Санкт-Петербургский государственный университет
Прикладная математика, информатика и искусственный интеллект

Отчет по учебной практике 1 (научно-исследовательская работа)

(семестр 1)

Прохождение курса по R на stepik

Выполнила:
Барабашева Анастасия Дмитриевна,
группа 22.Б04-мм

Научный руководитель:
кандидат физико-математических наук, доцент
Голяндина Нина Эдуардовна
Кафедра статистического моделирования

Работа выполнена на хорошем уровне
и может быть зачтена с оценкой А

Санкт-Петербург
2023

I. Введение

В ходе работы, я прошла курс «Основы программирования на R» на сайте stepik.org по ссылке [1]. Я получила сертификат с отличием, мои конспекты и решения некоторых задач находятся в [2]. В ходе курса я подробно изучила понятие «векторизация» и использовала полученные знания для исследования её достоинств.

II. Основная часть

1. Прохождение курса

Курс разделён на три модуля. Первый модуль содержал задачи на базовые структуры языка R, такие как вектор, условный оператор, цикл. Во втором рассматривались более продвинутые структуры: матрицы, списки, дата фреймы, факторы и строки. В задачах использовались встроенные дата фреймы attitude и quakes, а также таблица avianHabitat с данными о растениях Аляски.
Третий модуль был посвящён продвинутому программированию. В нём рассматривались элементы функционального программирования в языке R, а также способы обработки данных при помощи пакетов tidyr и dplyr. Ниже я приведу описания и решения самых интересных и сложных для меня задач.

Задача 1.
Одной из самых сложных и интересных задач была для меня последняя задача в курсе. Она заключалась в том, чтобы по всем комбинациям места (Site) и наблюдателя (Observer) и вида (Species) посчитать количество ненулевых наблюдений. Для этого нужно было привести данные к виду tiny data, и далее - получить необходимую статистику. Код решения приведён ниже.

```
avian <- read.csv("avianHabitat.csv")
library(dplyr)
library(stringr)
library(tidyr)

result <- avian %>%
  select(Site, Observer, contains("Ht")) %>%
  mutate(Site = factor(str_replace(Site, "[:digit:]+", ""))) %>%
  gather(Species, Height, -Site, -Observer) %>%
  mutate(Species = factor(str_replace(Species, "Ht", ""))) %>%
  group_by(Site, Observer, Species) %>%
  summarise(K = sum(Height > 0))
```

Задача 2.
Интересной была задача на казино. Надо было написать две рулетки. Честная рулетка должна выдавать все имеющиеся значения (всего их 37) с равной вероятностью. Нечестная рулетка должна выдавать все значения, кроме нуля, с равной вероятностью. Вероятность выпадения нуля должна быть в два раза больше, чем любого другого значения. Для написания этих функций я использовала функцию *sample*(). Функция *sample*() выбирает случайные элементы из вектора или списка без повторений. Мы можем использовать эту функцию для выбора случайных значений из вектора roulette-values. Функция для честной рулетки:

```
fair_roulette <- function(n) {
  roulette_values <- c("Zero!", 1:36)
  sample(roulette_values, n, replace = TRUE)
}
```

А вот для нечестной:

```
rigged_roulette <- function(n) {
  roulette_values <- c("Zero!", 1:36)
  probs <- c(2, rep(1, length(roulette_values) - 1))
  probs <- probs / sum(probs)
  sample(roulette_values, n, replace = TRUE, prob = probs)
}
```

В первой строке каждой функции мы создаем вектор roulette-values, содержащий все возможные значения на рулетке. Затем мы создаем вектор вероятностей *probs*, который определяет вероятности выпадения каждого значения. В честной рулетке все значения имеют одинаковую вероятность выпадения, поэтому мы используем вектор вероятностей из единиц. В нечестной рулетке мы устанавливаем вероятность выпадения нуля в два раза больше, чем любого другого значения. Затем мы используем функцию *sample*() для выбора случайных значений из вектора *roulettevalues*. В честной рулетке мы устанавливаем аргумент replace = TRUE, чтобы выбирать значения с повторениями (то есть одно и то же значение может выпадать несколько раз). В нечестной рулетке мы также используем аргумент *prob*, чтобы указать вероятности выпадения каждого значения.

Задача 3.
Интересно было решать задачу про котов. Нужно было, имея несколько списков с характеристиками котов, составить всевозможные комбинации, отсортировать их и выбрать нужное значение. Код для получения каталога для котов:

```
cat_catalogue <- expand.grid(cat_temper, cat_color, cat_age, cat_trait)
cat_catalogue <- apply(cat_catalogue, 1, paste0, collapse = ", ")
cat_catalogue <- sort(cat_catalogue)

cat_catalogue
```

Задача 4.
Навык работы со строками я оттачивала на задачах подобного типа:

Пусть функция *decoratestring* действует поверх функции *paste*, дополнительно приклеивая к результату аргумент *pattern*. При этом этот аргумент должен быть присоединён как в начале строки (строк), так и в конце, но перевёрнутый задом наперёд. Ее решение:

```
decorate_string <- function(pattern, ...) {
    paste(paste(pattern[1], paste(...), sep = ""),
    stringi::stri_reverse(strsplit(pattern, " ")[1]), sep = "")
}
```

2. Векторизация

Характерной особенностью R является векторизация вычислений. Векторизация представляет собой один из способов выполнения параллельных вычислений, при котором программа определенным образом модифицируется для выполнения нескольких однотипных операций одновременно. Такой подход потенциально может привести к значительному ускорению однотипных вычислений над большими массивами данных. Векторизация - это процесс преобразования последовательности операций в параллельную форму, которая может выполняться на векторном процессоре. Векторизация позволяет ускорить выполнение программы за счет одновременного выполнения нескольких операций над элементами массива. Циклы же выполняются последовательно и могут быть неэффективными при работе с массивами. Таким образом, векторизация может быть быстрее цикла за счет параллельного выполнения операций над элементами массива.

Посмотрим, во сколько раз векторизация ускоряет код на примере. Время исполнения блока кода будет отслеживать с помощью функции system.time().

```
v <- 1:1e5
system.time({
    x <- 0
    for (i in v) x[i] <- sqrt(v[i])
})
```

Время выполнения примерно 27 с. Сравним с

```
system.time({
    y <- sqrt(v)
})
```

Время выполнения примерно 0.004 с. Теперь рассмотрим на примере сложения двух векторов. Код без использования векторизации:

```
x <- rnorm(1000000)
y <- rnorm(1000000)
z <- rep(0, length(x))


system.time({
    for (i in 1:length(x)) {
    z[i] <- x[i] + y[i]
    }
})
```

Время выполнения 0.07 с.
Код с векторизацией:

```
x <- rnorm(1000000)
y <- rnorm(1000000)

system.time(z <- x + y)
```

Время выполнения 0.006 с.

Ускорение кода за счет векторизации работает. Более того, векторизация увеличивает читабельность кода. Векторизация такая быстрая за счет определенной работы с памятью. Она выполняет операции над целыми векторами данных, что позволяет использовать кэш-память более эффективно и уменьшить количество обращений к памяти. Векторизованные операции выполняются быстрее, чем циклы, потому что они используют меньше операций чтения/записи в память.

III. Заключение

В ходе прохождения курса, я научилась основам программирования на R. Эти знания и дальнейшая практика помогут больше разобраться в языке и анализе данных.
R является действительно удобным инструментом для выполнения таких задач, и имеет множество преимуществ. Одним из них я могу назвать высокоуровневость языка R. Такие особенности, как, например, конвееры (из пакета dplyr) или векторизованность большинства функций, позволяют быстро и эффективно (с точки зрения написания кода) выполнять операции над данными.

IV. Список литературы

1. https://stepik.org/course/497/ - курс “Основы программирования на R”
2. https://docs.google.com/document/d/1AJ6Ig9hrmZg930io4brNvdBOG9gMJ5Yc_4KZOvJhw1c/edit?usp=sharing - гугл-документ с конспектами и решениями некоторых задач
3. https://ggplot2.tidyverse.org/reference/index.html - документация пакета ggplot2, для визуализации данных
4. https://stackoverflow.com/questions/tagged/r - раздел форума, посвящённый языку R

V. Приложение

stepik

BIOINFORMATICS INSTITUTE

Результат

92%

С отличием

Настоящий сертификат подтверждает, что

Настя Барабашева

успешно завершил/а курс

Основы программирования на R

Антон Антонов

https://stepik.org/course/497 https://stepik.org/cert/2040988

23.04.2023