БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет радиофизики и компьютерных технологий
Кафедра радиофизики и цифровых медиа технологий

Лабораторная работа по курсу
Статистическая радиофизика

Введение в язык R с использованием RStudio

Минск, 2023

Цель работы: изучить основы языка R.

Общие сведения:

R — язык программирования системы статистической обработки данных и работы с графикой.

Язык был создан сотрудниками Оклендского университета Ross Ihaka и Robert Gentleman как развитие языка S, дополненное несколькими новыми идеями, позаимствованными из Scheme. Название языка основано на именах разработчиков и на отсылке к языку S. Разработка языка началась в 1993 году, и в апреле 1997 году был выпущен первый не-альфа релиз. С середины 1997 года разработка ведется так называемой R Core Team.

В настоящее время язык является частью GNU Project и распространяется под лицензией GNU GPL в виде исходных кодов и откомпилированных приложений под Linux, Freebsd, Windows, Solaris, Mac OS X и некоторые другие *nix-системы.

Основа системы R — интерпретируемый язык, поддерживающий структурный, модульный и объектно-ориентированный стили программирования. Язык R предоставляет программисту огромное множество встроенных статистических и графических инструментов, включая линейное модели, модели нелинейной регрессии, статистические тесты, анализ временных рядов, классификацию, кластеризацию и т.д. Кроме того, возможности базового языка легко расширяются с помощью пакетов: в базовую поставку включены 8 основных пакетов, а всего доступно более 800 специализированных дополнений. Наконец, R частично обратно совместим с S — многие программы, написанные на S, запустятся на R без изменений.

R широко используется в среде специалистов по статистике и анализу данных.

Основной тип данных языка — массивы. Массивы в R выступают как контейнеры (упорядоченные наборы однородных данных), а не математические вектора (элементы векторного пространства). Второй важнейший контейнер — списки, упорядоченные наборы неоднородных данных, некоторые из которых могут быть именованными. Скаляры представляют собой векторы длины 1. Скалярные типы данных, поддерживаемые языком, — logical, integer, double, complex, character, и raw.

R-Studio — группа полнофункциональных утилит для восстановления данных, включающая в себя как версии для WindowsOS, так и приложения, работающие в среде Mac OS и Linux OS. Данные могут восстанавливаться с жёстких дисков (HDD), твёрдотельных устройств (SSD), флэш-памяти и аналогичных внешних и внутренних накопителей данных. Предназначена для специалистов по восстановлению данных, но также может использоваться IT-профессионалами и простыми пользователями для самостоятельного восстановления утерянных файлов. Перед восстановлением выбранные файлы можно просмотреть во

встроенном просмотрщике. Считается одной из наиболее эффективных программ в своём классе и отличается высоким качеством восстановления.

Для непрофессиональных пользователей на ядре R-Studio разработан и предлагается программный продукт R-Undelete[7], работающий в только в среде Windows OS с упрощённым интерфейсом и отключёнными функциями по восстановлению с дисковых массивов, поддержкой восстановления по сети и модифицирования файлов редактором Hexedit.

Для домашних пользователей также предлагается бесплатная утилита R-Undelete Home, восстанавливающая файлы с файловой системы FAT/exFAT, наиболее используемой в USB-флеш памяти и SD-карт цифровых и видео камер.

Помимо самой программы восстановления данных, R-Studio включает в себя:

- Полнофункциональный редактор данных на диске.
- Модуль восстановления RAID. Поддерживаются как стандартные, так и пользовательские уровни. Есть автоматическое определение параметров RAID'ов.
- Модуль копирования дисков и создания их образов.
- Модуль мониторинга параметров S.M.A.R.T. для дисков.
- Модуль восстановления данных по локальной сети и Интернету.
- Аварийная версия R-Studio для восстановления данных с незагружаемых компьютеров. Может быть запущена с любого из съёмных медиа-устройств (USB, DVD, CD), поддерживаемых компьютером пользователя, независимо от инсталлированной операционной системы Windows, Macintosh или Linux.
- Модуль интеграции с DeepSpar Disk Imager — профессиональным устройством для создания образов с неисправных жёстких дисков. (Только для технической лицензии).

Что нужно для выполнения лабораторных работ?

1. Скачать и установить R-Studio. Установку производить с настройками по умолчанию.
2. Создать папку Rlabs, куда будут сохраняться проекты с лабораторными работами.
3. Установить дополнительные пакеты (RColorBrewer, dplyr, tree, caret, nnet, NeuralNetTools).

   Установка пакетов: Tools → Install Packages → в поле Packages ввести имя нужного пакета.
4. Взять у преподавателя папку с данными. Данные скопировать в папку Rlabs/Data/

При выполнении всех вышеперечисленных шагов у вас должно быть на компьютере всё необходимое для выполнения лабораторных работ

Основные команды в языке R:

- Создание и вывод переменной:

```
n <- 1.25
print(n)

## [1] 1.25
```

- Создание и вывод вектора:

```
v <- c(1, 2, 3, 4, 5);
print(v)

## [1] 1 2 3 4 5
```

- Создание и вывод фрейма данных:

```
df <- data.frame(
  Name = c("Cat", "Dog", "Cow", "Pig"),
  HowMany = c(5, 10, 15, 20),
  IsPet = c(TRUE, TRUE, FALSE, FALSE))
print(df)

##   Name HowMany IsPet
## 1  Cat       5  TRUE
## 2  Dog      10  TRUE
## 3  Cow      15 FALSE
## 4  Pig      20 FALSE
```

- Индексирование фрейма данных по 1 строке и 2 столбцу:

```
df[1, 2]

## [1] 5
```

- Индексирование фрейма данных по имени столбца:

```
df$Name

## [1] "Cat" "Dog" "Cow" "Pig"
```

- Доступ к подстрокам фрейма с использование оператора равенства:

```
df[df$IsPet == TRUE, ]

##   Name HowMany IsPet
## 1  Cat       5  TRUE
## 2  Dog      10  TRUE
```

Задание:

Создать новый проект: File → New Project → New Directory → New Project → Вводим имя новой директории R_Lab1 и указываем путь к папке, где будут храниться лабораторные работы. → Create project.

При выполнении данных команд у вас должен создаться новый проект. Далее создадим новый R Script, где непосредственно будут выполняться лабораторные работы. File → New File → R Script или комбинацией клавиш Ctrl + Shift + N. В появившемся окне будет вводиться код. Каждая команда с новой строки. Для выполнения команды следует нажать на кнопку Run (Ctrl + Enter).

Результаты выполнения будут отражаться в поле **Console**, **Enviroment** или **Plots**.

1. Создать логическую переменную и вывести её;
2. Создать и вывести целочисленную переменную типа int;
3. Создать и вывести нецелочисленную переменную типа float;
4. Создать и вывести строковую переменную типа string;
5. Создать и вывести вектор;
6. Создать и вывести последовательность, состоящую из 20 чисел.

   Пример создания последовательности:

   ```
   s <-1:5
   ```

7. Создать вектор, содержащий случайные числа в диапазоне от -5 до 15. Для создания вектора со случайными значениями можно использовать функции runif() или sample();
8. Вывести значения вектора, созданного в пункте **7**, которые будут больше 1, но меньше 10;
9. Вывести максимальное и минимальное значения вектора, созданного в пункте **7**;
10. Создать последовательность, используя функцию seq() со следующими параметрами: диапазон изменения: от -15 до 15; шаг 0.01;
11. Создать и выполнить функцию. Функция должна принимать один аргумент и возвращать результат выполнения операции $f(x) = 5 \cdot \sin(x^2)$.

    Пример использования функции:

    ```
    z <– function( переменная которая передаётся)
    {
        тело функции
    }

    z(значение)
    ```

12. Отобразить графическую зависимость используя функцию plot(), где в качестве аргументов использовать величины, реализованные в пунктах **10** и **11**. **За что отвечает аргументы функции plot(), такие как: type, col, lty и pch?**

13. Построить трехмерный график поверхности $f(x,y) = \frac{x}{2} \cdot exp\left(-\frac{x^2+y^2}{2}\right)$, со следующими параметрами: левая граница для x и y **-5**; правая граница для x и y **5**; шаг по осям выбрать равным **0,2**.

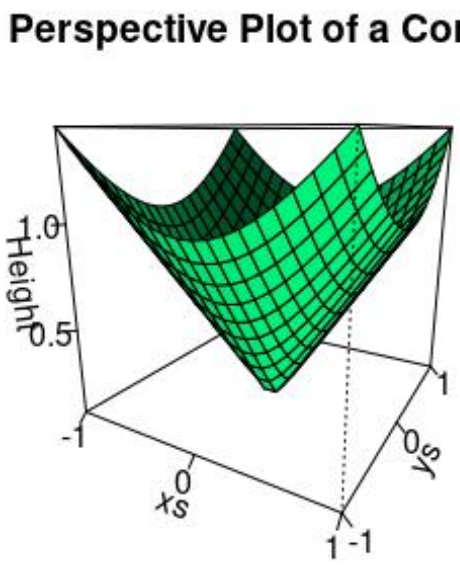
Пример построения поверхности:

```
xs <- seq(-1, 1, length= 20)
ys <- seq(-1, 1, length= 20)

cone <- function(x, y)
  {
    sqrt(x^2+y^2)
  }

z <- outer(xs, ys, cone)

persp(xs, ys, z,
      main="Perspective Plot of a Cone",
      zlab = "Height",
      theta = 30,
      phi = 15,
      col = "springgreen",
      shade = 0.5,
      ticktype='detailed',
      nticks=3)
```



14. Создать и вывести матрицу размером 10 x 10 и заполненную последовательностью от 1 до 100. Матрица создаётся при помощи функции matrix() с параметрами data – значения в матрице, nrow – количество строк, ncol – количество столбцов;

15. Определить число строк и столбцов матрицы;

16. Создать и вывести массив, заполненный последовательностью от 1 до 30, и разбитый на 3 части по 10 значений.

Массив создаётся при помощи функции array() с параметрами data – значения в массиве и dim – вектор.

```
a <- array(data = 1:8,dim = c(2, 2, 2))
print(a)

## , , 1
##
##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4
##
## , , 2
##
##      [,1] [,2]
## [1,]    5    7
## [2,]    6    8
```

17. Создать и вывести список, который содержит как минимум одно значение логического типа, целочисленного, нецелочисленного и строку. Для создания списков используется функция list() в которую передаются значения. Список может содержать значения абсолютно любого типа;

18. Создать вектор, который содержит 5 различных строковых переменных, каждая из которых содержится минимум 2 раза. Создать фактор, используя функцию factor() и передавая в качестве аргумента созданный вектор;

19. Используя функцию levels() вывести уровни фактора;

20. Сравнить результаты выполнения функций table() и unclass(), где в качестве аргумента выступает фактор;

21. Создать и вывести фрейм данных на тему транспорта, который будет состоять из **7** строк и **4** колонок, при этом колонки должны содержать различные типы данных (каждая колонка содержит свой тип данных);

22. Вывести последнее значение из последнего столбца фрейма данных;

23. Вывести первую и последнюю строки фрейма данных. Для вывода строки/колонки необходимо указать индекс строки/столбца, а поле для столбца/строки оставить пустым;

24. Вывести столбец, индексируя его по имени.

Например, df[[„name_of_col“]];

25. Вывести столбец индексируя его по имени в сокращённой форме. Для этого имя столбца передаётся через символ $ к переменной фрейма. Например, df$name_of_col;

26. Вывести подмножество чётных строк фрейма, используя вектор индексов. Для этого нужно передать вектор с индексами в качестве строкового индекса при обращении к фрейму;

27. Вывести подмножество чётных столбцов фрейма, используя вектор индексов;

28. Вывести подмножество первых 4 строк фрейма, используя последовательность индексов. Для этого нужно передать последовательность в качестве строкового индекса при обращении к фрейму;

29. Используя оператор равенства вывести подмножество строк. Для этого в строковый индекс, при обращении к фрейму, можно передать колонку с выбранным именем в сокращенной форме и сравнить её с нужным значением.

Например, df[df$ name_of_col == TRUE,];

30. Вывести подмножество строк, используя оператор соответствия.

Например: df[df$ name_of_col %in% c("Name_1", "Name_2"),]

Содержание отчёта

Файл отчёта должен содержать полный код программы с описанием производимых действий.