализированных пакетов: *paste*, *substr*, *nchar*, *unique*, *graphNEL*, *addEdge*, *eulerian*?

ЛАБОРАТОРНАЯ РАБОТА № 4

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 динамического

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. 3.1). Элементы матрицы определяют счета или штрафы, назначаемые заменам символов в выравниваемых последовательностях.

Таблица 4.1

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 Нидлмана-Вунша

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

		G	C	A	T	G
	0	-1	-2	-3	-4	-5
G	-1					
A	-2					
T	-3					

Заполнить ячейки матрицы F по рекуррентной формуле

$$F_{ij} = \max(F_{i-1,j-1} + S(A_i, B_j), F_{i,j-1} + d, F_{i-1,j} + d), \quad (4.1)$$

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. 4.1.

		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. 4.1. 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. 4.2), 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 4.2

Варианты заданий

Вариант	ДНК последовательность 1	ДНК последовательность 2
1	AAATGACC	AAAAAATGACCA
2	CCCTATATA	TACCCTATATACC
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*?

ЛАБОРАТОРНАЯ РАБОТА №5

ЛОКАЛЬНОЕ ВЫРАВНИВАНИЕ ПАР НУКЛЕОТИДНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ. АЛГОРИТМ СМИТА-УОТЕРМАНА

Цель работы. Изучение алгоритма Смита-Уотермана локального выравнивания двух нуклеотидных последовательностей для установления их сходства. Знакомство с основными мерами оценки значимости локального выравнивания.

КРАТКИЕ СВЕДЕНИЯ

Использование глобального выравнивания для поиска сходства между биологическими последовательностями не всегда целесообразно. Применение глобального выравнивания эффективно, когда сравниваемые последовательности схожи друг с другом на всей своей протяжённости и их размеры отличаются незначительно. Если же наблюдается сходство лишь некоторых участков сравниваемых последовательностей, либо длины последовательностей существенно различаются, применяют алгоритмы локального выравнивания.

Проиллюстрировать вышесказанное можно на следующем примере. Необходимо установить сходство между двумя последовательностями:

1: ACTTGTGAA

2: ACAGTCGCAACTGTGAAAAGTAGT

Выравнивание с использованием алгоритма Нидлмана-Вунша (балл за совпадение символов – 1, штраф за несовпадение – -1, штраф за пропуск – -1) приводит к результату:

AC - - T - - - - TGTG - - -A - - A - -

ACAGTCGCAACTGTGAAAAGTAGT

Несмотря на то, что алгоритм отработал верно – найдено взаимное расположение последовательностей, максимизирующее балл выравнивания – интерпретируемость результата минимальна. Более того, простота примера позволяет заметить, что в центральной части второй последовательности расположен фрагмент, схожий с первой последовательностью:

- - - - - ACTTGTGAA - - - - -

ACAGTCGCAAC -TGTGAAAAGTAGT

Сходство сравниваемых последовательностей имеет локальный характер, что требует применения алгоритмов локального выравнивания. Выравнивание в примере произведено с помощью алгоритма Смита-

Уотермана (балл за совпадение символов – 1, штраф за несовпадение – -1, штраф за пропуск – -1).

Рассматривать алгоритм Смита-Уотермана удобно в сравнении с уже изученным алгоритмом Нидлмана-Вунша. Общими элементами алгоритмов являются динамическое программирование, матрица замен S и точечная матрица сходства F . Основные различия состоят в способах заполнения точечной матрицы F и вариантах обратного прохода по матрице, определяющих наилучшие локальные совпадения или области выравнивания последовательностей.

Алгоритм Смита-Уотермана

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

Шаг 2. Заполнить матрицу сходства F . Внести в первые колонку и столбец матрицы нули. Заполнить оставшиеся ячейки матрицы F по рекуррентной формуле

$$F_{ij} = \max(0, F_{i-1,j-1} + S(A_i, B_j), F_{i,j-1} + d, F_{i-1,j} + d), \quad (5.1)$$

где A_i и B_j – символы в i -ой строке и j -ой колонках матрицы F соответственно. Отметим, что матрица F для алгоритма Смита-Уотермана, в отличие от аналогичной для алгоритма Нидлмана-Вунша, не содержит отрицательных значений.

Шаг 3. Определить наилучшее выравнивание последовательностей. Последовательность действий при обратном проходе в алгоритме Смита-Уотермана совпадает с алгоритмом Нидлмана-Вунша, за двумя исключениями: 1) проход начинается не из правой нижней ячейки матрицы F , а из ячейки, содержащей максимальное по всей матрице значение (являющееся одновременно и баллом выравнивания); 2) проход заканчивается, как только значение в ячейке обратного прохода равно нулю.

Пример матрицы выравнивания двух последовательностей с использованием алгоритма Смита-Уотермана представлен на рисунке 5.1.

Меры оценки значимости выравнивания

Локальное выравнивание часто применяется для поиска неточных вхождений одной строки в другую (например, при поиске последовательностей отдельных генов в геноме). При этом длина первой строки как правило намного меньше длины второй. В ходе данной лабораторной работы решается подобная задача. «Короткая» последовательность часто называется *запросом* (от англ. query), а «длинная» – *референсом* (от англ. reference).

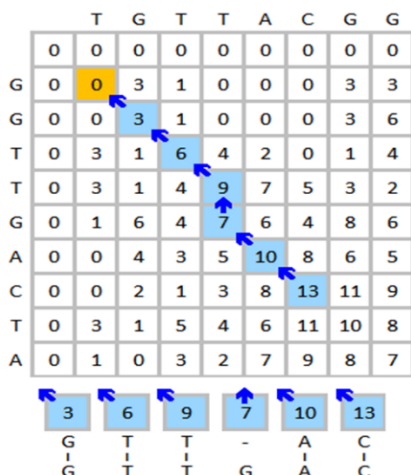


Рис. 5.1. Матрица выравнивания последовательностей TGTACGG и GTTGACTA, вычисленная по алгоритму Смита-Уотермана. Стрелками обозначен маршрут обратного прохода. Балл за совпадение символов – 3, штраф за несовпадение – -3, штраф за пропуск – -2

Для оценки значимости выравниваний используются:

1) длина выравнивания (l_a) – количество символов в фрагментах, на протяжении которых наблюдается сходство последовательностей. Для примера локального выравнивания последовательностей АСТТГТГАА и АСAGTCGCAACTGTGAAAАСТAGT длина выравнивания равна 9 пар нуклеотидов (п.н.);

2) покрытие (от англ. coverage) – отношение длин выравнивания l_a и запроса l_q).

$$cov = \min\left(\frac{l_a}{l_q}, 1\right) \quad (5.2)$$

Следует отметить, что значение l_a вследствие присутствия в выравнивании пропусков может превосходить l_q (как в примере на рисунке 5.1). В то же время значение $cov > 1$ неинформативно, поэтому в формуле (5.2) присутствует оператор \min . В примере выравнивания АСТТГТГАА и АСAGTCGCAACTGTGAAAАСТAGT покрытие равно 1, поскольку $l_a = l_q$;

3) процент совпадений – отношение количества совпадающих символов в выравнивании N_m к длине выравнивания l_a

$$p_i = \frac{N_m}{l_a} \cdot 100\% \quad (5.3)$$

В примере выравнивания последовательностей АСТТГТГАА и АСAGTCGCAACTGTGAAAАСТAGТ процент совпадений равен $\frac{8}{9} \cdot 100\% = 88,89\%$;

4) процент пропусков – отношение количества пропусков (вставок и удалений) в выравнивании N_g к длине выравнивания l_a :

$$p_g = \frac{N_g}{l_a} \cdot 100\%, \quad (5.4)$$

В примере выравнивания последовательностей АСТТГТГАА и АСAGTCGCAACTGTGAAAАСТAGТ процент пропусков равен $\frac{1}{9} \cdot 100\% = 11,11\%$;

5) координаты начала и конца выравнивания – позиции нуклеотидов, с которых начинаются области локального сходства. Пример таблицы координат начала и конца выравнивания последовательностей АСТТГТГАА и АСAGTCGCAACTGTGAAAАСТAGТ представлен в таблице 5.1.

Таблица 5.1.

Координаты начала и конца областей локального выравнивания последовательностей АСТТГТГАА и АСAGTCGCAACTGTGAAAАСТAGТ

Координата	Значение
Начало запроса	1
Конец запроса	9
Начало референса	10
Конец референса	17

Порядок выполнения

1. С помощью *help* изучить функции *diag*, *rownames*, *colnames*, *strsplit*, *unlist*, *length*, *paste*, *nchar*, *trimws*.

2. Разработать алгоритм Смита-Уотермана.

3. Загрузить в RStudio файл LocalAlignLab.R и изучить его содержимое. В файле размещен шаблон программного кода для выполнения лабораторной работы.

4. Программно реализовать алгоритм Смита-Уотермана.

5. Выполнить локальное выравнивание строк нуклеотидных последовательностей согласно вашему варианту (таблица 5.2), назначив балл за совпадение символов – 1, штраф за несовпадение – -1, штраф за пропуск – -1. При этом гарантируется, что лишь одна ячейка матрицы выравнивания содержит максимальное значение.

6. Программно реализовать расчёт мер значимости полученного выравнивания: длина, покрытие, проценты совпадений и пропусков, координаты начала и конца последовательностей запроса и референса.

7. Исследуйте влияние штрафа за удаление, счетов за совпадение и несовпадение символов на точность локального выравнивания последовательностей.

Форма отчета: работающий программный модуль, реализующий задание, тексты программ. Результаты работы алгоритма Смита-Уотермана. Выводы по результатам обработки экспериментальных данных.

Таблица 5.2.

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. Что такое локальное выравнивание последовательностей?
2. В чем заключается задача локального выравнивания последовательностей?
3. Что такое матрица замен и для чего её используют?
4. Что такое точечная матрица сходства и для чего её используют?
5. В чем суть алгоритма Смита-Уотермана?
6. Причислите основные этапы работы алгоритма Смита-Уотермана.
7. Перечислите и объясните основные меры значимости оцененного локального выравнивания последовательностей.
8. Для каких целей используются функции R: *diag*, *rownames*, *colnames*, *strsplit*, *unlist*, *length*, *paste*, *nchar*, *trimws*?

ЛАБОРАТОРНАЯ РАБОТА № 6

ИЕРАРХИЧЕСКИЕ МЕТОДЫ КЛАСТЕРНОГО АНАЛИЗА МОЛЕКУЛ ДНК

Цель работы. Применение методов иерархического кластерного анализа для группировки молекул ДНК.

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 молекул ДНК можно сформулировать следующим образом.

Имеется некоторое конечное множество объектов $C = \{n_1, n_2, \dots, n_N\}$, представляющее нуклеотидные последовательности молекул ДНК, произвольные молекулярные полимеры или текстовые строки, закодированные символами некоторого алфавита. 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 способы:

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

Для сравнения двух объектов n_i и n_j , молекул ДНК, могут быть использованы следующие меры или расстояния:

1. Левенштейна – минимальное число операций редактирования (удаление, вставка или замена), необходимых, чтобы преобразовать последовательность n_i в n_j ;

2. оптимального выравнивания (optimal string alignment или restricted Damerau-Levenshtein distance) – расстояние Левенштейна с учетом перестановок смежных символов; символы и подстроки редактируются только один раз;

3. Дамерау-Левенштейна – расстояние оптимального выравнивания с возможностью множественного редактирования символов и подстрок;

4. q -грамм. q -грамма является фрагментом из q последовательных символов строки последовательности. Если x и y – векторы частот q -грамм-событий в строках последовательностей n_i и n_j , то расстояние q -грамм есть

$$d_{qs}(n_i, n_j) = \sum_{l=1}^L |x_l - y_l|, \quad (6.1)$$

где L – количество q -грамм;

5. косинусное

$$d_{\cos}(n_i, n_j) = 1 - \frac{x \cdot y}{\|x\| \cdot \|y\|}, \quad (6.2)$$

где x и y – векторы частот q -грамм-событий в последовательностях n_i и n_j , $\|\dots\|$ – евклидова норма вектора;

6. Джаро

$$d_j(n_i, n_j) = \begin{cases} 0, & \text{если } m = 0 \\ \frac{1}{3} \left(\frac{m}{|n_i|} + \frac{m}{|n_j|} + \frac{m-t}{m} \right), & \text{если } m \neq 0 \end{cases}, \quad (6.3)$$

где $|n_i|$ и $|n_j|$ – число символов в строках n_i и n_j , m – число совпадающих символов последовательностей, t – половина числа транспозиций. Два символа рассматриваются совпадающими, если они одинаковы и расположены не дальше, чем $\max(|n_i|, |n_j|)/2 - 1$. Транспозиция – половина количества совпадающих и отличающихся порядковыми номерами символов. Пример. При сравнении слова CRATE со словом TRACE, только R, A и E являются совпадающими символами, то есть $m = 3$. С и Т появляются в обоих строках, но они расположены дальше, чем на 1 ($\text{floor}(5/2) - 1 = 1$) от тех мест, где бы они совпадали. Число транспозиций рано 0, т.к. совпадающие символы R, A и E не отличаются порядковыми номерами в последовательностях;

7. наибольшей общей подстроки (longest common substring distance) – определяется как самая длинная общая подстрока, полученная путем объединения символов из последовательностей n_i и n_j , сохраняя порядок символов.

Иерархические методы используют последовательное объединение объектов в кластеры, малые кластеры в большие, которое может быть визуально отображено в виде дерева вложенных кластеров – дендрограммы (рис. 6.1), 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.

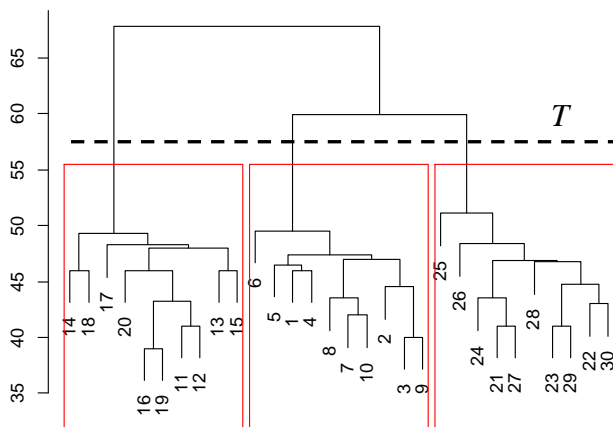


Рис. 6.1. Дендрограмма

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)

$$d_{rs} = \min(d_{is}, d_{js}); \quad (6.4)$$

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

$$d_{rs} = \max(d_{is}, d_{js}); \quad (6.5)$$

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

$$d_{rs} = \frac{m_i d_{is} + m_j d_{js}}{m_i + m_j}, \quad (6.6)$$

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

4. центроидный метод

$$d_{rs} = \frac{m_i d_{is} + m_j d_{js}}{m_i + m_j} - \frac{m_i m_j d_{ij}}{(m_i + m_j)^2}; \quad (6.7)$$

5. метод медианной связи

$$d_{rs} = (d_{is} + d_{js}) / 2 - d_{ij} / 4. \quad (6.8)$$

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

Оценка различия. Используется для определения наиболее оптимального выбора меры сходства 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 κ . Построение иерархического дерева считается успешным, если кофенетический корреляционный коэффициент κ 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. 6.1). 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.

Программная реализация алгоритмов в R

Различные способы вычисления мер схожести текстовых последовательностей реализованы в функции *stringdistmatrix()* пакета *stringdist*. Вызов функции:

```
stringdistmatrix(a, a, method)
```

где a – матрица входных данных размера N на 1 (объект класса *matrix*), содержащая последовательности молекул ДНК; *method* – метод вычисления меры схожести последовательностей ('lv' – Левенштейна, 'osa' – оптимального выравнивания, 'dl' – Дамерау-Левенштейна, 'qgram' – q -грамм, 'cosine' – косинусное, 'jw' – Джаро, 'lcs' – наибольшей общей подстроки).

Функция *stringdistmatrix()* возвращает объект класса *matrix*, содержащий матрицу расстояний между объектами по заданной мере сходства. Для преобразования данных к объекту класса *dist*, требуемому в ходе построения иерархических деревьев с помощью функции *hclust*, используется функция *as.dist*.

Для иерархического кластерного анализа повышенной сложности используется пакет *dendextend*. Например, для графического отображения медоидов кластеров, объектов, имеющих наименьшие средние расстояния до остальных объектов соответствующих кластеров (рис. 6.2), можно использовать фрагмент кода:

```
>> colors <- rep(2,N)
>> colors[c(medoid1_ind, medoid2_ind, medoid3_ind)] <- 3
>> dend <- hcTree %>% as.dendrogram
>> dend %>% plot
>> colored_bars(colors = colors, dend = dend)
```

где *medoid1_ind*, *medoid2_ind*, *medoid3_ind* – индексы объектов-медоидов трех соответственных кластеров в нумерации исходных данных; *hcTree* – объект класса *hclust*, возвращаемый функцией *hclust*.

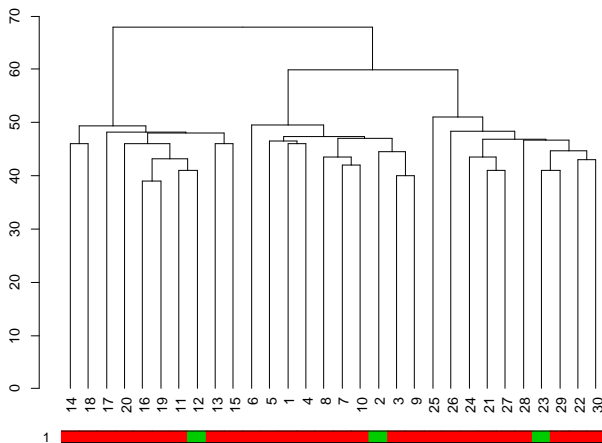


Рис. 6.2. Дендрограмма и медоиды трех кластеров

Основные этапы иерархического кластерного анализа в R

Этап 1. Вычисление матрицы расстояний между объектами D (функция *stringdistmatrix*).

Этап 2. Связывание или группировка объектов в иерархические деревья (дендрограммы) (функции *as.dist* и *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 функций *install.packages* и *library* установить и подключить пакеты *stringdist* и *dendextend*.

2. С помощью *help* изучить функции *dist*, *hclust*, *cophenetic*, *cutree*, *rect.hclust*, *points*, *stringdistmatrix*.

2. Загрузить данные согласно вашему варианту (табл. 6.1). Проверить правильность загрузки экспериментальных данных.

3. Вычислить расстояния между объектами. Использовать меры для расчета расстояний согласно варианту (табл. 6.1).

4. Выполнить кластерный анализ исходных данных методом иерархической кластеризации по способам 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 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 медоиды кластеров и отобразить их на рисунке дендрограммы с использованием программных конструкций пакета dendextend (см. пример программных кодов в тексте и на рис. 6.2).

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

Таблица 6.1

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

* Меры сходства или расстояния: 1 – Левенштейна, 2 – оптимального выравнивания, 3 – Дамерау-Левенштейна, 4 – q-грамм, 5 – косинусное, 6 – Джаро или наибольшей общей подстроки.

** 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, представленных текстовыми последовательностями.
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. Объясните понятие медоида кластера.
10. Каким образом определить значимое число кластеров?
11. Для каких целей используются функции R: *dist*, *hclust*, *cophenetic*, *cutree*, *rect.hclust*, *points*, *stringdistmatrix*?

ЛАБОРАТОРНАЯ РАБОТА № 7

МЕТОД ГЛАВНЫХ КООРДИНАТ ДЛЯ ВЕКТОРИЗАЦИИ И ВИЗУАЛИЗАЦИИ МОЛЕКУЛ ДНК

Цель работы. Практическое освоение метода главных координат для решения задач векторизации и визуализации молекул ДНК.

КРАТКИЕ СВЕДЕНИЯ

Метод главных координат (часто называют методом главных проекций или линейным шкалированием Торгерсона, по имени автора) предназначен для снижения размерности данных, путем вычисления главных координат или проекций, где объекты n_1, n_2, \dots, n_N не векторизованы, но расстояния между ними могут быть вычислены в той или иной мере сходства. Примеры подобных типов объектов – текстовые строки, рекламные объявления, молекулы РНК, ДНК и белков, представленные символьными обозначениями.

В методе решаются две задачи: 1) векторизация объектов (переход в новое пространство признаков) и 2) нахождение главных координат или компонент, обладающих свойствами, аналогичными главным компонентам в методе главных компонент (на первую главную компоненту приходится максимальная доля дисперсии в данных, на вторую – меньшая или равная первой доли дисперсии и т.д. – $\sigma^2(Z_1) \geq \sigma^2(Z_2) \geq \dots$). Векторизация нуклеотидной последовательности – это эффективное преобразование символьной последовательности в вектор признаков.

Необходимо вычислить матрицу квадратов расстояний D^2 между объектами n_1, n_2, \dots, n_N набора данных в некоторой мере сходства

$$D^2 = \begin{pmatrix} 0 & d_{12}^2 & \dots & d_{1N}^2 \\ d_{21}^2 & 0 & \dots & \dots \\ \dots & \dots & \dots & \dots \\ d_{N1}^2 & d_{N2}^2 & \dots & 0 \end{pmatrix} \quad (7.1)$$

где $d_{ij} = d_2(n_i, n_j)$ – некоторая мера сходства последовательностей (Левенштейна, косинусное, наибольшей общей подстроки или другое).

Применяется процедура двойного центрирования матрицы D^2 , в результате которой средние значения по столбцам и строкам равны 0. Пусть

матрица G есть дважды центрированная матрица D^2 , тогда ее элементы равны

$$g_{ij} = -\frac{1}{2}(d_{ij}^2 - d_{\bullet j}^2 - d_{i\bullet}^2 + d_{\bullet\bullet}^2) \quad (7.2)$$

$$d_{\bullet j}^2 = \frac{1}{N} \sum_{i=1}^N d_{ij}^2 \quad (7.3)$$

$$d_{i\bullet}^2 = \frac{1}{N} \sum_{j=1}^N d_{ij}^2 \quad (7.4)$$

$$d_{\bullet\bullet}^2 = \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N d_{ij}^2 \quad (7.5)$$

Метод главных координат основан на двух свойствах матрицы G :

1) имеет место равенство

$$G = X \cdot X^T, \quad (7.6)$$

где X – матрица размера $N \times K$ некоторых исходных K признаков объектов n_1, n_2, \dots, n_N , как если бы они были измерены в реальном эксперименте;

2) отличные от 0 собственные значения λ_j^G матрицы G и собственные значения λ_j^C матрицы $C = X^T \cdot X$ одинаковы, а собственные векторы α_j^G и α_j^C связаны следующим соотношением

$$\sqrt{\lambda_j^G} \alpha_j^G = X \alpha_j^C = Z_j, \quad (7.7)$$

где Z_j – есть искомая главная координата (или компонента в методе главных компонент). Отметим, что если набор данных X центрирован и шкалирован, то матрица $C / (N - 1)$ есть *корреляционная матрица*, используемая в методе главных компонент.

Таким образом, не зная исходных признаков объектов X , можно на основе преобразованной матрицы оценок сходства объектов D^2 восстановить конфигурацию точек в пространстве главных координат/компонент. Следует отметить, что сама матрица X не восстанавливается однозначно по матрице D^2 .

Неизвестные собственные значения и векторы матрицы G можно оценить применив степенной метод. Идея степенного метода состоит в последовательном итерационном нахождении максимальных собственных значений λ_j (таких, что $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_N$) и векторов $\alpha_j, j = 1, 2, 3, \dots, N$, матрицы G , воспользовавшись свойствами ортогональности собственных векторов и симметричности матрицы G .

Степенной метод

Алгоритм нахождения собственного значения $\lambda_j, j=1, 2, 3, \dots, N$, и соответствующего ему вектора $\alpha_j = (\alpha_{1j}, \alpha_{2j}, \dots, \alpha_{Nj})^T$ диагональной матрицы G :

Шаг 1. Установить $i = 0, t = 0$ и ε (точность вычисления). Сгенерировать случайным образом вектор $x_i = (x_{i1}, x_{i2}, \dots, x_{iN})$ (размер матрицы $G - N \times N$; если $x_i \alpha_j^T = 0$, то регенерировать x_i).

Шаг 2. Ортогонализация x_i к векторам $\alpha_l, l=1, 2, \dots, (j-1)$ (при вычислении λ_1 не производится):

$$x_i^o = x_i - a_1^T x_i^T a_1^T - a_2^T x_i^T a_2^T - \dots - a_{j-1}^T x_i^T a_{j-1}^T. \quad (7.8)$$

Шаг 3. Выполнить нормировку x_i^o :

$$x_i^{o'} = \frac{x_i^o}{\sqrt{\sum_{j=1}^N x_{ij}^{o2}}}, \quad (7.9)$$

Шаг 4. Вычислить :

$$x_{i+1} = x_i^{o'} G \quad \text{и} \quad t_1 = x_{i+1} (x_i^{o'})^T. \quad (7.10)$$

Шаг 5. Выполнить нормировку x_{i+1} :

$$x_{i+1}' = \frac{x_{i+1}}{\sqrt{\sum_{j=1}^N x_{(i+1)j}^2}}, \quad (7.11)$$

Шаг 6. Если $|t_1 - t| \leq \varepsilon$, то $\lambda_j = t_1$ и $a_j = x_{i+1}'$, завершить алгоритм. Иначе $i = i + 1$ и $t = t_1$, перейти к *ш. 2.*

Алгоритм метода главных координат

Шаг 1. Сформировать матрицу исходных данных (объекты n_1, n_2, \dots, n_N – нуклеотидные последовательности молекул ДНК) и на ее основе вычислить матрицу D^2 .

Шаг 2. Вычислить матрицу G (двойное центрирование матрицы D^2).

Шаг 3. Вычислить M первых максимальных собственных значений λ_j и соответствующих им векторов α_j матрицы G , используя степенной метод.

Шаг 4. Рассчитать проекции объектов на оси главных координат

$$Z_j = \sqrt{\lambda_j} \alpha_j. \quad (7.12)$$

В общем случае, векторизация нуклеотидной последовательности молекулы ДНК производится в N главных координат, представляющих элементы вектора признаков. Однако вклад большей части главных координат в объясняемую дисперсию оказывается небольшим. Исключают из рассмотрения те координаты, вклад которых мал. С помощью $M, M \ll N$,

первых (наиболее весомых) главных координат можно объяснить основную часть суммарной дисперсии в данных. При этом, различия между объектами зависят от доли изменчивости, связанной с данной главной координатой.

Каким образом оценить необходимое число главных координат? Наиболее простой способ – построить график зависимости собственных значений от порядкового номера главной координаты и провести аппроксимирующую прямую линии через точки координат с малым вкладом (рис. 7.1.). Отобразить те координаты, которые не попадают на прямую линию.

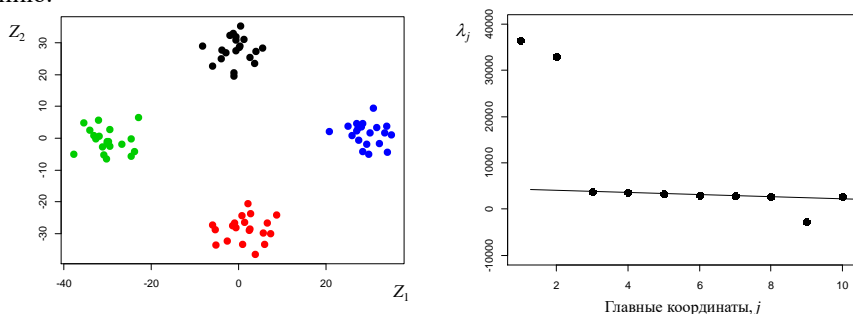


Рис. 7.1. Молекулы ДНК в пространстве первых $M = 2$ главных координат и график зависимости собственных значений от порядкового номера главной координаты

В заключение отметим, что преобразование главных координат является линейным.

Порядок выполнения

1. С помощью функции *library* подключить пакет *stringdist*.
2. Разработать алгоритм метода главных координат с использованием степенного метода нахождения собственных значений и векторов; программно реализовать его в среде R.
3. Выполнить анализ экспериментальных данных методом главных координат согласно варианту (табл. 7.1).
 - Загрузить данные. Проверить правильность загрузки экспериментальных данных.
 - Меры сходства между нуклеотидными последовательностями по вариантам расположены в табл. 7.1.
 - На основе первых $M = 2$ главных координат построить диаграмму рассеяния (рис. 7.1). Отобразить графически кластеры молекул ДНК, определённые иерархическим методом кластерного анализа

(для цветового обозначения кластеров используется вектор индексов принадлежности объектов к кластерам из лабораторной работы №6).

- Построить график зависимости собственных значений от порядкового номера главной координаты для $M = 10$ первых координат (рис. 7.1). Определить оптимальное число главных координат.

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

Таблица 7.1
Варианты заданий

Вариант	Меры сходства	Данные
1	Левенштейна	data1.txt
2	оптимального выравнивания	data2.txt
3	Дамерау-Левенштейна	data3.txt
4	q-грамм	data4.txt
5	косинусное	data5.txt
6	Джаро	data6.txt
7	наибольшей общей подстроки	data7.txt
8	Левенштейна	data8.txt
9	оптимального выравнивания	data9.txt
10	Дамерау-Левенштейна	data10.txt
11	q-грамм	data11.txt
12	косинусное	data12.txt

Контрольные вопросы

1. Для каких задач обработки экспериментальных данных используется метод главных координат?
2. В чем состоят задачи векторизации и визуализации нуклеотидных последовательностей молекул ДНК?
3. В чем суть метода главных координат?
4. В чем сходство методов главных компонент и координат?
5. С какой целью в работе используется степенной метод и в чем его суть?
6. Что представляют собой главные координаты?
7. Каким образом оценить необходимое число главных координат?

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. Стефанов, В. Е. Биоинформатика: учебник для академического бакалавриата / В. Е. Стефанов, А. А. Тулуб, Г. Р. Мавропуло-Столяренко. – М.: Издательство Юрайт, 2019. – 252 с.
4. Часовских, Н. Ю. Биоинформатика: учебник / Н. Ю. Часовских. – М.: ГЭОТАР-Медиа, 2020. – 352 с.
5. Наглядная математическая статистика: учеб. пособие / Б. М. Лагутин. – М.: БИНОМ. Лаборатория знаний, 2021. – 472 с.
6. Норман, М. Искусство программирования на R. Погружение в большие данные / М. Норман; пер. с англ. – СПб.: Питер, 2019. – 416 с.
7. 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 БОЛЬШИХ НАБОРОВ БИОЛОГИЧЕСКИХ ДАННЫХ	14
ЛАБОРАТОРНАЯ РАБОТА №3 СБОРКА ГЕНОМА <i>DE NOVO</i> . МЕТОД ГРАФОВ ДЕ БРЮЙНА.....	19
ЛАБОРАТОРНАЯ РАБОТА № 4 ГЛОБАЛЬНОЕ ВЫРАВНИВАНИЕ ПАР НУКЛЕОТИДНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ. АЛГОРИТМ НИДЛМАНА-ВУНША	24
ЛАБОРАТОРНАЯ РАБОТА №5 ЛОКАЛЬНОЕ ВЫРАВНИВАНИЕ ПАР НУКЛЕОТИДНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ. АЛГОРИТМ СМИТА-УОТЕРМАНА	29
ЛАБОРАТОРНАЯ РАБОТА № 6 ИЕРАРХИЧЕСКИЕ МЕТОДЫ КЛАСТЕРНОГО АНАЛИЗА МОЛЕКУЛ ДНК	34
ЛАБОРАТОРНАЯ РАБОТА № 7 МЕТОД ГЛАВНЫХ КООРДИНАТ ДЛЯ ВЕКТОРИЗАЦИИ И ВИЗУАЛИЗАЦИИ МОЛЕКУЛ ДНК	42
СПИСОК ЛИТЕРАТУРЫ	47

Учебное издание

Яцков Николай Николаевич
Сиколенко Максим Александрович
Чепелева Марина Кирилловна

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, М.К. Чепелева*

Подписано в печать 16.01.2021. 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.