**УДК 004.423.22** **Информационные технологии**

***Яблонцева Арина Дмитриевна,*** *студент направления подготовки информатика и вычислительная техника, Хакасский государственный университет имени Н.Ф. Катанова, г. Абакан, Россия*

***Голубничий Артем Александрович,*** *научный руководитель, старший преподаватель кафедры ПОВТиАС, Хакасский государственный университет имени Н.Ф. Катанова, г. Абакан, Россия*

# СРАВНЕНИЕ ПАКЕТА MAGRITTR И БАЗОВОГО PIPE В ЯЗЫКЕ R

**Аннотация:** В статье рассматриваются операторы типа «pipe», реализованные посредством пакета magrittr и базового пакета языка R версии 4.1+.

**Ключевые слова:** pipe, язык R, magrittr, оптимизация вычислений, оптимизация разработки.

**Annotation:** This article discusses the "pipe" type operators implemented by the magrittr package and the base package of the R language version 4.1+.

**Keywords:** pipe, R language, magrittr, computation optimization, development optimization.

Оператор «pipe» – одна из самых отличительных особенностей кода, написанного в стиле tidyverse/dplyr [1], он также получил широкое распространение во многих пакетах, в том числе, при работе с картами в leaflet и многих других, предполагающих передачу значений последующим функциям. Ранее для написания программного кода в стиле оператора «pipe» использовался magrittr, пример реализации приведен в листинге на рисунке 1.

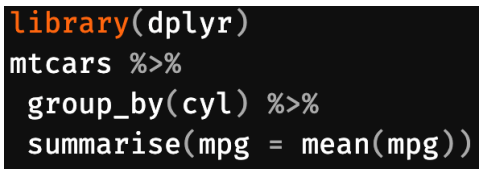


Рисунок 1 – Пример листинга кода оператора pipe в пакете dplyr

Конструкция %>% является оператором, который позволяет соединять передать результат вызова первой функции последующей без необходимости присваивать промежуточную переменную и без использования вложенных скобок. Технически то, что он выполняет является вычислением выражения в правой части pipe (или в следующей строке), используя выражение слева (или в той же строке) в качестве первого аргумента. Пакет dplyr зависит от пакета magrittr, который делает всю эту работу, такая же логика реализована во многих других пакетах.

Начиная с версии 4.1.0 можно писать без код без подключения библиотеки magrittr. Пример работы приведен в листинге на рисунке 2 (здесь и далее в примерах используется шрифт с лигатурами, что приводит к замене некоторых конструкций на их аналоги).

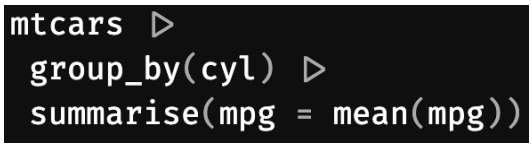


Рисунок 2 – Пример листинга кода оператора pipe в базе языка R

Основное отличие заключается в том, что теперь можно использовать pipe, не полагаясь на пакет magrittr. Подключение пакета обычно не вызывает сложностей, однако, как правило, всегда желательно, чтобы ваш анализ зависел от как можно меньшего количества различных пакетов. Чем больше зависимостей, тем выше вероятность того, что какое-то обновление изменит что-то важное, что разрушит все, что было сделано ранее.

Для тех, кто использует dplyr, |> и %>% скорее всего взаимозаменяемы. Также конструкция |> быстрее в наборе в сравнении с %>%. При этом magrittr

делает много вещей «за кулисами», в то время как нативный pipe – это просто преобразование синтаксиса. Но реальность такова, что, за исключением очень особых случаев, разница незначительна. Исходный код для сравнения скорости операторов и результат сравнения приведены на рисунках 3,4.

```
library(magrittr)
library(microbenchmark)
res_m ← function() {
 mtcars %>%
 group_by(cyl) %>%
 summarise(mpg = mean(mpg))
}
res_p ← function() {
 mtcars ▷
  group_by(cyl) ▷
  summarise(mpg = mean(mpg))
}
microbenchmark(res_m, res_p, times = 50000)
```

Рисунок 3 – Исходный код, для проведения бенчмаркинга

```
> microbenchmark(res_m, res_p, times = 50000)
Unit: nanoseconds
  expr min lq     mean median uq   max neval
 res_m  21 27 33.74474     28 36 16416 50000
 res_p  22 28 34.22114     29 37 12633 50000
> microbenchmark(res_m, res_p, times = 50000)
Unit: nanoseconds
  expr min lq     mean median uq   max neval
 res_m  22 27 34.13050     29 37 16431 50000
 res_p  23 28 35.06328     30 37 31436 50000
```

Рисунок 4 – Результаты бенчмаркинга

Как видно из рисунка 4 результаты отличаются в незначительной степени, и в случае повторных запусков могут отличаться.

При любом анализе данных накладные расходы на использование magrittr малы по сравнению со временем, которое требуется для выполнения (и записи) остальных вычислений. На что действительно нужно обратить внимание, так это на различия между двумя pipe. Единственным наиболее важным отличием

является то, что в magrittr есть элемент-заполнитель, когда кто-то не хочет, чтобы левый результат переходил к первому аргументу правого выражения. Каноническим примером является линейная модель, листинг кода реализации, которой приведен на рисунке 5.

```
mtcars %>%
 lm(mpg ~ disp, data = .)
```

Рисунок 5 – Листинг кода реализации линейной модели (оператор %>%)

Поскольку первый аргумент lm() это не данные, необходимо использовать data = . для того чтобы сказать magrittr, что mtcars не обязательно должен быть первым аргументом lm(). Нативный канал в настоящее время не имеет заполнителя, для реализации аналогичной логики необходимо создать анонимную функцию (рисунок 6).

```
mtcars |>
 (function(x) lm(mpg ~ cyl, data = x))()
```

Рисунок 6 – Листинг кода реализации линейной модели (оператор |>)

Внешне реализация, представленная на рисунке 6 выглядит не эстетично, поэтому в R версии 4.1 добавлен новый синтаксис по созданию функций. Начиная с R 4.1.0 эти два выражения эквивалентны: function(x) x + 1 и \(x) x + 1

Новый синтаксис «машущего человека» \(x) существенно экономит символы при создании функций. Итак, совместив это с pipe, можно реализовать код, приведённый на рисунке 7.

```
mtcars |>
 (\(x) lm(mpg ~ disp, data = x))()
```

Рисунок 7 – Листинг кода реализации линейной модели в синтаксисе «машущего человека» (оператор |>)

Этот вариант немного более читабелен, но до сих пор выглядит не удобно для реализации. Альтернативный синтаксис приведён в листинге на рисунке 8.

```
Sys.setenv(`_R_USE_PIPEBIND_` = TRUE)
mtcars |>
  . => lm(mpg ~ disp, data = .)
```

Рисунок 8 – Листинг альтернативного кода реализации линейной модели в синтаксисе «машущего человека» (оператор |>)

Таким образом, сначала устанавливается символ-заполнитель (в данном случае «.») и после = >, можно также написать тот же код, что и в канале magrittr. Подведя итоги можно сказать, что для случаев, когда заполнитель «.» используется, заменой для % > % было бы | > . = >..

В итоге программа выведет следующий идентичный результат, показанный на рисунке 9.

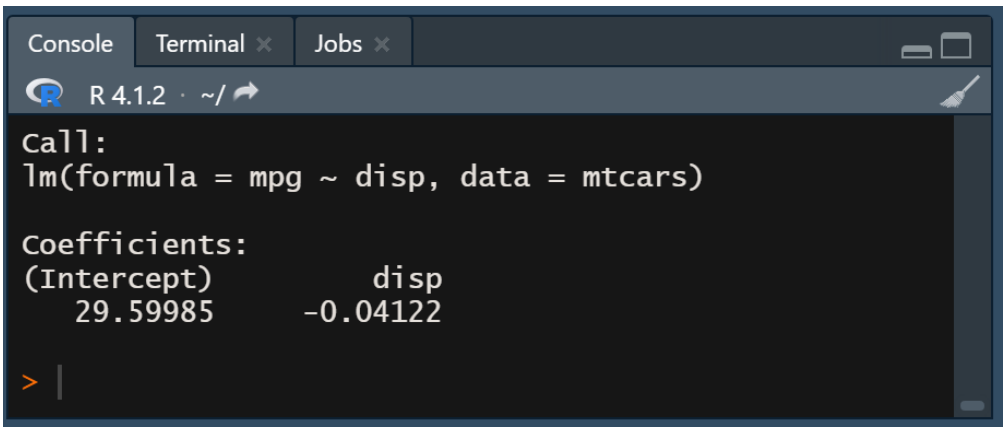


Рисунок 9 – Итоговый результат вывода кода рисунков 7-8

Далее на рисунке 10 представлен код, совмещающий data.table и magrittr.

```
mtcars <- data.table::as.data.table(mtcars)
mtcars %>%
  .[, .(mpg = mean(mpg)), by = cyl]
```

Рисунок 10 – Листинг кода, совмещающего data.table и magrittr

Учитывая, что точка в начале первой строки является не чем иным, как заполнителем magrittr, и что новый pipe не имеет заполнителя, можно догадаться, что этот синтаксис не будет транслироваться в родной канал, просто изменив %>% на |>. Есть также некоторые ограничения, например, новый pipe не принимает «специальные символы», такие как [, + или * в правильном выражении.

Однако синтаксис => является тонким, поэтому можно было бы подумать, что правильным переводом было бы добавить |> .=> , но к сожалению, это не так. Результат работы такой программе показан на рисунке 11.

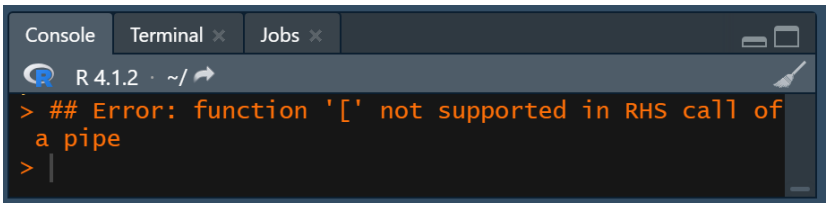


Рисунок 11 – Пример выполнения кода с добавлением |> .=>

У нас появилось ограничение специальных символов. Это происходит потому, что R смотрит только на имя функции, поэтому все, что нужно сделать, это переименовать функцию -[. Например, код, представленный в листинге на рисунке 12, идеально подходит для пояснения действий.

```
.d ← `[`
mtcars ▷
 .d(, .(mpg = mean(mpg)), by = cyl)
```

Рисунок 12 – Листинг кода с переименованной функцией

**Заключение**

Версия R 4.1.0 вышла не так давно, и вполне вероятно, что большинство пользователей еще не обновлялись и не планируют обновляться в ближайшее время. В корпоративных или серверных средах многие пользователи R, вероятно, даже не имеют контроля над тем, какую версию они устанавливают, а администраторы, вероятно, довольно неохотно обновляются [2]. «Код

производственного уровня», работающий в определенных версиях R для обеспечения стабильности и воспроизводимости, вероятно, потребует годы для обновления. По всем этим причинам, хотя дни magrittr действительно сочтены, это произойдёт ещё не скоро.

**Библиографический список:**

1. R: The R Project for Statistical Computing [Электронный ресурс] URL: https://www.r-project.org (дата обращения 14.02.2023).

2. Хакимова, Т. В. Популярность и перспективы языка программирования R / Т. В. Хакимова, А. А. Голубничий // E-Scio. – 2019. – № 6(33). – С. 817-821. – EDN APCDKU.