

Федеральное государственное образовательное бюджетное учреждение
высшего профессионального образования
«ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»
(Финансовый университет)

Департамент математики

Дисциплина «Программирование в среде R»

П.Б. Лукьянов

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ
ЛАБОРАТОРНОЙ РАБОТЫ № 1

Скрипты и переменные R

Для студентов, обучающихся по направлению подготовки
«Прикладная информатика»
(программа подготовки бакалавра)

Москва 2021

Цель лабораторной работы заключается в изучении

- возможностей работы с консолью
- пакетной обработки команд с использованием скриптов R
- в написании простейших программ в среде RStudio.

1. Консоль среды R – умный калькулятор

Консоль R удобно использовать в качестве умного калькулятора: можно писать формулы любой сложности, вызывать математические функции и получать результаты с высокой точностью, которые затем можно сохранять в переменные для последующего использования.

В среде RStudio в строке ввода после приглашающего значка `>` как правило, вводится одна команда. После нажатия `Enter` эта команда сразу выполняется и выводится результат выполнения этой команды. Например:

```
> 1 + 2 + 4 # наша команда – выполнить сложение
```

```
[1] 7          # ответ среды R
```

В строке можно ввести несколько команд, если разделять их точкой с запятой. В случаях, когда команда длинная (например, вызывается функция с большим количеством параметров), можно нажать `Enter` и продолжить ввод на следующей строке. В этом случае слева вместо значка `>` появляется `+`.

Клавиши «Стрелка вверх», «Стрелка вниз» позволяют перемещаться по ранее введенным командам, а клавиши «Стрелка влево», «Стрелка вправо» используются для редактирования текущей команды. История всех команд сохраняется, поэтому для повторного вызова набранной ранее команды достаточно нажать клавишу «Стрелка вверх» нужное число раз.

Проверьте работу R в режиме калькулятора. Наберите в строке ввода несколько простейших математических выражений, в конце нажмите `Enter`:

```
5 * 12 / 3 + 0.1
```

```
sin(54); cos(45); sin(54) + cos(45);
```

```
sqrt(abs( sqrt(-9 * (-5)) / (-27.7^2)))
```

```
11 %/% 3; 11 %% 3
```

Обратите внимание, что в качестве десятичного разделителя используется точка; количество пробелов в выражениях может быть любым, имена функций обычно пишутся с маленькой буквы. Если в командной строке есть разделитель точка с запятой, то результаты выводятся независимо друг от друга.

2. Арифметические операторы

Для операций с числами и числовыми переменными используются обычные операторы сложения, вычитания, умножения, деления и возведения в степень. Кроме этого, есть операция целочисленного деления (`%/%`) и операция получения остатка от деления (`%%`). Операции имеют обычный приоритет, т.е. сначала выполняется возведение в степень, затем умножение или деление, затем сложение или вычитание. Если используются круглые скобки, то операции в скобках выполняются в первую очередь. Все арифметические операторы представлены в Таблице 1.

Таблица 1. Арифметические операторы

+	Сложение
—	Вычитание
*	Умножение
/	Деление
^	Возведение в степень
%/%	Целочисленное деление
%%	Остаток от деления

3. Первая программа на R

В языке R файлы с командами языка называются скриптами (сценариями) и имеют расширение R (например, test12.R, exDefDeviation.R, hello.R и т.д.). По сути, это обычные текстовые файлы, и они могут быть открыты любым текстовым редактором. По расширению .R среда R видит «свои» файлы, быстро их загружает и запускает на выполнение. Вместе с тем, код на языке R может храниться в файле с любым расширением, в этом случае загрузка скрипта в R будет происходить дольше.

Подготовим первый скрипт и запишем в нем код программы, выводящей сообщение “Это моя первая программа!”.

Для хранения своих собственных скриптов создайте на одном из локальных дисков компьютера (c:\, d:\, ...) или сетевом диске папку для работы с R, назовите ее lessons.R и перейдите в нее. Папка должна быть пуста.

Запускаем RStudio, выбираем меню File– NewFile–RScript. Появляется пустое окно, в котором нужно написать строчку

```
print(“Это моя первая программа!”)
```

Запуск содержимого скрипта может быть выполнен несколькими способами:

1. Установить курсор на нужной строке и нажать сочетание Ctrl-Enter. Будет выполнена эта строчка, курсор перейдет на следующую строку с командой, если она есть.
2. Выделить или весть код (Ctrl-A), или несколько строк кода и нажать сочетание Ctrl-Enter. Весь выделенный код будет выполнен построчно.
3. Вместо сочетания клавиш можно нажимать на кнопку Run в верхней части среды RStudio.

Сейчас результат работы программы выводится в консоль (специальное текстовое окно для отображения результатов), и постепенно

консоль будет забиваться разными результатами и сообщениями. Для удаления всех строк из консоли используется сочетание Ctrl-L.

Сохраним скрипт в папку lessons.R (File – Save). После сохранения в папке lessons.R должен появиться файл с расширением R. Для загрузки скрипта используйте пункт меню File – Open File или соответствующую пиктограмму на панели инструментов.

Добавим в программу несколько строк. Откройте скрипт и скопируйте строчку программы несколько раз, добавив для вывода номер строки и комментарий:

```
print(“Это моя первая программа! 1”)    # первая строка
print(“Это моя первая программа! 2”)    # вторая
print(“Это моя первая программа! 3”)    # третья
...
print(“Это моя первая программа! 10”)   # последняя строчка
```

Запустите всю программу. Запустите только строки 2 и 3.

Отдельно следует сказать об именовании файлов со скриптами: имя файла может быть на любом языке, но, если файл назван русскими буквами, то на другом компьютере Вы можете увидеть непонятные крючки вместо имени на русском языке. Эта проблема связана с различной кодировкой символов на разных компьютерах. Поэтому лучше для имен файлов использовать символы английского алфавита.

2. Комментарии в программе

Код, который вам понятен сегодня, но в котором нет комментариев, через неделю превращается в сложный кроссворд. Поэтому общепринятой практикой программистов является документирование всех ключевых действий программы посредством написания комментариев. Комментарии пишутся программистом для удобства сопровождения программы и ее

возможного последующего развития. Всегда оставляйте комментарии, какой бы простой разрабатываемая программа вам не казалась.

Комментарии в языке R начинаются со знака # и продолжаются до конца строки. Текст комментариев может быть на любом языке, может занимать любое количество строк.

Не нужно беспокоиться о том, что комментарии увеличивают программу или замедляют работу. При запуске программы ее код проходит предварительную обработку, и из него автоматически удаляются все пробелы и комментарии, и лишь затем программа выполняется.

3. Имена переменных и констант в языке R

Язык R чувствителен к регистру, поэтому будьте внимательны при вводе команд, использовании функций и переменных. Так, функции `makeSpecialActions()` и `MakeSpecialActions()` различны. Аналогично, переменные `sumXY` и `sumXy` – это разные переменные.

Проще всего переменные представлять как ящики разного размера, в которых хранятся различные данные; все ящики подписаны, у каждого ящика есть имя. Другими словами, переменная – это именованная ячейка в памяти компьютера, выделенная для хранения некоторой величины.

В любом языке программирования используются переменные и константы. В R имена переменных и констант могут быть как на латинице, так и на кириллице, ошибки здесь нет. Но если предположить, что в будущем ваши программы будут включены в международные проекты, то для всех имен рекомендуется использовать символы английского языка.

Переменные используются для хранения данных, значения которых могут быть изменены, в этом их отличие от констант. Переменные и константы однозначно определяются своими именами, и при задании имен в R есть свои правила.

Имя – это непрерывная последовательность букв, цифр, точек и символов подчеркивания. Начинать имя переменной с цифры или символа подчеркивания нельзя. В имени переменной могут использоваться точка и символ подчеркивания. В настоящее время в программировании отказываются от использования подчеркивания в пользу точки в качестве смыслового разделителя.

Например, точку удобно использовать в таком сложном имени переменной: `local.param.for.regression`. Без точки сложное имя принято конструировать так: `localParamForRegression`, это так называемый CamelCase – один из стандартов оформления кода. Заглавные буквы для первого символа переменной обычно не используют.

Если имя начинается с точки, то второй символ не может быть цифрой. Как уже отмечалось, прописные и строчные символы различаются: `res`, `Res`, `RES` – это разные имена.

Корректные имена в R:

`result`, `Sum`, `.everage.value`, `is_true`, `File29_5`

Недопустимые имена в R:

`1ff`, `min@1`, `14days`, `_no_`, `TRUE`, `.02`

Еще одно ограничение при конструировании имен переменных и констант заключается в следующем: для имени нельзя использовать **зарезервированные** слова, т.е. слова, которые уже имеют вполне определенный смысл в языке R. Ниже представлен список зарезервированных слов:

`if`, `else`, `repeat`, `while`, `function`, `for`, `in`, `next`, `break`, `TRUE`, `FALSE`, `NULL`, `Inf`, `NaN`, `NA`, `NA_integer`, `NA_real`, `NA_complex`, `NA_character_`

Часть этих слов (`if`, `else`, `repeat`, `while`, `function`, `for`, `in`, `next`, `break`) используется в управляющих конструкциях языка, таких как проверка условий, циклы, функции, заданные Пользователем.

`TRUE` и `FALSE` – логические константы.

NULL представляет собой «ничто». NULL характеризует пустое, неопределенное значение. Например, при обращении к несуществующему элементу списка мы получим в качестве результата NULL.

Inf – бесконечность (infinity); как правило, результат деления числа на ноль. Бесконечность может быть как положительная ($1/0$), так и отрицательная ($-1/0$).

NaN – неопределенный результат (Not a Number). NaN возвращается при операциях над числами, результат которых не определён (не является числом).

NA – отсутствующее значение (Not Available). NA возникает, если значение некоторого объекта не доступно или не задано. NA используется для замещения отсутствующих значений, например, при анализе данных. Так, при обращении к несуществующему элементу массива мы получим в качестве результата NA.

7. Операторы присваивания

Рассмотрим пример:

```
b <- 5.1
```

Создается переменная `b` действительного типа, ее значение 5.1.

Вместо присваивания с помощью `<-` (для быстрого набора присвоения в RStudio используется сочетание `Alt-`) можно пользоваться как традиционным равенством `=`, так и присваиванием в другую сторону с помощью `->`:

```
x = 87
```

```
87 -> x
```

Можно создавать и такие конструкции:

```
r <- t -> m
```


Проверьте, что задавать значения переменным можно всеми этими способами. Общепринятым оператором присваивания в R является `<-`, именно им и будем пользоваться в дальнейшем.

Одно значение можно присвоить сразу нескольким переменным:

```
a <- b <- c <- 5.05
```

В языке есть еще два редко используемых оператора присваивания:

```
x <<- 10
```

```
10 ->> x
```

Присваивание значений переменным с использованием этих операторов используется в функциях, создаваемых Пользователем. Такое присвоение позволяет переопределять значения глобальных переменных (переменных, видимых во всех частях программы), находясь внутри функции. Результат использования этих операторов не такой очевидный, что связано с понятием глобальной среды, поэтому оставим этот вопрос для рассмотрения в разделе, посвященном функциям, создаваемым Пользователями.

Мы можем создать переменную, мы можем ее и удалить. Удаление выполняется с помощью функции `rm(имя_переменной)`. Создайте несколько переменных разных типов, проверьте значения и тип переменных, затем удалите некоторые из них.

Как убедиться в том, что переменная действительно была удалена?

8. Считывание данных с клавиатуры

Большинство программ должно так или иначе взаимодействовать с Пользователем – реагировать на его команды, получать данные, менять алгоритм работы в зависимости от выбора Пользователя. В программах с графическим интерфейсом стандартом является взаимодействие с Пользователем через систему меню, различные окна и формы ввода.

На первом этапе для обеспечения интерактивности в своих программах нужно научиться решать более простую задачу:

- считывать ввод Пользователя с клавиатуры
- обрабатывать то, что было им введено

Для считывания данных с клавиатуры в R предназначена функция `readline()`. Разберем использование этой функции. Создайте новый скрипт, наберите код

```
name<- readline('Введите свое имя')
```

и запустите программу. Чтобы ввести имя, нужно перейти из окна скрипта в окно консоли и там ввести имя, нажать Enter. Программа отработала, в переменной `name` хранится введенное имя. Как его вывести? Наберите в консоли `name` и нажмите Enter.

Если мы хотим после ввода имени сразу его показать, то очевидным решением было бы написать так:

```
name<- readline('Введите свое имя ')  
name
```

Запустите программу. Что мы видим? Функция `readline()` вместо того, чтобы ждать ввода имени с клавиатуры, записала в переменную `name` тот код, который идет следом за вызовом этой функции. Причина такого поведения заключается в том, что выполнение программ на языке R происходит построчно, в режиме интерпретатора.

Что происходит при выполнении программы? Первая строка выполнена средой R, и функция `readline()` ожидает ввода. Но следом среда R выполняет следующую строчку, и именно ее и ловит `readline()` и помещает в переменную `name`.

Как заставить R работать «разумно»? Решение есть, и оно заключается в использовании фигурных скобок, говорящих среде R, что код в скобках нужно рассматривать как одну сложную команду. Добавьте в скрипт открывающую и закрывающую фигурные скобки:

```
{
  name<- readline('Введите свое имя ')
  name
}
```

Повторите выполнение программы. Теперь все работает правильно.
Сделаем вывод более изящным:

```
{
  name<- readline('Введите свое имя ')
  print(paste("Здравствуй,", name, "!"))
}
```

Теперь после ввода имени программа здоровается с Пользователем, обращаясь к нему по имени. Что добавлено в вызов функции `print()`? Функция `print()` выводит строку, переданную как параметр в круглых скобках. Мы конструируем строку вывода, объединяя три строки в одну с помощью функции `paste()`. В функцию `paste()` передается три аргумента, разделенных запятыми:

- Первый аргумент – строка “Здравствуй,”
- Второй аргумент – имя переменной
- Третий аргумент – строка с восклицательным знаком, состоящая из одного символа.

В результате функция `paste()` собирает одну строчку, добавляя пробелы между аргументами. Если мы не хотим добавлять пробелы, то используем похожую функцию – `paste0()`, добавив нужный пробел самостоятельно:

```
{
  name<- readline('Введите свое имя ')
  print(paste0("Здравствуй, ", name, "!"))
}
```

Чуть усложним задачу. Пусть в программе требуется регистрация Пользователя, для этого ему нужно ввести имя и фамилию. Код следующий:

```
{  
  name<- readline('Введите свое имя ')  
  surname<- readline('Введите свою фамилию ')  
  print(paste0("Здравствуйте, ", name, " ", surname, "!"))  
}
```

Последнюю сложную строчку можно разбить на две строки кода, в первой формировать строку вывода, во второй – печатать ее:

```
str4out <- paste0("Здравствуйте, ", name, " ", surname, "!")  
print(str4out)
```

Внесите соответствующие изменения в свой скрипт и выполните его.

Что еще можно улучшить? Обратите внимание, что итоговая строка выводится в кавычках, что не всегда нужно. Можно вывести строку без кавычек, для этого в функцию `print()` необходимо передать еще один параметр – логический переключатель, который определяет, выводить кавычки, или нет. Самостоятельно разберитесь, как вывести строку без кавычек.

Во многих задачах от Пользователя требуется ввести некоторые числовые значения. Пусть в программе требуется получить число, а затем умножить его на 10. Вот код, который, казалось бы, должен решить эту задачу (см. рис. 1).

Создайте новый скрипт, наберите этот код и запустите на выполнение. Правильно ли работает эта программа? Нет, мы получили сообщение об ошибке. Если мы зададим значение `n`, равное 3 и затем выведем в консоли значение переменной `n`, то увидим следующий результат: "3".

```
{  
  n <- readline('Введите число: ')  
  print(n*10)  
}
```

Рис. 1. Попытка считать число с клавиатуры

Кавычки означают, что в переменной `n` сидит не число, а строка. Почему в переменной `n` строка, понять не сложно: все, что мы вводим с клавиатуры, это последовательность символов, и функция `readline()` просто собирает наши нажатия и складывает их в строку. Поэтому после получения ввода с клавиатуры нужно выполнить еще одно действие: преобразовать строку в число с помощью функции `as.integer()` (рис. 2):

```
{  
  n <- readline('Введите число: ')  
  n <- as.integer(n)  
  print(n*10)  
}  
  
# или сразу после ввода преобразуем ввод в целое:  
  
{  
  n <- as.integer(readline('Введите число: '))  
  print(n*10)  
}  
  
# или сразу считывание и преобразование к целому внесем в print():  
  
print(as.integer(readline('Введите число: '))*10)
```

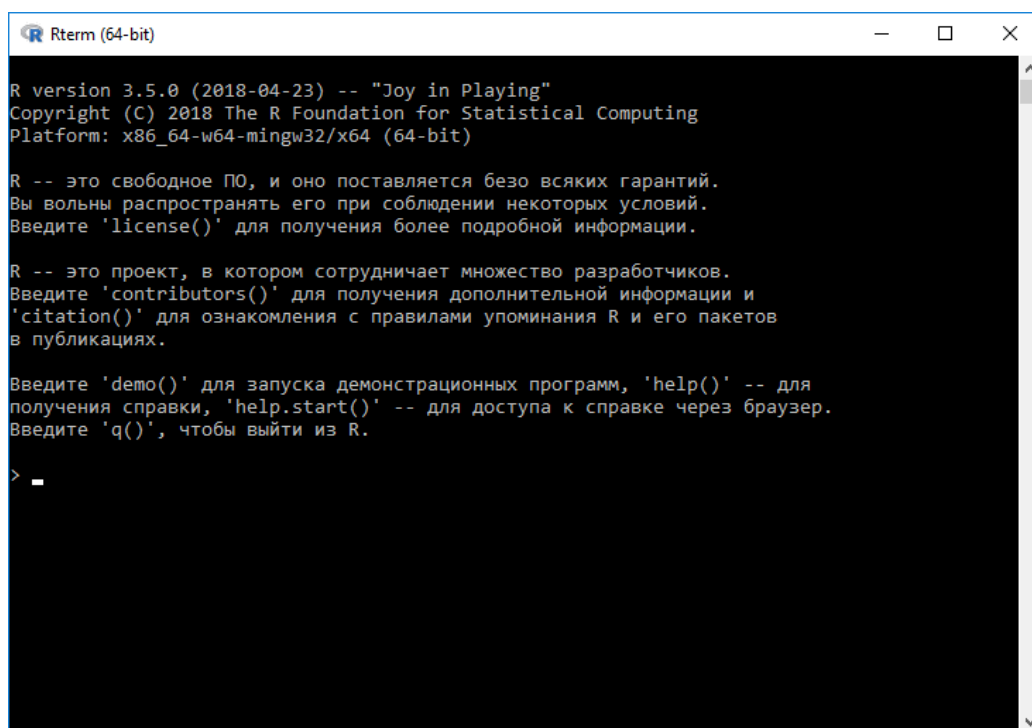
Рис. 2. Варианты получения целого числа

На рисунке 2 представлено три варианта решения задачи. Последний вариант самый короткий, но малопонятный. Лучше пользоваться вторым вариантом, сразу результат функции `readline()` преобразовывать в число.

Контрольные вопросы и задания

Решения заданий 1 – 4 подготовить в файле формата Word с именем Группа_Фамилия_Лаб1. Код выполнения остальных заданий сохранить в скрипте Группа_Фамилия_Лаб1.R

1. Среду R можно запустить и в классическом виде консольной программы (рис. К1). Найти способ, как это сделать. Представить скриншот рабочего стола с изображением, аналогичным на рис.



```
Rterm (64-bit)
R version 3.5.0 (2018-04-23) -- "Joy in Playing"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R -- это свободное ПО, и оно поставляется безо всяких гарантий.
Вы вольны распространять его при соблюдении некоторых условий.
Введите 'license()' для получения более подробной информации.

R -- это проект, в котором сотрудничает множество разработчиков.
Введите 'contributors()' для получения дополнительной информации и
'citation()' для ознакомления с правилами упоминания R и его пакетов
в публикациях.

Введите 'demo()' для запуска демонстрационных программ, 'help()' -- для
получения справки, 'help.start()' -- для доступа к справке через браузер.
Введите 'q()', чтобы выйти из R.

> _
```

Рис. К1. Запуск R в виде классической консольной программы

2. Пройти регистрацию и создать учетные записи на сайтах `stackoverflow.com` и `ru.stackoverflow.com`. Представить скриншот учетной записи.
3. Найти 5 вопросов по языку R на сайтах `stackoverflow.com` или `ru.stackoverflow.com` и понять, о чем в них идет речь и какие решения предлагают опытные программисты. Для одного вопроса, самого интересного для вас, представить скриншот.
4. Найти, запустить и исследовать демонстрационные версии программ, которые автоматически были установлены со средой R. Обращать внимание на представленные варианты графиков и диаграмм. Представить скриншоты двух самых интересных для вас графиков.
5. Проверить работу арифметических операторов и приоритет выполнения операций в сложных выражениях. Для этого, используя Таблицу 1, выполнить серию из 8-10 вычислений по различным самостоятельно придуманным формулам. Использовать все операторы из таблицы. Выяснить и записать правило с учетом приоритета операций, по которому выполняется расчет, если в формуле используется и целочисленное деление, и остаток от деления.
6. В языке R есть несколько специфических значений, которые могут быть получены после выполнения некоторых операций или в результате вычислений: `NULL`, `Inf`, `NaN`, `NA`. Выяснить смысл этих обозначений и получить их в результате некоторых действий с переменными в среде R.

7. Написать программу, в которых требуется

- получить вводом с клавиатуры два целых числа
- первое число возвести в степень второго, вывести эти числа и результат
- затем второе число возвести в степень первого, вывести результат
- разделить первое число на второе, вывести результат
- выполнить деление на ноль, посмотреть на результат

Указание. Для ввода значения с клавиатуры использовать функцию `readline()`. Для преобразования строки в число использовать функцию `as.integer()`. Для формирования строки с результатами и поясняющим текстом использовать функции `paste()`, `paste0()`. Для вывода результатов использовать функцию `print()`.