Рабочая группа
разработки R

Определение языка R

Версия 3.5.2 (2018-12-20)
DRAFT

Александр Александрович Фоменко

Определение языка R. Версия 3.5.2 (2018-12-20) DRAFT

http://www.litres.ru/pages/biblio_book/?art=42224238
ISBN 9785449660299

Аннотация

Данная книга является переводом одноименной книги из комплекта технической документации, поставляемой в составе дистрибутива R, и призвана восполнить пробел в русской локализации системы R.

Содержание

Определение языка R
Версия 3.5.2 (2018-12-20)
DRAFT

Версия 3.5.2 (2018-12-20) DRAFT
Рабочая группа разработки R

Переводчик Александр Александрович Фоменко

ISBN 978-5-4496-6029-9
Создано в интеллектуальной издательской системе Ridero

R Language Definition

Version 3.5.2 (2018-12-20) DRAFT
R Development Core Team

По вопросам перевода обращаться по адресу:
http://www.aafomenko@yandex.ru

1. Введение

R – система для статистического вычисления и графики. Она включает, между прочим, язык программирования, высокоуровневую графику, интерфейсы к другим языкам и средства отладки. Этот справочник детализирует и определяет язык ***R.***

Язык ***R*** – диалект ***S,*** который был разработан в 1980-ых и с тех пор находится в широком использовании в статистическом сообществе. Его основной разработчик, Джон М. Чемберс, был награжден Премией по системному программному обеспечению ACM 1998 года за S.

У синтаксиса языка есть поверхностное подобие с ***C,*** но семантика принадлежит к семейству FPL (языкам функционального программирования) с более сильной аффилированностью с Lisp и APL. В частности, возможно «вычислять на языке», который поочередно позволяет записать функции, которые берут выражения в качестве входа, что-то, что часто полезно для статистического моделирования и графики.

Можно получить довольно отдаленное использование ***R*** в интерактивном режиме, выполняя простые выражения из командной строки. Некоторые пользователи никогда, возможно, не уйдут с этого уровня, другие захотят написать свои собственные функции или оперативно систематизиро-

вать однообразную работу или на перспективу записать дополнительные пакеты для новой функциональности.

Цель этого справочника состоит в документировании языка по существу. Это означает, объекты показаны, так как они работают, и детали процесса вычисления выражений, которые полезно знать при программировании функций ***R.*** Главные подсистемы для определенных задач, таких как графика, описаны только кратко в этом справочнике и будут задокументированы отдельно.

Хотя большая часть текста одинаково применима к ***S,*** есть также некоторые существенные различия, и чтобы не перепутать проблему, сконцентрируемся на описании ***R.***

Проект языка содержит много тонкостей и распространенных ошибок, которые могут удивить пользователя. Большинство из них происходит из-за соображений непротиворечивости на более глубоком уровне, как мы объясним. Есть также много полезных ярлыков и идиом, которые позволяют пользователю выражать вполне сложные операции кратко. Многие из них становятся естественными по мере ознакомления с базовыми понятиями. В некоторых случаях существует много способов выполнить задачу, но некоторые из методов основаны на реализации языка, а другие работают на более высоком уровне абстракции. В таких случаях укажем на преимущественное использование.

Предполагается некоторое знакомство с ***R.*** Данное руководство не введение в ***R,*** а скорее справочник програм-

миста. Другие справочники предоставляют дополнительную информацию: в особенности раздел «Предисловие» во Введении в ***R*** предоставляет введение в ***R,*** и раздел «Система и интерфейсы внешних языков» в Написание расширений ***R*** детализирует расширение ***R,*** используя скомпилированный код.

2. Объекты

На каждом машинном языке переменные обеспечивают средство доступа к данным, хранящимся в памяти. ***R*** не обеспечивает прямой доступ к памяти компьютера, а скорее обеспечивает много специализированных структур данных, именуемых как объекты. Эти объекты упомянуты через символы или переменные. В ***R,*** однако, символы – самостоятельно объекты и могут управляться таким же образом как любой другой объект. Этим он отличается от многих других языков и имеет широко распространяющиеся следствия.

В этой главе даны предварительные описания различных структур данных, предоставленных в ***R.*** Более детальные обсуждения многих из них будут найдены в последующих главах. Функция определения *typeof* в ***R*** возвращает *mun* объекта ***R.*** Заметим, что в коде ***C,*** лежащем в основе ***R,*** все объекты являются указателями на структуру с определением типа SEXPREC; различные типы данных ***R*** представлены в ***C*** SEXPTYPE, который определяет, как используется информация в различных частях структуры.

Следующая таблица описывает возможное значение, возвращенное *typeof*, и их значение.

«NULL»

NULL

«symbol»

имя переменной

«pairlist»

парный объект (в основном внутренний)

«closure»

функция

«environment»

окружающая среда

«promise»

объект, используемый для отложенной оценки

«language»

*конструкция языка **R***

«special»

внутренняя функция, которая не вычисляет свои аргументы

«builtin»

внутренняя функция, которая вычисляет свои аргументы

«char» a «scalar»

*строковый объект (только внутренний) ****

«logical»

вектор, содержащий логические значения

«integer»

вектор, содержащий целые значения

«double»

вектор, содержащий реальные значения

«complex»

вектор, содержащий комплексные значения

«character»

вектор, содержащий символьные значения

«…»

аргумент определенной переменной длины ***

«any»

специальный тип, который заменяет все типы: не существует объектов такого типа

«expression»

объект выражение

«list»

список

«bytecode»

код в байтах (только внутренне) ***

«externalptr»

объект внешнего указателя

«weakref»

объект слабой ссылки

«raw»

вектор, содержащий байты

«S4»

объект S4, который не является простым объектом

Пользователи не могут просто получить объекты, помеченные «***».

Функциональный режим дает информацию о режиме объекта в смысле Becker, Chambers & Wilks (1988), и является более совместимым с другими реализациями языка ***S***. Наконец, функция *storage.mode* показывает режим хранения ее аргумента в смысле Беккера и др. (1988). Она обычно используется при вызове функции, записанной на другом языке, таких как ***C*** или ФОРТРАН для гарантирования, что объекты ***R*** имеют тип данных, который ожидает вызываемая подпрограмма. (На языке ***S*** векторы с целочисленными или действительными значениями имеют оба «числовой» режим, таким образом, их режимы хранения нужно отличать.)

```
> x <– 1:3
> typeof (x)
[1] «integer»
> mode (x)
[1] «numeric»
> storage.mode (x) [1] «integer»
```

Объекты в ***R*** часто преобразовываются к различным типам во время вычислений. Также имеется много доступных функций для выполнения явного преобразования. При программировании на языке ***R*** тип объекта обычно не влияет на вычисления, однако, имея дело с внешними языками или операционной системой, часто необходимо гарантиро-

вать корректность типа объекта.

2.1. Основные типы

2.1.1. Векторы

Вектора рассматриваются как непрерывная последовательность ячеек, содержащих данные. Доступ к ячейкам осуществляется через операции индексирования, такими, как x [5]. Более детально рассмотрено в разделе 3.4 [индексирование].

R имеет шесть основных («атомарных») типов векторов: *logical, integer, real, complex, string (or character)* и *raw*. Режим и режим хранения для разных типов векторов перечислены в следующей таблице.

Тип	mode	storage.mode
logical	*logical*	*logical*
integer	*numeric*	*integer*
double	*numeric*	*double*
complex	*complex*	*complex*
character	*character*	*character*
raw	*raw*	*raw*

Отдельные числа, такие как 4.2, и строка, такая как «*four point two*», все еще векторы, длины 1; нет больше основных типов. Возможны (и полезны) векторы с нулевой длиной.

У векторов строки есть режим и режим хранения

«*character*». Отдельный элемент символьного вектора часто упоминается как *символьная строка*.

2.1.2. Списки

Списки («универсальные векторы») являются другим видом хранения данных. У списков есть элементы, каждый из которых может содержать любой тип объекта ***R,*** то есть элементы списка не обязательно имеют одинаковый тип. К элементам списка получают доступ посредством трех различных операций индексации. Они объяснены подробно в Разделе 3.4 [Индексирование].

Списки – векторы, и основные типы векторов упоминаются как атомарные векторы, где необходимо исключить списки.

2.1.3. Языковые объекты

Есть три типа объектов, которые составляют язык ***R,*** а именно: *call* (вызов), *expressions* (выражения) и *name* (имя). Так как у ***R*** есть объекты типа «выражение», то попытаемся избежать использования слова «выражение» в других контекстах. В определенных синтаксически корректных высказываниях выражения будут упоминаться как *операторы*.

У этих объектов есть режимы «*call», «expression»* и «*name*», соответственно.

Они могут быть созданы непосредственно из выражений, используя механизм кавычек и преобразованы «в» и «из» списков функциями *as.call* и *as.list*. Могут быть извлечены компоненты дерева синтаксического анализа, используя

стандартные операции индексации.

2.1.4. Символьные объекты

Символы обращаются к объектам ***R.*** Обычно имя любого объекта ***R*** – символ. Символы могут быть созданы через функции *as.name* и кавычку.

Символы имеют режим «*name*», режим хранения «*symbol*» и тип «*symbol*». Они могут быть преобразованы «в» и «из» символьных строк, используя *as.character* и *as.name*. Они естественно появляются как атомы проанализированных выражений, попробуй, например, *as.list (quote (x + y)).*

2.1.5. Выражения – объекты

В ***R*** можно иметь объекты типа» *expression*». Выражение содержит одно или более предложений. Оператор – синтаксически корректный набор маркеров. Объекты выражения – специальные объекты языка, которые содержат проанализированные, но неоцененные операторы ***R.*** Основное различие состоит в том, что объект выражения может содержать несколько таких выражений. Другие более тонкие различия состоят в том, что объекты типа «*expression*» оцениваются лишь при явной передаче на вычисление, тогда как другие объекты языка могут быть оценены в некоторых неожиданных случаях.

Объект выражения ведет себя также как список, и к его компонентам можно получить доступ таким же образом как компонентам списка.

2.1.6 Объекты функции

В ***R*** функции – объекты и могут управляться почти таким же способом как любой другой объект. У функций (или более точно, обертка функции) есть три основных компонента: формальный список аргументов, тело и окружающая среда. Список аргументов – список разделенных запятой значений аргументов. Аргумент может быть символом, или конструкцией «*symbol* = *default*», или специальным аргументом «…'. Вторая форма аргумента используется для указания значения по умолчанию для аргумента, которое будет использоваться при вызове функция без какого-либо значения, указанного для этого аргумента. Аргумент «…» является особенным и может содержать любое число аргументов. Он обычно используется, если число аргументов неизвестно или в случаях, где аргументы будут переданы другой функции.

Тело – синтаксически проанализированный оператор ***R,*** обычно набор операторов в фигурных скобках, но также может быть отдельный оператор, символ или даже константа.

Окружающая среда функции является средой, которая была активной при создании функции. Любой символ ограничен своей окружающей средой, связан и доступен функции. Комбинацию кода функции и привязки в ее окружающей среде называют «оберткой функции», термином из теории функционального программирования. Здесь обычно используется термин «функция», но используется «обертка», чтобы подчеркнуть значимость присоединенной среды.

Можно извлечь и управлять тремя частями обертки объекта, используя конструкции *formals, body* и *environment* (все три могут также использоваться на левой стороне присваивания). Последний из них может использоваться для удаления нежелательной привязки среды.

При вызове функции создается новая среда (называемая средой оценки), чье пространство (см. раздел 2.1.10 [Окружающая среда]) является средой от обертки функции. Новая среда первоначально заполнена неоцененными аргументами функции; поскольку оценка продолжается, локальные переменные создаются в ее пределах.

Есть также средство для преобразования функции «в» и «из» списочной структуры, используя *as.list* и *as. function*. Они были включены для совместимости с ***S*** и их использование обескураживает.

2.1.7. NULL

Существует специальный объект, называемый *NULL.* Он используется всякий раз, когда есть потребность идентифицировать или указать отсутствие объекта. Его не следует путать с вектором или списком нулевой длины.

Объект *NULL* не имеет типа и каких-либо поддающихся изменению свойств. В ***R*** есть только один объект *NULL*, к которому обращаются все экземпляры. Для проверки на *NULL* используют *is.null*. Нельзя установить атрибуты для *NULL.*

2.1.8. Встроенные объекты и специальные формы

Эти два вида объекта содержат встроенные функции ***R,***

то есть, те, которые выведены на экран как. *Primitive* в листингах кода (так же как те, к которым получают доступ через функцию. *Internal* и, следовательно, не видимые пользователем как объекты). Различия между ними заключается в обработке аргумента. Все собственные аргументы встроенных функций оцениваются и передаются внутренней функции в соответствии с *вызовом по значению,* тогда как специальные функции передают не оцененные аргументы внутренней функции.

Для языка ***R*** эти объекты – только другой вид функции. Функция *is.primitive* может отличить их от интерпретируемых функций.

2.1.9. Обещанные объекты

Объекты обещания – часть механизма отложенных вычислений ***R.*** Они содержат три слота: значение, выражение и окружающая среда. При вызове функции сравниваются аргументы, а затем каждый из формальных аргументов является обязательством к обещанию. Выражение, которое было дано для формального аргумента, и указатель на окружающую среду функции вызываются из сохраненных в обещании.

Конец ознакомительного фрагмента.