Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Пермский государственный аграрно-технологический университет

имени академика Д.Н. Прянишникова»

Кафедра информационных технологий и программной инженерии

**ЛАБОРАТОРНАЯ РАБОТА № 0**

Основы языка R.

|  |  |
| --- | --- |
| Выполнил: | группа ПИНб-4  П.С. Плотников |
| Проверил: | доцент каф. ИТиПИ,  Т.А. Казаченко |

Пермь-2024 г.

**Презентация**

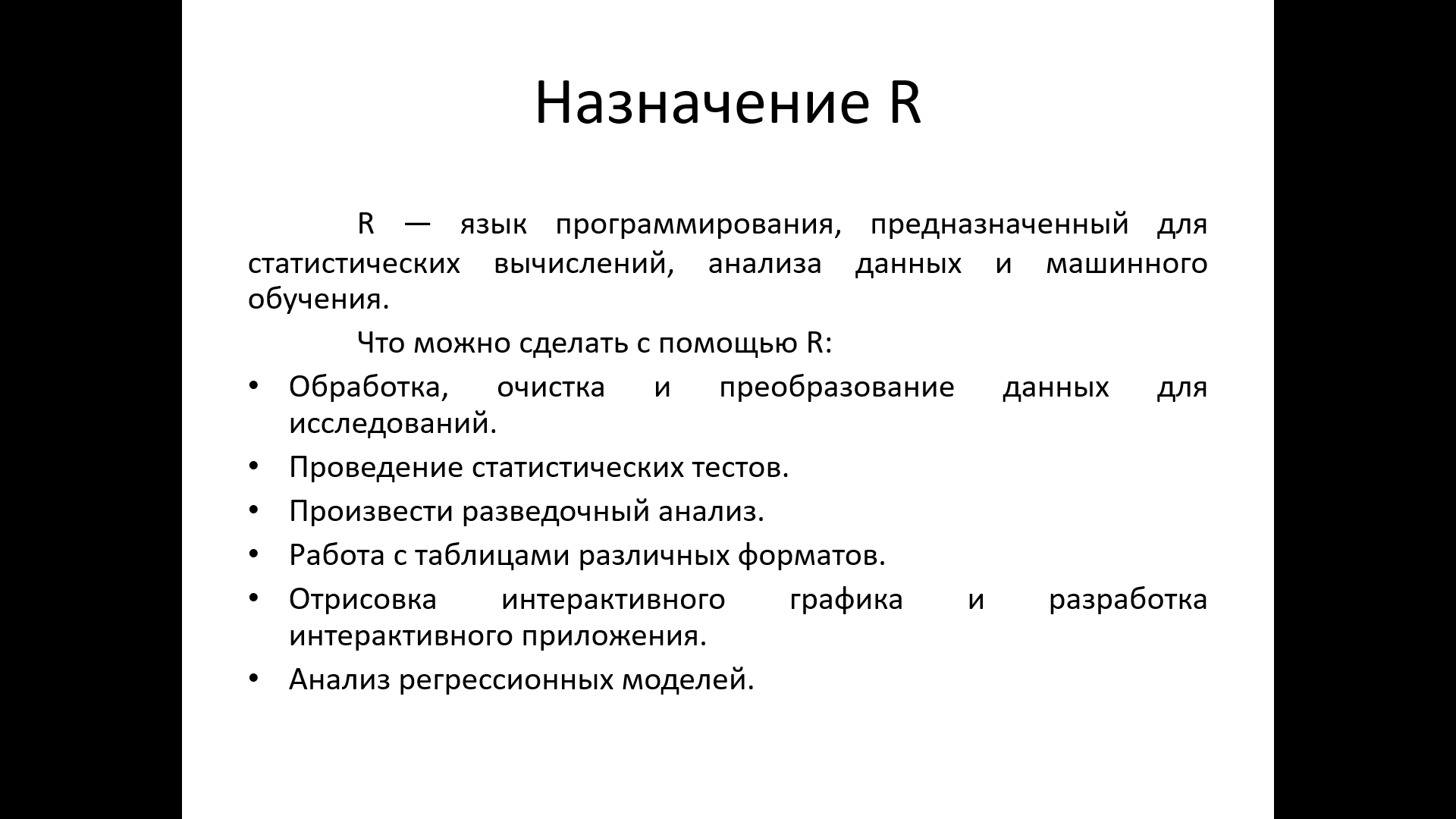
Изображение выглядит как текст, снимок экрана, Шрифт, графический дизайн

Автоматически созданное описание

R — это диалект S, так как полностью наследует идеологию и конструкцию языка, вплоть до обратной совместимости базовых выражений. Язык S был создан в отделе статистики компании AT&T в её исследовательском подразделении Bell Labs в 1976 году. Первая версия языка была призвана упростить работу со статистической библиотекой, написанной на Fortran.

Язык программирования R разработали ученые факультета статистики Оклендского университета. Первоначально это был внутренний инструмент, который впоследствии стал доступен широкой общественности.

R — язык с открытым исходным кодом, который до сих пор применяется в научных программах и упрощает работу статистиков по всему миру.



Одним из основных мотивов создания системы (позднее R уже стали называть языком) послужила проприетарность S. R был разработан для статистики. Соответственно, в нем есть популярные тексты, методы анализа данных и практичные в использовании инструменты для построения графиков.

У языка R чёткая сфера применения — статистические вычисления, анализ данных и машинное обучение. R — не только язык для анализа данных, но и целая рабочая среда, куда уже встроены готовые методы статистического анализа и инструменты для визуализации.

Синтаксис R — это набор правил и структур, которые определяют формат и организацию программ на этом языке. Программа на R состоит из трёх элементов: переменных, комментариев и ключевых слов.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Преимущества:

1. Чтобы писать программы, не обязательно соблюдать четкую структуру – можно просто последовательно вводить данные.
2. Его структура и синтаксис понятны аналитикам и статистикам по умолчанию.
3. Можно построить двумерную графику и трехмерные модели.
4. Это очень важный параметр для начинающих программистов.
5. Не обязательно устанавливать сторонние приложения.

Недостатки:

1. Многие действия можно выполнить несколькими способами.
2. По этой причине многие действия можно выполнить несколькими способами.
3. В отличие от прямолинейного «питона», программа обучения R кажется немного сложной и запутанной.

**Формулировка задачи**

**Задача 1**

**Условие**

1. Создайте три переменные: числовую, символьную, логическую, присвойте им значения. Выведите тип каждого объекта.

**Программный код**

# Создание числовой переменной

number = 100

# Создание строковой переменной

string = "Пример строки"

# Создание логической переменной

logical = TRUE

# Вывод типов данных переменных

typeof(number)

typeof(string)

typeof(logical)

**Вывод в консоли**

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 1 – Создание переменных и вывод из типов данных

**Условие**

1. Создайте три вектора: с числовыми, с символьными, с логическими данными.

**Программный код**

# Создание числового вектора

vekt\_1 = c(10,20)

# Создание символьного вектора

vekt\_2 = c("x", "y")

# Создание логического вектора

vekt\_3 = c(TRUE, FALSE)

# Вывод результата

vekt\_1

vekt\_2

vekt\_3

**Вывод в консоли**

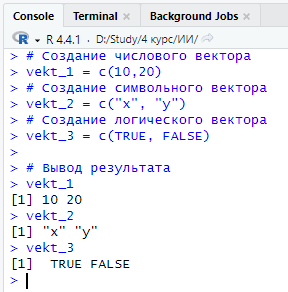


Рисунок 2 – Создание векторов разных типов

**Условие**

1. Создайте из этих векторов фрейм.

**Программный код**

# Создание числового вектора

vekt\_1 = c(10,20)

# Создание символьного вектора

vekt\_2 = c("x", "y")

# Создание логического вектора

vekt\_3 = c(TRUE, FALSE)

# Создание фрейма из созданных векторов

result <- data.frame(vekt\_1, vekt\_2, vekt\_3)

# Вывод результата

Result

**Вывод в консоли**

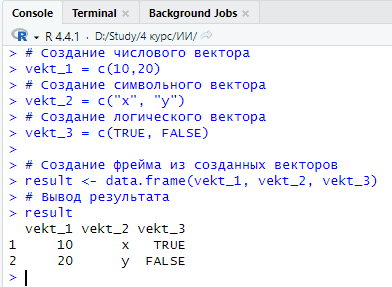


Рисунок 3 – Создание фрейма из полученных ранее векторов

**Задание 2**

**Условие**

1. Создать фрейм с помощью функции fix().

**Программный код**

# Создаем пустой фрейм

frame\_1 = data.frame(Number\_1 = numeric(), Number\_2 = numeric(), Number\_3 = numeric())

# Редактируем фрейм

fix(frame\_1)

# Выводим результат

frame\_1

**Окно редактирования fix()**

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 4 – Окно редактирования функции fix()

**Вывод в консоли**

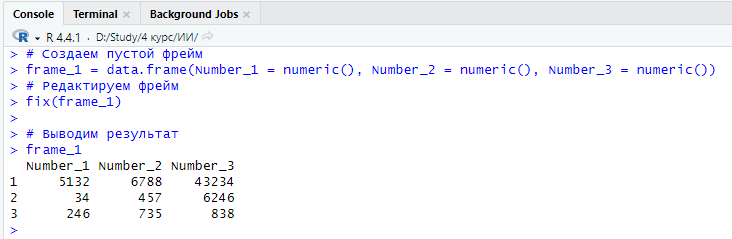


Рисунок 5 – Создание фрейма при помощи fix()

**Условие**

1. Создать фрейм импорта файла csv. Для этого найдите и скачайте из сети файл titanic.csv.

**Программный код**

# Создаем фрейм, загружая в него данные csv-файла

data\_csv <- data.frame(read.csv("./titanic.csv"))

**Вывод в консоли**

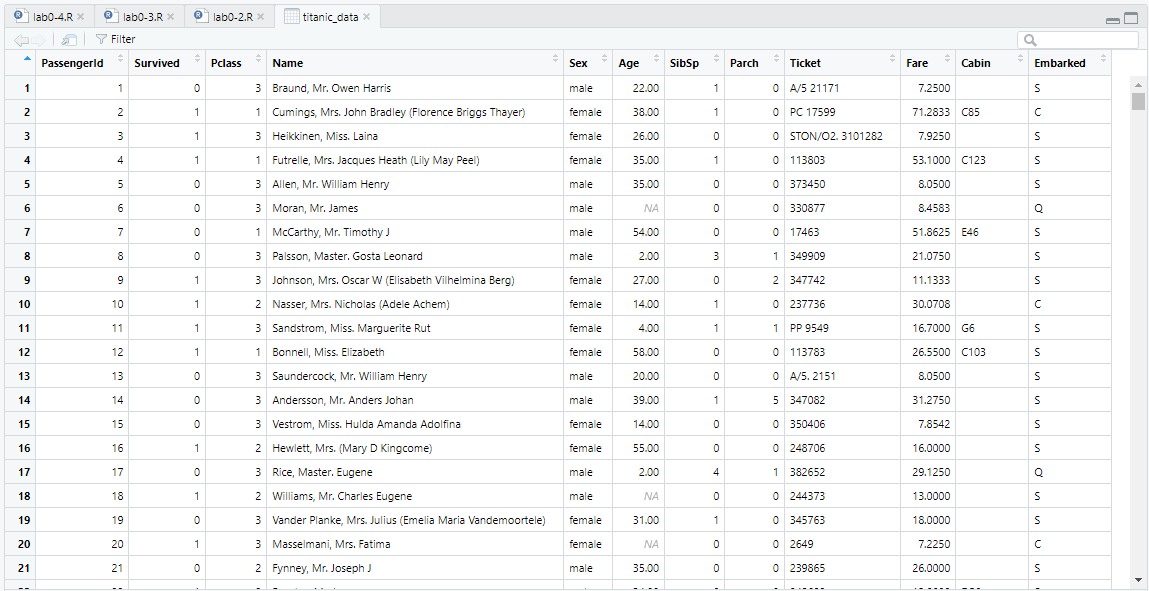


Рисунок 5 – Загруженный csv-файл

**Задание 3**

**Условие**

Допустим, три дня подряд выполнялось по три измерения каждый день. Данные представляют собой выборку от 1 до 9.

Требуется превратить выборку, состоящую из цифр от 1 до 9, в таблицу из трех колонок.

**Программный код**

# Функция для генерации выборки

get\_random\_vektor <- function() {

# Получение случайного вектора

vektor = c(sample(1:9, 1), sample(1:9, 1), sample(1:9, 1))

}

# Получем выборку по каждому дню

day\_1 = get\_random\_vektor()

day\_2 = get\_random\_vektor()

day\_3 == get\_random\_vektor()

# Выводим полученные векторы

day\_1

day\_2

day\_3

# Создаем фрейм из полученных векторов

days\_data <- data.frame(day\_1, day\_2, day\_3)

# Выводим результат

days\_data

**Вывод в консоли**

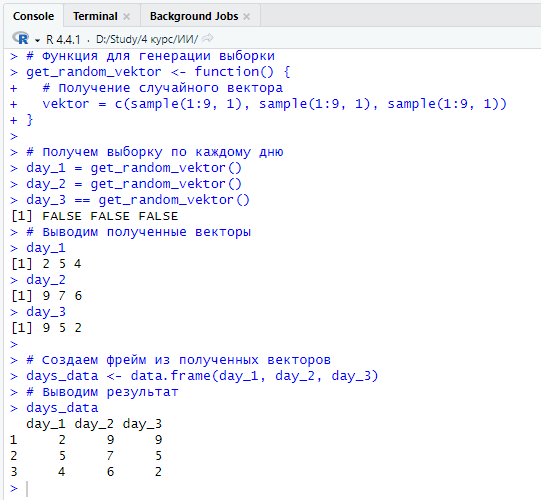


Рисунок 6 – Получение выборки по результатам измерения

**Задание 4**

**Условие**

1. Какое количество мужчин и женщин ехало на корабле? В качестве ответа приведите два числа через пробел.

**Программный код**

# Функция для подсчета мужчин и женщин

get\_count\_of\_pass <- function(peoples) {

# Берем подмножество значений с полом "male"

male\_data = subset(peoples, Sex =="male")

# Берем подмножество значений с полом "female"

female\_data = subset(peoples, Sex =="female")

# Берем кол-во строк в первом подмножестве

male\_count = nrow(male\_data)

# Берем кол-во строк во-втором подмножестве

female\_count = nrow(female\_data)

# Формируем вектор-результат

result = c(male\_count, female\_count)

# Вывод результата в консоль

result

}

# Получение данных csv-файла

titanic\_data <- data.frame(read.csv("./titanic.csv"))

# Вызов функции с передачей в нее подмножества столбца "Sex"

get\_count\_of\_pass(titanic\_data[5])

**Вывод в консоли**

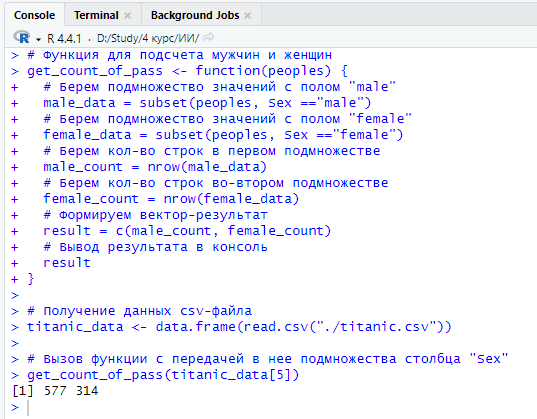


Рисунок 7 – Количество мужчин и женщин на корабле

**Условие**

1. Какой части пассажиров удалось выжить? Посчитайте долю выживших пассажиров. Ответ приведите в процентах (число в интервале от 0 до 100, знак процента не нужен), округлив до двух знаков.

**Программный код**

# Функция для подсчета доли выживших

get\_percent\_of\_surv <- function(survivors) {

# Берем подмножество значений с выжившими

survival\_data = subset(survivors, Survived == 1)

# Берем общее кол-во строк в файле

all\_count = nrow(survivors)

# Берем кол-во строк в полученном подмножестве

alive\_count = nrow(survival\_data)

# Формируем результат

result = round(alive\_count / all\_count \* 100, 2)

# Вывод результата в консоль

result

}

# Получение данных csv-файла

titanic\_data <- data.frame(read.csv("./titanic.csv"))

# Вызов функции с передачей в нее подмножества столбца "Survived"

get\_percent\_of\_surv(titanic\_data[2])

**Вывод в консоли**

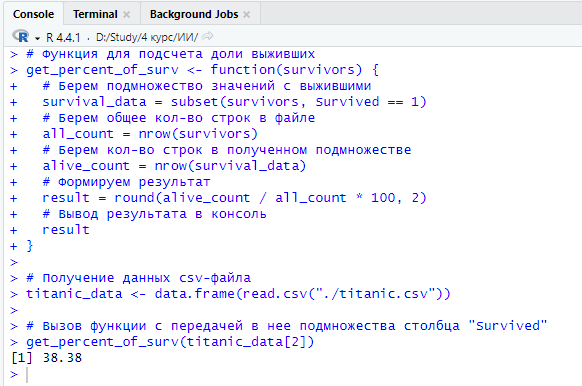


Рисунок 8 – Доля выживших на корабле

**Условие**

1. Какую долю пассажиры первого класса составляли среди всех пассажиров? Ответ приведите в процентах (число в интервале от 0 до 100, знак процента не нужен), округлив до двух знаков.

**Программный код**

# Функция для подсчета доли пассажиров первого класса

get\_percent\_of\_f\_class <- function(classes) {

# Берем подмножество значений пассажирами 1-го класса

f\_class\_data = subset(classes, Pclass == 1)

# Берем общее кол-во строк в файле

all\_count = nrow(classes)

# Берем кол-во строк в полученном подмножестве

first\_count = nrow(f\_class\_data)

# Формируем результат

result = round(first\_count / all\_count \* 100, 2)

# Вывод результата в консоль

result

}

# Получение данных csv-файла

titanic\_data <- data.frame(read.csv("./titanic.csv"))

# Вызов функции с передачей в нее подмножества столбца "Pclass"

get\_percent\_of\_f\_class(titanic\_data[3])

**Вывод в консоли**

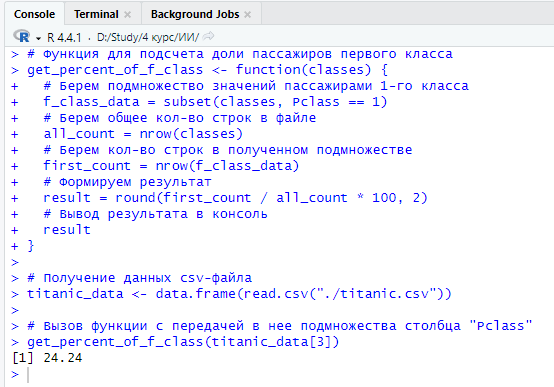


Рисунок 9 – Доля пассажиров первого класса

**Условие**

1. Какого возраста были пассажиры? Посчитайте среднее и медиану возраста пассажиров. В качестве ответа приведите два числа через пробел.

**Программный код**

# Функция для подсчета среднего возраста и медианы

get\_mid\_age\_of\_pass <- function(ages) {

# Берем подмножество, исключая null

ages\_data = na.omit(ages)

# Берем среднее значение возвраста пассажиров

mid\_ages = mean(ages\_data$Age)

# Берем медиану возраста пассажиров

median\_ages = median(ages\_data$Age)

# Формируем вектор-результат

result = c(round(mid\_ages, 2), round(median\_ages, 2))

# Вывод результата в консоль

result

}

# Получение данных csv-файла

titanic\_data <- data.frame(read.csv("./titanic.csv"))

# Вызов функции с передачей в нее подмножества столбца "Age"

get\_mid\_age\_of\_pass(titanic\_data[6])

**Вывод в консоли**

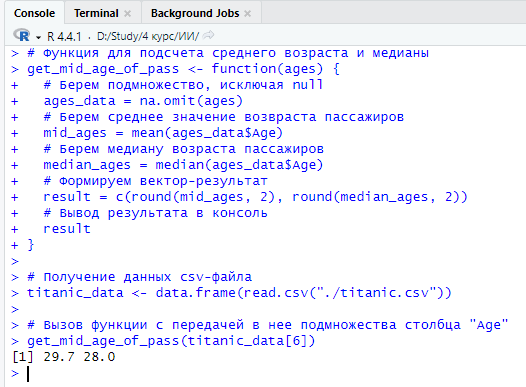


Рисунок 10 – Среднее значение и медиана возраста пассажиров