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 № 2

Кодировки и типы данных R

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

Москва 2021

Цель лабораторной работы заключается в изучении правил корректного вывода информации с использованием национальных алфавитов, а также в знакомстве с правилами преобразования типов данных языка R.

1. Кодировки и корректный вывод сообщений на русском языке

Нередко встречаются ситуации, когда программа, написанная на одном компьютере, а затем запущенная на другом, вместо сообщений на русском языке выводит непонятные символы. Проблема заключается в том, что на этих компьютерах используются разные кодовые страницы для отображения русских букв.

Кодировка символов (другое название – кодовые страницы) – это наборы чисел, которые ставятся в соответствие группе алфавитно-цифровых символов, знакам пунктуации и специальным символам, расположенным на клавиатуре. При использовании одной кодовой страницы на экране виден один результат – понятный текст, при использовании другой страницы вместо текста на экране нечитаемые символы.

Символы английского алфавита являются стандартными и кодируются единым способом во всех кодировках, поэтому проблем с отображением англоязычного текста нет. Проблемы начинаются, когда в программе требуется вывести текст на каком-либо другом языке. Отображение текста на компьютере конечного пользователя программы зависит от двух кодировок: от кодировки, которая использовалась на компьютере, где создавалась программа, и от кодировки на компьютере конечного пользователя.

Если кодировки на компьютерах А и Б использовались одинаковые, мы увидим один и тот же текст. К настоящему времени способов кодировать

символы национальных алфавитов придумано много, поэтому в общем случае кодировки на разных компьютерах различны.

Еще одна сложность может возникнуть при выполнении файлов R в папках, в именах которых есть пробелы или русские буквы. В некоторых случаях среда R может работать не стабильно, поэтому по возможности не используйте пробелы и названия на русском языке для папок с файлами R.

Как правильно настроить вывод сообщений в программе на русском языке? Русский текст должен набираться при установленной кодовой странице с именем UTF-8 или Windows-1251. Это универсальные кодировки для кириллицы, и при наборе текста в этих кодировках текст вашей программы на другом компьютере с такими же кодировками будет отображаться корректно.

Если для написания программы R используется сторонний редактор или среда разработки, нужно убедиться, что при сохранении файла будет выбрана правильная кодировка. Так, например, для популярного редактора Notepad++ выбор кодировки, в которой будет сохраняться текст, выполняется через специальное меню (см. рис. 1).

Особенно актуален вопрос с кодировками, если программы разрабатываются на машинах с операционными системами, в которых языком по умолчанию является английский. В этом случае нужно самостоятельно разобраться, как установить в операционной системе нужную кодировку или как принудительно сохранять текст в нужной кодировке при работе в различных редакторах или средах разработки.

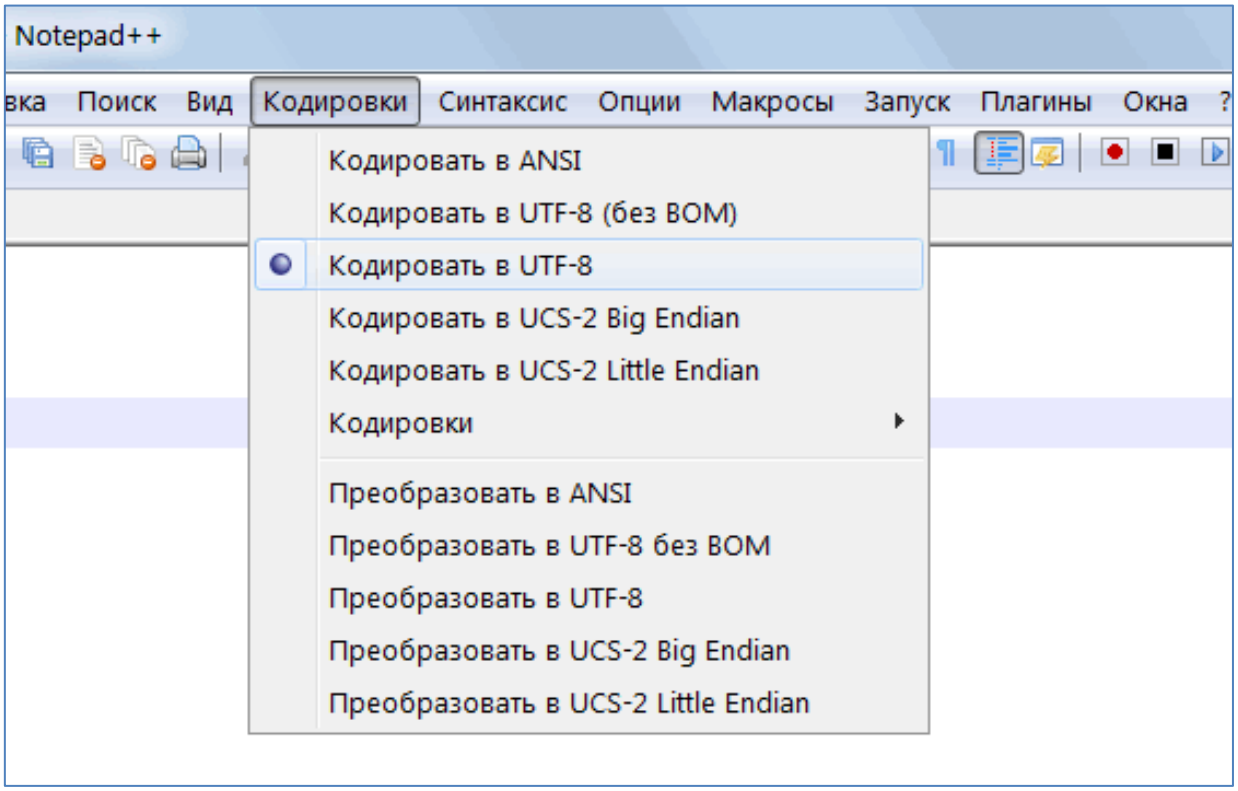

Рис. 1. Выбор кодировки в программе Notepad++

Общая рекомендация: сохранять файлы с кодом, используя кодировку UTF-8, а не Windows-1251. Причины заключаются в следующем:

- Windows-1251 – набор символов и кодировка, являющаяся стандартной 8-битной кодировкой для всех русскоязычных версий Microsoft Windows
- Кодировка UTF-8 универсальная, не зависит от ОС
- Для кодирования одного символа в Windows-1251 используется 1 байт, поэтому количество возможных символов ограничено 255 (2^8-1)
- В UTF-8 один символ может кодироваться 6 байтами, в настоящее время используется 4. Таким образом, количество используемых в кодировке символов превышает 100000

- UTF-8 позволяет работать одновременно с несколькими языками, т.е. поддерживает тексты, в которых используются символы разных алфавитов и даже иероглифы. С использованием кодировки 1251 это невозможно
- Почти все современные веб-платформы и web-сервисы по умолчанию отображают текст через UTF-8
- UTF-8 используется для создания мультиязычных проектов

Для закрепления материала выполнить Задание 1.

2. Динамическая и статическая типизация

Язык R является динамически типизированным языком (как и Python, Objective-C, Ruby, PHP, Perl, JavaScript и др.), в отличие от языков со статической типизацией (C++, C#, Java, Pascal и др).

Идея динамической типизации заключается в том, что пока переменной не присвоено какое-либо значение, тип этой переменной не определен. Только получив значение, например строковое, переменная приобретает и тип. В случае строкового значения тип переменной будет определен как символьный. В другом месте программы этой же переменной может быть присвоено числовое значение, соответственно тип переменной изменится на числовой.

Динамическая типизация дает программисту дополнительную свободу в написании программ, так как одну и ту же переменную можно многократно использовать в разных местах программы в разных контекстах. В идее динамической типизации есть и минус: о возможных ошибках в выражениях с переменными мы узнаем только при выполнении программы.

В отличие от динамической типизации статическая типизация требует определения типа переменной сразу при ее объявлении, и в дальнейшем тип переменной не может быть изменен. Такой подход повышает надежность программ и позволяет выявить ряд ошибок сразу, до запуска программы, на

этапе автоматических проверок, перед преобразованием рукописных команд в машинные коды.

Одним из недостатков статической типизации считается сложность работы с базами данных и внешними структурами данных, так как в одних и тех же полях могут храниться данные разных типов, и при их считывании в переменные определенного, фиксированного типа будут возникать ошибки.

3. Типы переменных и констант

Тип переменной или константы определяется типом значения, присвоенного этой переменной или константе. В R выделяют три числовых типа:

- целые (**integer**)
- действительные (**double**)
- комплексные (**complex**)

Все числовые типы относятся к более общему типу **numeric**, т.е. переменные типа **integer**, **double** и **complex** одновременно являются переменными типа **numeric**.

Кроме числовых типов, переменная или константа может быть логической (**logical**) или символьной (**character**). Логическая переменная может принимать только два значения TRUE (истина) или FALSE (ложь). Используют и сокращенные обозначения: T или F.

Как уже отмечалось, TRUE и FALSE – зарезервированные слова в языке R, поэтому использование их в качестве имен переменных приведет к ошибке, но использование T и F в качестве имен переменных к ошибке не приводит.

Символьный тип предназначен для хранения одиночных символов и их последовательностей (строк).

Числовые константы

Числовые константы, за которыми следует L, считаются целыми, а те, за которыми следует i, считаются комплексными. Если после числа никакого символа нет, число считается типа **double**, это значение по умолчанию.

Если перед числом стоит 0x или 0X, число считается шестнадцатеричным.

Символьные константы

Символьные константы (**character**) задаются при помощи одинарных (') или двойных кавычек (") .

Примеры:

'Просто строчка', "17L * 0.1”, " ", 'abc cba'

Логические константы

Логические константы (**logical**) задаются присвоением им значений TRUE или FALSE.

Встроенные константы

Ниже приведены некоторые из встроенных констант языка R.

Число пи. Наберите pi и нажмете Enter, должно получиться следующее:

> pi

3,141593

Названия месяцев на английском языке хранятся в специальном массиве, наберите в консоли название этого массива:

month.name

Что должны увидеть:

"January" "February" "March" "April" "May" "June"

"July" "August" "September" "October" "November" "December"

Сокращения названий месяцев:

month.abb

Результат:

"Jan" "Feb" "Mar" "Apr" "May" "Jun" "Jul" "Aug" "Sep" "Oct" "Nov" "Dec"

Переопределение констант

Константы легко переопределить. Так, присвоим pi другое значение, например, 99 и заодно вспомним, какой оператор присваивания используется в R. Чтобы объявить переменную или константу и задать ей какое-либо значение, используют конструкцию, состоящую из знаков «меньше» и «тире»: <- (сочетание Alt- в RStudio).

Наберите в консоли следующие две строчки и выполните их:

```
pi <- 99
pi
[1] 99
```

Таким образом, с точки зрения языка, константы ничем не отличаются от переменных, всю логику создания и использования констант и переменных определяет сам программист.

Для того, чтобы отличать константы от переменных, программисты часто используют дополнительные соглашения об именовании констант: в именах констант принято использовать только заглавные буквы и символ подчеркивания, например

```
MIN_LEVEL
MAX_ATTEMPT_AMOUNT
TAX_VALUE
FLAG_OK
```

Значения констант определяются в программе один раз, как правило, в самом начале, в специальном разделе инициализации констант и переменных программы.

4. Проверка и явное преобразование типов

Узнать, к какому типу принадлежит переменная или константа, можно с помощью следующих функций языка:

typeof(имя), где имя – имя переменной или константы

is.numeric(имя)

is.integer(имя)

is.double(имя)

is.numeric(имя)

is.logical(имя)

is.character(имя)

is.finite(имя)

is.infinite(имя)

is.na(имя)

is.nan(имя)

Четыре последние функции предназначены для определения того, является ли значение переменной соответственно конечным значением, бесконечностью, отсутствующим значением, неопределенным результатом. Эти функции, как правило, используются при обработке результатов экспериментов и при анализе данных.

Можно явно изменить тип переменной, в результате чего значение, которое она содержит, будет преобразовано к новому типу. Функции для преобразования типов следующие:

as.numeric(имя)

as.integer(имя)

as.double(имя)

as.numeric(имя)

as.logical(имя)

as.character(имя)

Проверьте, что будет, если преобразовывать к числу строчку, содержащую нечисловую последовательность.

7. Условные операторы и неявное преобразование типов

Условные операторы нужны для проверки истинности различных условий, они представлены в Таблице 1. Создайте несколько числовых, логических и символьных переменных и проверьте работу этих операторов. Обратите внимание, что сравнивать между собой можно не только числовые переменные, но и переменные других типов. Результат сравнения всегда принимает значение TRUE или FALSE.

При написании программ возможны ситуации, когда в выражениях сравнения участвуют переменные разных типов. В этом случае перед выполнением сравнения автоматически происходит неявное преобразование типов. Делается это для того, чтобы операция сравнения была корректной, а это возможно только с переменными одного типа.

Таблица 1. Условные операторы

<	Меньше
>	Больше
<=	Меньше или равно
>=	Больше или равно
==	Равно
!=	Не равно

Рассмотрим пример. Наберите на компьютере следующие команды, после каждой команды нажимайте Enter:

```
a <- 29
b <- '29'
c <- a == b
```

Чему равно значение переменной c? Наберите c и нажмите Enter, убедитесь, что значение c равно TRUE. Из этого результата следует, что

1) или числовое значение переменной a было преобразовано к строке ‘29’
2) или строковая переменная b была преобразована в число 29.

Выясним, какой вариант верный. Рассмотрим второй пример. Наберите:

```
a <- 28
b <- '30 - 2'
c <- a<b
```

Чему равно значение c? Какой вывод отсюда можно сделать? Проверьте свои заключения, выполнив третий пример:

```
a <- 28
b <- '20 + 10'
c <- a<b
```

Чему равно c?

8. Логические операторы

С помощью логических операторов можно тестировать на истинность не одно выражение, а сразу несколько. Для связывания нескольких условий в одно логическое выражение используются логические операторы (Таблица 2).

Таблица 2. Логические операторы

!	Логическое НЕ (NOT)
&	Поэлементное логическое И (AND)
&&	Сокращенное логическое И (AND)
|	Поэлементное логическое ИЛИ (OR)
||	Сокращенное логическое ИЛИ (OR)

Например, требуется, чтобы определенная часть программы была выполнена, только если совпадут все требуемые условия – в этом случае необходимо использовать логическое И; или при анализе значений нескольких переменных требуется, чтобы хотя бы одно из значений находилось в нужном диапазоне – в логическом выражении нужно использовать логическое ИЛИ.

Как видно из таблицы, для логического И и логического ИЛИ существует по два оператора. Важно понимать разницу в их работе. Отличие заключается в способе обработки операндов (сравниваемых переменных). Если сравниваются скалярные значения, то разницы в работе операторов нет, выполните два примера

Пример 1. Используем поэлементное ИЛИ:

```
q1 <-2
q2 <-0
q3 <- (q1 == 2) | (q2 ==2)
```

Чему равно q3?

Пример 2. Используем сокращенное ИЛИ:

```
q1 <-2
q2 <-0
q3 <- (q1 == 2) || (q2 ==2)
```

Чему равно q3?

В R одной операцией можно сравнивать не только скалярные значения. Результаты изменятся, если вместо скалярных переменных сравниваются массивы (другое название – **векторы**). Рассмотрим примеры 3 и 4, где переменны q1 и q2 – вектора (последовательности чисел).

Пример 3. Используем поэлементное ИЛИ:

```
q1 <-(2:4)       # вектор из элементов 2, 3, 4
q2 <-(0:2)       # вектор из элементов 0, 1, 2
q3 <- (q1 == 2) | (q2 ==2)
```

Чему равно q3?

Пример 4. Используем сокращенное ИЛИ:

q1 <-(2:4)

q2 <-(0:2)

q3 <- (q1 == 2) || (q2 ==2)

Чему равно q3?

Как видно из результатов выполнения примеров 3 и 4, значения q3 различны. Именно в этом смысл использования как сокращенного, так и поэлементного И.

Контрольные вопросы и задания

Все задания выполнять в одном скрипте с форматом имени Группа_Фамилия_Лаб2.

1. Написать скрипт, в котором создать 5-10 переменных разных типов, проверить их тип с помощью функции typeof(). Проверить с помощью этой функции типы следующих констант: 29, 23i, -34L, 2/3, 4/2, 0xA, 0XbL – 120L, 0XbL – 120, 0XbL * 17. Объяснить полученные результаты в комментариях после выполнения проверок.

2. Исследовать правила неявного преобразования типов, для чего написать программу, которая получает от Пользователя некоторое значение (не обязательно числовое). Затем это значение преобразовать в переменные разного типа, и эти переменные сравнить попарно между собой (см. Табл. 1):
   integer и double
   integer и logical
   logical и character
   double и logical

double и character

Для вывода сообщений использовать функцию print(). Напечатать сообщения в кавычках и без кавычек.

По результатам работы программы сформулировать общее правило неявного преобразования типов в R. Правило записать в комментариях после текста программы.

3. Исследовать результат применения арифметических операций (*, /, +, -, ^, %%, %/%) в выражениях с переменными разного типа. Для этого написать программу, которая получает от Пользователя два значения (не обязательно числовые). Затем эти значения преобразовать в переменные разных типов, и с этими переменными выполнить арифметические операции в различных вариантах:

   integer и double

   integer и logical

   logical и character

   double и logical

   double и character

   Для вывода сообщения использовать функцию cat(). Сформулировать отличия в работе функций print() и cat(). Отличия в работе функций записать в комментариях к программе.

   По результатам работы программы сформулировать общее правило применимости арифметических операций к переменным разного типа. Правило записать в комментариях после текста программы.

4. Написать код, в котором по аналогии с примерами 1-4 проверить работу логических операторов & и && со скалярными значениями и с векторами. Рассмотреть сравнение скалярного значения с

вектором. Ввод с клавиатуры не делать, сразу присваивать значения переменным, как в примерах 1-4. Объяснить полученные результаты, написав комментарий к каждому рассмотренному случаю.

5. Последовательно выполнить операции:
   3/7
   3/7 – 0.4285714
   Объяснить полученный результат в комментарии.

6. Последовательно выполнить операции:
   sqrt(2)*sqrt(2)
   (sqrt(2)*sqrt(2))-2
   Объяснить полученный результат в комментарии.