Лабораторная работа №5

ФУНКЦИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ (ИТЕРАЦИИ И КОНВЕЙЕРЫ)

Цель: познакомиться с особенностями функционального программирования. Научиться применять функциональное программирования с использованием пакета purrr. Решить задания в соответствующем стиле программирования. Составить отчет.

Теоретические сведения

Язык R поддерживает функциональную парадигму программирования, однако для лучшей ее реализации, и для увеличения функционала, рекомендуется использовать пакет purrr, кроме того, необходимо также скачать и подключить пакет repurrrsive, содержащий наборы данных для выполнения задания №1.

Итерации

Итерация – организация обработки данных, при которой действия повторяются многократно. В программировании итерации чаще рассматриваются в качестве элементов структурного программирования, а именно как единичный шаг выполнения цикла. На практике итерации также важны и в функциональном программировании, например в качестве итерабельного процесса можно рассматривать применение одной и той же функции к разным элементам. Рассмотрим пример выполнения кода, написанного в структурном и функциональном стиле над однотипным набором данных.

```
# Создание тестового списка
iris_list <- as.list(iris[1:4])

# Реализация в структурном стиле
iris_mean <- list()
for (i in seq_along(iris_list)) {
  iris_mean[[i]] <- mean(iris_list[[i]])
}

# Реализация в функциональном стиле через map
iris_mean <- map(iris_list, mean)
```

Рисунок 1 – Сравнение функциональной и структурной реализации

У функции map есть два варианта реализации (рисунок 2).

```
# Реализация map без указания точки входа в функцию
iris_mean <- map(iris_list, mean)

# Реализация map с указанием точки входа в функцию
iris_mean <- map(iris_list, ~ mean(.x))
```

Рисунок 2 – Варианты синтаксиса map

Второй вариант реализации позволяет показать место передачи элемента листа (“.x”) через знак тильда (“~”). Второй способ является предпочтительнее для реализации.

Вариации функции map. map_*

Функция map на выходе выдает список, аналогичной структуры, однако часто необходимо сразу преобразовать вывод в некоторый вариант, для этого используются производных от функции map:

```
map(.x, .f, ...)
map_if(.x, .p, .f, ...)
map_at(.x, .at, .f, ...)
map_lgl(.x, .f, ...)
map_chr(.x, .f, ...)
map_int(.x, .f, ...)
```

```
map_dbl(.x, .f, ...)
map_dfr(.x, .f, ..., .id = NULL)
map_dfc(.x, .f, ...)
walk(.x, .f, ...)
```

```r
# Создание именованного числового вектора с помощью map_dbl
iris_mean <- map_dbl(iris_list, ~ mean(.x))
```

Рисунок 3 – Создание именованного числового вектора с помощью map_dbl

Конвейеры (pipeline)

Конвейер – инструмент для передачи значения исходной функции в последующую. Конвейер представляет типичный элемент функционального программирования. В языке R конвейер имеет вид %>%. Синтаксис конвейера представлен на рисунке 4.

```r
function_before() %>%
  function_after()
```

Рисунок 4 – Синтаксис конвейера

Практическая часть

Задание 1. Используя тестовые данные пакета repurrrsive выполните следующее задание. Создайте именованный список аналогичный по структуре списку sw_films, для установления имени полезно использовать функцию set_names пакета purrr. В качестве имени элементов списка необходимо использовать соответствующие название фильмов (обратите внимание, что обращаться к элементам списка можно используя как индекс, так и название элемента). Выполните задание в функциональном стиле.

Задание 2. Используя документацию пакета purrr опишите отличия и особенности функций семейства map_*. Приведите примеры реализации с использование различных тестовых данных. Данные можно брать из пакета datasets или создав свои тестовые наборы. Для просмотра данных из пакета datasets выполните код library(help = "datasets")