Содержание

Осваиваем язык Julia

Julia – это хорошо структурированный язык программирования с большим быстродействием, устраняющий классическую проблему выполнения анализа на одном языке и трансляции его результатов на второй с целью повышения производительности. Эта книга поможет вам развить и усовершенствовать свои навыки программирования на Julia для решения задач автоматизации, возникающих в реальной жизни.

Книга начинается с небольшого инструктажа по поводу инсталляции и выполнения Julia в разных операционных средах. Затем вы сравните самые разные способы работы с языком и подробно изучите его ключевой функционал, разбирая практические примеры, построенные на основе пошагового принципа. Пользуясь простыми статистическими и аналитическими показателями, вы откроете для себя быстродействие языка, его реальную мощь, которая делает его особенно полезным в высокоинтенсивных вычислительных задачах, и отметите, что язык Julia способен сотрудничать с внешними процессами, получая значительное улучшение качества графики и визуализации данных. Наконец, вы займетесь метапрограммированием и узнаете, как оно укрепляет мощь языка и формирует его сетевую и распределенную вычислительную среду.

Для кого эта книга написана

Это практическое руководство предназначено для специалистов в области науки о данных. Книга предполагает наличие некоторых навыков работы с Julia и навыков программирования на скриптовом языке, таком как Python и R, либо на компилируемом языке, таком как C или Java.

Чему вы научитесь, прочитав эту книгу

- Инсталлировать среду разработки и выполнять ее сборку и настройку под вашу операционную среду.
- Создавать проекты в области науки о данных в рамках полного цикла ETL, анализа и визуализации данных.
- Понимать систему типов и принципы множественной диспетчеризации для получения большей отдачи от программирования на Julia.
- Взаимодействовать с файлами и таблицами данных с целью изучения простых статистических и аналитических показателей.
- Отображать графики и визуальные данные с целью проведения на Julia имитационного моделирования.
- Использовать Julia для взаимодействия с базами данных SQL и NoSQL.
- Работать с распределенными системами в веб-среде и в облаке.
- Разрабатывать свои собственные программные пакеты и участвовать в деятельности сообщества программистов на Julia в качестве соавтора.

Об авторе

Малкольм Шеррингтон работает в сфере информационных технологий более 35 лет. Он имеет степени в области математики, химии и инженерно-технических наук и читал лекции в двух различных университетах Великобритании, а также работал в аэрокосмической и медицинской отраслях экономики. В настоящее время руководит собственной компанией в финансовом секторе, с определенными интересами к высокоэффективным вычислениям и приложениям на основе графического процессора и параллельных вычислений.

Будучи деятельным специалистом, Малкольм начал программировать научные задачи на Fortran и C, совершенствуя навыки программирования на Ada и Common LISP, и недавно занялся обработкой данных и аналитикой на Perl, Python и R.

Малкольм Шеррингтон является организатором Лондонской ассоциации программистов на Julia. Кроме того, он является соорганизатором британской meetup-группы по высокопроизводительным вычислениям и финансовым технологиям и лондонской meetup-группы финансовых аналитиков.

Я бы хотел посвятить эту книгу памяти моей покойной жены, Хэйзел Шеррингтон, без поддержки которой я бы не взялся осваивать Julia и которой не суждено было увидеть подтверждения своей проницательности.

Кроме того, хочу выразить особую благодарность Барбаре Доре и Джеймсу Уэймсу за их существенную помощь и материальную поддержку при подготовке этой книги.

О рецензентах

Гурурагав Гопал в настоящее время работает консультантом по управлению рисками в стартапах. Ранее он работал в Paterson Securities в качестве разработчика в области финансовой аналитики и консультанта по трейдингу. Кроме того, занимал должность консультанта по анализу данных и был связан с организацией электронной торговли. Он преподавал студентам и аспирантам в Технологическом университете VIT в Веллуру (Индия), специализируясь на распознавании образов, машинном обучении и больших данных. Как научный сотрудник связан с несколькими исследовательскими организациями, а именно IFMR и NAL. Кроме того, выступил рецензентом книги Learning Data Mining with R («Изучение интеллектуального анализа данных при помощи R»), вышедшей в издательстве Packt Publishing, а также нескольких журналов и конференций.

Он имеет степень бакалавра в области электротехники и электроники, а также степень магистра в области информатики и инженерии. Позже он выполнил свою курсовую работу от Института финансового управления и исследования (IFMR), Индия, по финансовой инженерии и управлению рисками, и с тех пор был занят в финансовой отрасли. Завоевал множество наград и имеет несколько международных публикаций.

Гурурагав Гопал интересуется программированием, обучением и выполняет консалтинговые услуги, а в свободное время слушает музыку.

По вопросам профессиональных консультаций к нему можно обратиться на его странице в соцсети LinkedIn (in.linkedin. com/in/gururaghavg).

Жуо СЛ – свободный разработчик из Китая с десятилетним опытом разработок в Linux, на C, C++, Java и Perl. Любит принимать участие в деятельности сообщества программистов (разумеется, включая сообщество программистов на Julia), в том числе в качестве соватора. Ведет свой личный веб-сайт на http://kdr2.com, где о Жуо можно узнать еще больше.

Дэн Власюк – автор различных пакетов Julia, включая пакеты TimeSeries и Quandl; основатель группы JuliaQuant на веб-сайте Github, связанной с программными пакетами в области квантитативных финансов.

Предисловие

Julia – относительно молодой язык программирования. Первоначальные проектные работы по языку Julia начались в Массачусетском технологическом институте (MIT)[1] в августе 2009 года, и к февралю 2012-го он стал общедоступным. Заслуга по созданию этого языка в основном принадлежит трем разработчикам: Стефану Карпински, Джеффу Безансону и Виралу Шаху. Все трое, наряду с Аланом Эдельманом, до сих пор активно развивают проект Julia в MIT, где в настоящее время проводится ряд курсов по Julia, многие из которых доступны в Интернете.

Первоначально разработчики задумывали Julia как язык для научного программирования с быстродействием, достаточным для восполнения потребности в моделировании на интерактивном языке с последующей неизбежной переработкой кода на компилирующем языке, таком как C или Fortran. В то время главные языки для научного программирования, как, например, MATLAB и Mathematica, были защищены правами интеллектуальной собственности и до сих пор остаются относительно медленными. При этом имеются клоны этих языков в области открытого программного обеспечения, например GNU Octave и Scilab, но они работают еще медленнее. Когда язык Julia был запущен, сообщество программистов увидело в нем замену MATLAB, однако это не совсем верно. Несмотря на то что синтаксис Julia похож на MATLAB настолько, что любой компетентный в MATLAB специалист может легко изучить Julia, он не разрабатывался как клон. Это язык с более развитым функционалом и со многими существенными отличиями, которые будут подробно рассмотрены далее.

Период с 2009 года засвидетельствовал рост двух новых вычислительных дисциплин: больших данных / облачных вычислений и науки о данных. Обработка больших данных на Hadoop традиционно рассматривается как область программирования на Java, поскольку Hadoop выполняется в виртуальной машине Java. Разумеется, существует возможность обрабатывать большие данные при помощи языков программирования помимо тех, которые основаны на Java и используют парадигму jar-файлов с потоковой передачей, и тут Julia может применяться по аналогии с тем, как это делается на C++, C# и Python.

Появление науки о данных возвестило о начале использования языков программирования, которые просты для аналитиков, обладающих некими навыками программирования, но не являющихся специалистами в этой области. Два языка, развитие которых ускорилось, чтобы заполнить этот пробел, – R и Python. Оба они относительно стары и уходят своими корнями в 1990-е годы. Однако популярность обоих демонстрирует стремительный рост, по иронии, примерно с того

---

[1] Массачусетский технологический институт – университет и исследовательский центр, расположенный в Кембридже (штат Массачусетс, США). Также известен как Массачусетский институт технологий и Массачусетский технологический университет.

времени, когда общественности был представлен язык Julia. Тем не менее, даже имея такую признанную и солидную оппозицию, этот язык взволновал научное программистское сообщество и продолжает проводить рейды в этом направлении.

Цель этой книги состоит в том, чтобы охватить все аспекты языка Julia, которые делают его привлекательным для аналитиков данных. Язык развивается быстро. Двоичные дистрибутивы доступны для Linux, Mac OS X и Windows, но они отстают от текущих исходников. Поэтому, чтобы выполнять при помощи Julia серьезную работу, важно понимать, как получать и собирать рабочую систему из исходных текстов. Кроме того, для Julia доступны интерактивные среды разработки (IDE), и в данной книге будут проанализированы IDE Jupyter, Juno и плагин JuliaDT для среды Eclipse.

О чем рассказывает эта книга

Глава 1 «*Среда разработки Julia*» рассказывает о том, как запустить дистрибутив Julia и привести его в состояние готовности к работе. Важно уметь получать самые последние исходные тексты и собирать систему с нуля, а также находить и устанавливать надлежащие пакеты и при необходимости их удалять.

Глава 2 «*Разработка на Julia*» содержит краткий обзор части стандартных синтаксических конструкций языка. Julia – язык новый, но он отнюдь не покажется новым читателям с компетенцией в MATLAB, R или Python, поэтому цель главы состоит в том, чтобы с помощью примеров кратко довести до читателей информацию о Julia и направить их к онлайновым источникам. Кроме того, важно понимать разницу работы через консоль и с интегрированными средами разработки Jupyter/Juno/JuliaDT.

Глава 3 «*Типы и диспетчеризация*» посвящена системе типов Julia и показывает, каким образом она предоставляет разработчику мощные методы посредством ее de facto функциональной системы диспетчеризации.

Глава 4 «*Функциональная совместимость*» касается методов, посредством которых Julia может взаимодействовать с операционной системой и другими языками программирования. Эти методы являются в основном нативными для Julia. Глава заканчивается введением в параллелизм, который будет рассмотрен подробнее в главе 9.

Глава 5 «*Работа с данными*» рассматривает выполняемую аналитиком данных процедуру – от источника данных до результатов анализа. Большинство проектов начинается с данных, которые нужно прочесть, очистить и отобрать. Об этом и идет речь в главе. Далее описываются простые статистические и аналитические показатели.

Глава 6 «*Научное программирование*» фактически освещает главную причину для программирования на Julia. Мощь языка заключена в его быстродействии в сочетании с простотой разработки на языке сценариев, что делает его особенно ценным при решении задач с процессами, вычислительно ограниченными возможностями ЦПУ. В главе рассматриваются различные подходы, используемые при решении математических и естественнонаучных задач.

Глава 7 «*Графика*» описывает тот аспект, в котором Julia часто не выдерживает сравнения с другими альтернативными языками, такими как MATLAB и R. Действительно, более ранние версии языка имели достаточно ограниченную поддержку графики, но теперь дело обстоит иначе. В главе представлено большое разнообразие сложных подходов к визуализации графики на экране и сохранению ее в дисковых файлах.

Глава 8 «*Базы данных*» рассказывает о взаимодействии языка Julia с базами данных. В базе данных могут храниться данные для анализа – либо там требуется сохранить результаты анализа. В этой главе будет проанализировано несколько подходов к хранению данных в хранилищах SQL и NoSQL. Они не встроены в язык, а скорее полностью опираются на сторонние пакеты, поэтому в ближайшем будущем могут быть усовершенствованы.

Глава 9 «*Сетевое взаимодействие*» касается аспектов работы с распределенными источниками данных. В науке о данных большие данные и облачные системы приобретают все большую популярность, и в этой главе рассказывается о сетевом программировании на уровне сокета и взаимодействии через сеть Интернет. Кроме того, она посвящена анализу работы Julia в веб-службах Amazon и с вычислительным сервером Google.

Глава 10 «*Работа с Julia*» предоставляет читателям дополнительную информацию и побуждает продолжить работу, участвуя в совместной разработке на Julia. Можно попробовать свои силы в качестве соавтора, вносящего свой вклад в существующий пакет, либо просто примкнуть к одному из сообществ программистов на Julia.

Что вам потребуется для работы

Разработка на Julia может выполняться в любой из известных вычислительных операционных систем: Linux, OS X и Windows. В целях углубленного исследования языка читатель может захотеть получить последние версии и собрать язык из исходников под Linux. Однако для того, чтобы начать работать с языком в любой из трех операционных платформ, предусмотрена простая и удобная инсталляция языка с использованием двоичного дистрибутива. Кроме того, можно отдельно скачать и установить интегрированную среду разработки (IDE) Juno и плагин для среды разработки Eclipse.

Некоторые примеры в более поздних главах, касающиеся поддержки баз данных, сетевого взаимодействия и облачных служб, потребуют установки дополнительных компонентов и использования дополнительных ресурсов, и вопросы их получения будут обсуждаться в соответствующих частях книги.

Для кого эта книга

Эта книга не является введением в программирование, и потому предполагается, что читатель знаком с понятийным аппаратом по крайней мере одного языка программирования. Для тех, кто знаком с языками сценариев, такими как Python, R

и MATLAB, эта задача не является трудной – как, впрочем, и для тех, кто использует похожие языки: C, Java и C#.

Однако для аналитика данных, возможно с компетенцией в методах аналитики на основе электронных таблиц, таких как Excel, или статистических пакетов, таких как SPSS и Stata, значительная часть текста должна оказаться полезной.

Соглашения

В этой книге используется несколько разных стилей оформления, каждый из которых имеет свое назначение. Ниже приведены примеры.

Фрагменты программного кода в тексте, названия таблиц баз данных, папок и файлов, расширения файлов, пути, фиктивные URL, данные, вводимые пользователем, и дескрипторы Twitter выделяются моноширинным шрифтом: «Папка `test` содержит программный код, который иллюстрирует, как писать тестовые сценарии и применять систему `Base.Test`».

Блок кода выглядит следующим образом:

```
function isAdmin2(_mc::Dict{ASCIIString,UserCreds}, _name::ASCIIString)
  check_admin::Bool = false;
  try
    check_admin = _mc[_name].admin
  catch
    check_admin = false
  finally
    return check_admin
  end
end
```

А ввод или вывод командной строки записывается следующим образом:

```
julia> include("asian.jl")
julia> run_asian()
```

Новые термины и *важные слова* выделены *курсивом*. Элементы интерфейса, системные сообщения и клавиши оформляются полужирным шрифтом: «**404, Страница не найдена**».

Сочетания клавиш, которые следует нажимать одновременно, оформляются с помощью знака + (плюс), например: «нажать сочетание клавиш **Ctrl+P**».

Последовательность команд обозначается знаком стрелки, например: «**File** → **Settings** (Файл → Настройки)».


Предупреждения или важные примечания приводятся в отдельном текстовом блоке.


Подсказки и приемы обозначены таким символом.


Так оформляются дополнения к тексту оригинала книги.

Обратная связь

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге – что понравилось или, может быть, не понравилось. Благодаря обратной связи мы получаем возможность выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв прямо на нашем сайте www.dmkpress.com: зайдите на страницу книги и оставьте комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу dmkpress@gmail.com; при этом укажите название книги в теме письма.

Если вы специалист в конкретной области и заинтересованы в написании новой книги, заполните на нашем сайте форму, размещенную на странице http://dmkpress.com/authors/publish_book/ или напишите в издательство по адресу dmkpress@gmail.com.

Скачивание исходного кода программ

Скачать сопроводительные файлы к книгам издательства «ДМК Пресс» можно на сайте www.dmkpress.com или www.дмк.рф. Зайдите на страницу с описанием интересующей вас книги и просмотрите список доступных материалов к ней.

Ошибки и опечатки

Хотя мы стремимся удовлетворить требования самых взыскательных читателей, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг, мы будем очень благодарны, если вы сообщите нам о ней. Таким образом, вы избавите других читателей от досадных недоразумений и поможете нам улучшить последующие издания книги.

Если вам встретятся какие-либо ошибки в тексте, пожалуйста, сообщите об этом главному редактору по адресу dmkpress@gmail.com. Мы учтем ваши пожелания при выпуске следующих тиражей.

Нарушение авторских прав

Пиратство в Интернете по-прежнему остается насущной проблемой. Издательства «ДМК Пресс» и Packt очень серьезно относятся к вопросам защиты авторских прав и лицензирования. Если вы обнаружите в Интернете незаконно размещенные публикации любой нашей книги, пожалуйста, сообщите нам об этом с указанием ссылки на веб-страницу, чтобы мы могли применить соответствующие санкции.

Ссылки на подозрительные материалы направляйте по адресу dmkpress@gmail.com.

Мы высоко ценим любую помощь по защите наших авторов – это позволяет нам предоставлять читателям качественные материалы.

Вопросы

Если у вас имеются какие-либо вопросы, связанные с этой книгой, вы можете обратиться к нам по электронному адресу dmkpress@gmail.com, и мы приложим все усилия, чтобы помочь вам решить проблему.

Readme от автора

По ходу изложения я неоднократно отмечал, что Julia находится в состоянии разработки и даже еще не достиг уровня версии v.1.0.

Одна из целей написания данной книги состояла в том, чтобы помочь вам почувствовать себя комфортно во время изучения среды разработки на Julia и при взаимодействии с сообществом программистов на Julia.

Хотя на момент перевода книги приводимый в ней программный код работал в версии v0.4 (и был адаптирован к последней версии v0.4.6) на разных платформах, в следующих версиях могут возникнуть расхождения.

Проблемы могут быть вызваны:

1. Не рекомендованными для использования (устаревшими) функциями или синтаксическими конструкциями. Например, к последней версии v0.4.x был пересмотрен синтаксис для массивов `Any[]`, кортежей, функции `convert()` и пр.
2. Включением пакетов в Базу (стандартную библиотеку).
   Например, функции `DateTime` были включены в состав Базы для версии v0.4. Распределенные массивы были изъяты из Базы и существуют в виде пакета `DiscributedArrays.jl`, который необходимо включать в среду в версиях v0.4 и выше.
3. Изменениями в пакетах.
   Разработчикам пакетов необходимо некоторое время, чтобы принять к сведению изменения в функционале Базы и модифицировать свой программный код соответствующим образом. В некоторых случаях результатом таких изменений может стать изменение поведения всего пакета.

Кроме того, пакеты могут быть заменены на другие, программный код перераспределен либо пакет реструктурирован, в том числе путем создания подмодулей, изъятия либо переименования функций, модификации структуры типов.

Поэтому, если имеются проблемы с программным кодом, загляните в раздел issues стандартной библиотеки или конкретного пакета на Github либо попробуйте найти решение в Интернете. И наконец, задайте вопрос/заявите о проблеме в разделе issues на Github разработчика или на сайте вопросов и ответов для программистов Stackoverflow (http://stackoverflow.com/questions/tagged/julia-lang?page=1&sort=newest&pagesize=15).

Скорее всего, вы будете не первым, кто с этим столкнулся.

Комментарий переводчика

На сегодняшний день книга «Осваиваем язык Julia» представляет собой наиболее полное описание возможностей программирования на Julia. Год с момента изда-

ния оригинала книги, посвященной такому молодому и изменяющемуся языку как Julia, – это очень большой срок. Поэтому сразу стоит отметить, что весь материал книги приведен в соответствие с последней версией языка (v0.4.6 на июль 2016 года; в оригинале же рассматривалась стабильная на тот момент версия 0.3.x) и дополнен свежей информацией.

Книга содержит много аббревиатур и технических терминов из разных областей науки. Для удобства большинство аббревиатур расшифровывается в сносках, а для некоторых наименований в силу отсутствия единой терминологии приведены соответствующие варианты или пояснения.

В целом в книге принят сбалансированный подход к теории и программированию. Разумеется, можно найти более подробное изложение аспектов науки о данных и на основе других языков программирования. Однако основная задача книги заключается в том, чтобы объяснить читателю, каким образом можно использовать мощь языка Julia в научных вычислениях, в частности обращаясь к методам науки о данных.

Книга может быть интересной широкому кругу специалистов, в том числе в области машинного обучения, начинающим аналитикам данных, преподавателям, студентам, а также всем, кто интересуется программированием.

Выполнение примеров программного кода на Julia

Приведенные в книге примеры протестированы в операционных системах Windows 8.1/10 и Linux/Lubuntu 16.4.

Прилагаемые к книге адаптированные и скорректированные примеры программного кода должны находиться в подпапке `julia_projects` домашней папки пользователя (`/home/julia_projects` или `C:\Users\[ИМЯ_ПОЛЬЗОВАТЕЛЯ]\julia_projects`). Ниже приведена структура папки с проектами:

`Alice`	данные для примеров из папки `code` и записных книжек Jupyter
`code`	консольные программы
`data`	данные для примеров из папки `code` и записных книжек Jupyter
`gaston`	скорректированный графический пакет `Gaston`
`images`	файлы изображений из книги и полученные в результате выполнения программ
`Глава 01..Глава 10`	записные книжки (практически полностью дублируют папку `code`)
`Полезная информация`	дополнительные примеры кода, записные книжки и пр.

Рекомендуем всегда начинать консольную сессию с команды

```
julia> cd(joinpath(homedir(),"julia_projects"))
```

или аналогичной ей, чтобы сделать папку с проектами Julia текущей. Впрочем, эту команду можно прописать в файле `.juliarc.jl`, который находится в папке `homedir()` (`C:\Users\[ИМЯ_ПОЛЬЗОВАТЕЛЯ]` в Windows; образец файла имеется в папке с прилагаемыми примерами), и тогда эта команда будет исполняться при каждом открытии консоли интерпретатора REPL.

В простейшем случае работу большинства приводимых в книге примеров можно проверить в окне консоли интерпретатора REPL, просто копируя в него фрагменты программного кода из буфера обмена (вставка правой кнопкой мыши возле подсказки `julia>`). В Windows вместо консоли Julia можно использовать консоль, например, Cmder (http://cmder.net/) с расширенными возможностями, включая эмуляцию команд Unix; она идет в минимальной и полной версиях и не требует установки (Julia запускается изнутри консоли).

Как вариант, можно перенести все примеры в корневую папку пользователя; скажем, в Windows ее расположение будет таким: `C:\Users\[ИМЯ_ПОЛЬЗОВАТЕЛЯ]\julia_projects`. В этом случае для выполнения примеров сначала нужно перейти в папку `;cd julia_projects` и затем включать оттуда примеры в среду Julia для исполнения, используя команду `include` – к примеру, `include("code/asian_option.jl")`.

Остальные подразумевают использование интегрированной среды разработки Juno/Atom, плагина JuliaDT для интегрированной среды разработки Eclipse, локального сервера записных книжек Jupyter и перспективной облачной веб-службы JuliaBox (https://www.juliabox.org/), куда можно загружать локальные записные книжки и файлы с программным кодом на Julia для просмотра и исполнения.

В режиме разработки целесообразно в строке запуска консоли Julia прописать следующие переключатели:

- `-q, --quiet` – «тихий» запуск, то есть без стартового заголовка;
- `--depwarn={yes|no|error}` – активировать/деактивировать депрекационные предупреждения (`error` делает их ошибками);
- `--precompiled={yes|no}` – использовать прекомпилированный код из системного образа при наличии такового.

В следующей версии (0.5.0) предполагается новый переключатель:

- `--compilecache={yes|no}` – активировать/деактивировать обновляющую (incremental) прекомпиляцию модулей.

В целом строка должна иметь следующий вид:

```
julia.exe -q --precompiled=yes --depwarn=no
```

Установка среды разработки Julia

В Windows надо просто скачать установщик, соответствующий разрядности вашей операционной системы, и установить (http://julialang.org/downloads/). В Linux/Ubuntu сначала нужно выполнить небольшую подготовительную работу. Через терминал следует установить компиляторы:

```
sudo apt-get install gcc
sudo apt-get install g++
sudo apt-get install gfortran
```

Компоновщик:

```
sudo apt-get install cmake
```

И Git-клиент (факультативно):

```
sudo apt-get install git
```

А затем приступить собственно к установке Julia. Для операционных систем Ubuntu предусмотрен *Персональный архив пакетов* (*PPA*), который предоставляет возможность автоматического обновления программного обеспечения до самой свежей стабильной версии, в данном случае языка Julia. Чтобы воспользоваться PPA и установить Julia в Ubuntu версий 12.04 и выше, в терминале следует выполнить следующие команды:

```
sudo add-apt-repository ppa:staticfloat/juliareleases
sudo add-apt-repository ppa:staticfloat/julia-deps
sudo apt-get update
sudo apt-get install julia
```

С другой стороны, можно установить ночные сборки, выполнив следующие команды:

```
sudo apt-add-repository ppa:staticfloat/julianightlies
sudo apt-add-repository ppa:staticfloat/julia-deps
sudo apt-get update
sudo apt-get install julia
```

Сборка новых версий происходит каждую ночь. Если вы уже установили Julia и желаете обновиться до последней версии, выполните следующее:

```
sudo apt-get update
sudo apt-get upgrade
```

Установка пакетов в среду разработки Julia

После установки Julia следует обновить пакеты:

```
Pkg.update()
```

И затем установить ряд пакетов, которые будут использоваться в примерах:

```
Pkg.add("ASCIIPlots")
```

Кроме упомянутого пакета `ASCIIPlots` аналогичным образом следует установить следующие пакеты: `IJulia`, `Atom`, `Winston`, `Gadfly`, `PyPlot`, `HDF5`, `Match` и некоторые другие. Как правило, при установке пакетов автоматически устанавливаются другие необходимые пакеты, например `DataStructures`, `DataFrames`, `DataArrays`, `Distributions`, `StatsBase`, `StatsFuns`, `Compose` и др.

Если по какой-то причине в работе пакета возникнет сбой, то, как правило, его повторная сборка решает проблему:

```
Pkg.build("ZMQ")
```

Если пакет выдает депрекационное предупреждение о нецелесообразности использования какой-либо конструкции или выдает ошибку, то ситуацию можно исправить, внеся изменения в сам пакет и собрав его заново (кстати, именно так и работает Github). Например, пакет Match выдает предупреждение, что вместо модификатора String следует использовать AbstractString. Чтобы исправить ситуацию, смотрим на предупреждение – там будет указан конкретный файл модуля. Далее открываем папку (на примере Windows) C:\Users\[ИМЯ_ПОЛЬЗОВАТЕЛЯ]\.julia\v0.4, находим там папку пакета Match и в нем искомый файл.

В данном случае предупреждение было вызвано в matchmacro.jl в нижеприведенном блоке (String уже заменено на AbstractString):

```
# Regex strings (r"[a-z]*")
    elseif isexpr(expr, :macrocall) && expr.args[1] == symbol("@r_str")
        append!(info.tests, [:(isa($val, AbstractString)), :(Match.ismatch($expr, $val))])
        info
```

После сохранения изменений собираем пакет заново при помощи Pkg.build("Match"), и теперь он будет работать без назойливых предупреждений. После сборки следует обновить базу пакетов командой Pkg.update(). Кроме того, следует регулярно обновлять всю базу пакетов на предмет каких-либо изменений. В редких случаях, для того чтобы вновь установленный пакет определился в среде Julia, следует закрыть окно консоли и открыть его заново.

Перед началом использования пакета PyPlot в окне консоли интерпретатора REPL при помощи Pkg.status() необходимо проверить его наличие и, если его нет, установить, используя Pkg.add("PyPlot"). Неплохо также обновить или установить браузер Chrome или Firefox. По отзывам, они хорошо ладят с IJulia. Более того, у них имеются расширения для работы с файлами записных книжек .ipynb. При первом использовании пакета оператор using PyPlot выполнит его инициализацию, что потребует некоторого времени; при последующих запусках визуализация выполняется почти мгновенно.

Под Windows 8 пакет Winston может сразу не заработать, но, скорее всего, эта проблема разрешится сама собой в связи с обновлениями. И тем не менее проблема связана с конфликтом версий, в частности, 8.6 и 8.5, библиотек Tk/Tcl (скорее всего, вызвана установленным в системе дистрибутивом Anaconda Python). Чтобы решить проблему, нужно перейти в папку .julia (C:/Users/ИМЯ_ПОЛЬЗОВАТЕЛЯ/.julia/) , перейти на три уровня ниже ([ВЕРСИЯ]/Tk/deps/) и открыть в текстовом редакторе файл build.jl. Перейти в раздел зависимостей (dependencies) в самом начале файла и отредактировать две строки, удалив все, что содержит цифры 8.6 или 86, например tcl86t. Это касается зависимостей как для tcl, так и для tk. Затем перезапустить Julia и собрать Tk заново: Pkg.build("Tk"). Если проблемы продолжатся, переустановите Julia повторно.

Вообще желательно сперва установить дистрибутив Anaconda Python и только потом Julia.

Установка и удаление IDE Juno/Atom

В Windows следует скачать установщик универсального редактора программного кода Atom (https://atom.io/) и запустить его на выполнение.

Установка редактора Atom в Linux/Ubuntu через PPA выполняется следующим образом:

```
sudo add-apt-repository ppa:webupd8team/atom
sudo apt-get update
sudo apt-get install atom
```

Команды удаления IDE Atom/Juno:

```
sudo apt-get remove atom
sudo add-apt-repository --remove ppa:webupd8team/atom
```

Процедура настройки IDE Juno/Atom одинакова как для Windows, так и для Linux/Ubuntu (не забываем добавить соответствующий пакет в среду разработки Julia – `Pkg.add("Atom")`):

1. Запустить Atom и открыть окно настроек. Для этого выбрать элемент меню **File → Settings** (Файл → Настройки) и в окне настроек выбрать установку расширений (+**Install**).
2. Установить пакет-надстройку `uber-juno`, набрав в поле ввода его название (Juno работает поверх редактора Atom) и нажав кнопку **Install** (Установить). Он настроит среду Juno, загрузив ряд других пакетов; при этом может понадобиться вручную удалить и заново установить пакет `ink`.
3. Факультативно, чтобы получить LaTeX-подобное автодополнение, установите пакет `latex-completions`. Тогда, например, запись `\alpha<TAB>` будет генерировать символ Unicode α.

Установка и работа с плагином для Eclipse

Скачайте интерактивную среду разработки Eclipse для разработчиков на Java **Eclipse IDE for Java Developers** (https://www.eclipse.org/downloads/) в версии 4.5 или выше, например `eclipse-java-mars-2-win32-x86_64.zip`, и просто распакуйте архивный файл, после чего он будет готов к работе.

Затем скачайте плагин JuliaDT с сайта JuliaComputing.com (http://juliacomputing.com/blog/2016/02/06/Eclipse-JuliaDT.html). На сегодняшний день имеются два релиза плагина: v0.0.1 и v0.0.2 (https://github.com/JuliaComputing/JuliaDT/releases/tag/v0.0.1, https://github.com/JuliaComputing/JuliaDT/releases/tag/v0.0.2).

Для установки плагина в Eclipse выберите **Help → Install New Software...** (Справка → Установить новое ПО). Нажмите **Add...** (Добавить), далее **Archive** (Архив) и выберите предварительно загруженный с сайта зазипованный файл плагина. Наберите JuliaDT в поле для названия ПО и затем нажмите **Next → Next** (Далее → Далее), подтвердите лицензионное соглашение, нажмите **Finish** (Готово) и в конце перезагрузите Eclipse (инструкция по установке приведена на странице https://github.com/JuliaComputing/JuliaDT/wiki/Installation).

Затем следует интегрировать в среду Eclipse интерпретатор Julia. Для этого необходимо выполнить следующее (подразумевается, что Julia уже установлен):

В Eclipse выбрать **Preferences → Julia → Interpreters** (Свойства → Julia → Интерпретаторы). Затем нажать **Add**... (Добавить) и **Browse**... (Обзор). Выбрать исполнимый файл Julia, расположенный в подпапке `/bin`, затем нажать **OK** и в конце еще раз **OK**.

Для создания проектов на Julia следует выбрать **File → New → Project → Julia → Empty Julia Project** (Файл → Создать → Проект → Julia → Пустой проект Julia), нажать **Next** (Далее), затем в поле **Project name** (Имя проекта) ввести название проекта и в конце нажать **Finish** (Готово); инструкция по работе приведена на странице https://github.com/JuliaComputing/JuliaDT/wiki/Use-Guide.

Для выполнения программы на Julia надо в навигаторе пактов Package Explorer выбрать нужный файл с программой, правой кнопкой вызвать контекстное меню и в нем выбрать **Run as → REPL Julia Script** (Выполнить как → Сценарий Julia в консоли интерпретатора REPL). Результаты выполнения программы будут выведены во встроенную в Eclipse консоль либо (для графиков и изображений) отражены в отдельном окне или в браузере.

Работа с Julia в облаке JuliaBox

Чтобы начать работу в облаке JuliaBox, необходимо сначала зарегистрироваться; можно воспользоваться существующей учетной записью Google+. Для загрузки файла (записной книжки) в облако нужно нажать на значок файлового менеджера Files и в открывшемся окне перетащить файл с расширением `.ipynb` или `.jl` из папки на вашем компьютере в зону для перетаскивания Drag and Drop.

Данная веб-служба также предоставляет веб-консоль для написания и выполнения несложных фрагментов кода. Для этого возле подсказки `juser@juliabox:~$` нужно набрать `julia`, после чего консоль перейдет в нормальный режим ожидания команд, как в обычной локальной консоли интерпретатора REPL. Правда, следует учесть, что в консоли теперь используется последняя версия (0.4.6) языка.

Далее приводятся процедуры установки и удаления ряда компонентов в операционных системах Windows и Linux/Ubuntu. Соответствующие процедуры для Mac OS X здесь не рассматриваются.

Установка и удаление дистрибутива Anaconda Python

Anaconda – это полностью свободный дистрибутив Python (предназначенный в том числе для коммерческого использования и повторного распространения). Он содержит более 400 самых популярных библиотек Python для вычислений в области естественных наук, математики, инженерии и анализа данных.

Установка дистрибутива в Windows выполняется стандартным образом после загрузки с сайта бинарного файла дистрибутива. В Linux сначала необходимо скачать установщик – скриптовый файл с расширением .sh для оболочки Bash (https://www.continuum.io/downloads#_unix). На примере Anaconda Python версии 3 (для Python 3.5) команда установки выглядит так:

```
bash Anaconda3-4.0.0-Linux-x86_64.sh
```

Отметим, что инсталляция вступит в силу после того, как вы закроете и заново откроете окно терминала.

Для обновления дистрибутива Anaconda Python следует набрать в терминале следующую команду:

```
conda update conda
```

Удаление дистрибутива Anaconda:

```
rm -rf ~/anaconda
```

Более подробная информация по деинсталляции Anaconda содержится по указанной выше ссылке.

Чтобы удостовериться, что дистрибутив Anaconda установлен успешно, воспользуйтесь следующей командой:

```
conda --version
```

В результате будет выведен номер установленной версии.

Запуск сервера записных книжек Jupyter

В Windows и Linux/Ubuntu локальный сервер записных книжек запускается из командной строки двумя способами. Для этого открываем окно командной оболочки (`cmd`) в Windows или окно терминала в Linux/Ubuntu и просто набираем:

```
jupyter notebook
```

Или, как вариант:

```
ipython notebook -profile julia
```

Теперь интерактивная вычислительная среда IPython – это составная часть интегрированной среды Jupyter, поэтому второй вариант запуска не рекомендован к применению и впоследствии будет удален; к тому же в нем используется формат записной книжки предыдущей версии 3.0 (текущая – 4.0). Запуск из консоли интерпретатора REPL в обоих случаях одинаков:

```
using IJulia; notebook()
```

Факультативные настройки среды

Установка инструментальной среды Spyder

Spyder – это инструментальная среда для научных вычислений для языка Python (Scientific PYthon Development EnviRonment) для Windows, Mac OS X и Linux. Это простая, легковесная и бесплатная интерактивная среда разработки на Python, которая предлагает функционал, аналогичный среде разработки на MATLAB, включая готовые к использованию виджеты `PyQt4` и `PySide`: редактор исходного кода, редактор массивов данных `NumPy`, редактор словарей, консоли Python и IPython и многое другое.

Для пользователей Windows хорошая новость заключается в том, что Spyder уже включен в состав дистрибутива Anaconda Python; исполняемый файл находится в папке `C:\Users\[ИМЯ_ПОЛЬЗОВАТЕЛЯ\Anaconda3\Scripts\spyder.exe`. Чтобы установить среду Spyder в Linux/Ubuntu, используя официальный менеджер пакетов, нужна всего одна команда:

```
sudo apt-get install spyder
```

Чтобы установить с использованием менеджера пакетов pip:

```
sudo apt-get install python-qt4 python-sphinx
sudo pip install spyder
```

И чтобы обновить:

```
sudo pip install -U spyder
```

Управление пакетами .deb в Linux/Ubuntu

Установка пакетов .deb в терминале (пакет должен находиться под корневым каталогом home):

```
sudo dpkg -i [пакет].deb
```

или

```
sudo dpkg --install [пакет].deb
```

Удаление пакетов .deb:

```
sudo dpkg -r [пакет].deb
```

Удаление вместе с конфигурационными файлами:

```
sudo dpkg -P [пакет].deb
```

Глава 1

Среда разработки Julia

В этой главе мы выясним, что необходимо для того, чтобы начать работать с языком Julia, собрать его из исходников либо получить предварительно собранные двоичные файлы (или бинарники). Для программирования на Julia существует интегрированная среда разработки (IDE) Juno, которая устанавливается отдельно (http://junolab.org/), а также плагин JuliaDT для среды разработки Eclipse (https://github.com/JuliaComputing/JuliaDT). Язык Julia может выполняться в интерактивной среде программирования Jupyter локально и в Интернете на веб-сайте https://juliabox.org/. Julia – это высокоуровневый, высокоэффективный язык динамического программирования, предназначенный для технических вычислений, который выполняется в Linux, OS X и Windows. В этой главе мы обратимся к его сборке из исходников в операционной системе CentOS Linux, а также скачаем в виде предварительно собранного двоичного дистрибутива. Как правило, мы будем пользоваться версией v0.4.x, которая на июль 2016 года остается стабильной; тем не менее версия v0.5.x находится в стадии текущих разработок, и ее ночные сборки можно скачать с веб-сайта языка[1].

Введение

Язык Julia (http://julialang.org/) впервые был представлен в феврале 2012 года после нескольких лет разработки в Массачусетском технологическом институте (MIT).

Все основные разработчики – Джефф Безансон (Jeff Bezanson), Стефан Карпински (Stefan Karpinski), Вирал Шах (Viral Shah) и Алан Эдельман (Alan Edelman) – продолжают играть активную роль в разработке языка и отвечают за его ядро, а кроме того являются авторами и соавторами многих программных пакетов.

Язык Julia является открытым, поэтому все доступно для просмотра. Небольшое количество кода написано на C/C++, а также немного на Lisp и Scheme[2],

---

[1] На момент сдачи перевода книги в печать (осень 2016 г.) текущей стабильной версией среды разработки Julia является v4.6.0. В переводе учтены все изменения, произошедшие с момента издания данной книги в оригинале, и внесены соответствующие исправления в текст приводимых примеров кода. – *Прим. перев.*

[2] *Scheme* – это функциональный язык программирования, один из двух наиболее популярных в наши дни диалектов языка Lisp (другой популярный диалект – это Common Lisp), созданный в MIT в 1975 году. – *По материалам Википедии.*

но подавляющая часть ядра написана (очень хорошо) непосредственно на Julia, и с ним можно ознакомиться на досуге. Если вы хотите писать на Julia безукоризненный программный код, это именно то место, где можно почерпнуть вдохновение. К концу этой главы мы быстро пробежимся по дереву исходного кода Julia в рамках исследования среды разработки Julia.

Язык Julia часто сравнивают с такими языками программирования, как Python, R и MATLAB. Важно понять, что Python и R появились еще в середине 1990-х, а MATLAB – в 1984 году. Поскольку MATLAB является патентованной разработкой (® MathWorks), то существует несколько клонов, в особенности GNU Octave, который датируется той же самой эпохой, что и Python и R. То, насколько далеко язык продвинулся в своем развитии, – заслуга первоначальных разработчиков и многих энтузиастов, которые присоединились к этой работе. Julia использует веб-службу GitHub в качестве хранилища как для своих исходных текстов, так и для зарегистрированных программных пакетов. Хотя целесообразно, чтобы на компьютере был установлен Git-клиент[1], в обычных условиях взаимодействие с Git от пользователя в основном скрыто, поскольку в состав Julia включена его рабочая версия, завернутая в диспетчер пакетов (Pkg), который можно вызывать из консоли. Пока в Julia нет простой встроенной графики, зато существует несколько разных графических пакетов, и им будет посвящена отдельная глава.

Философия

Julia разрабатывался с прицелом на научные вычисления. Разработчики языка рассказывают, что они пришли в проект, имея за плечами огромный багаж навыков программирования на Lisp, Python, Ruby, R и MATLAB. Некоторые вроде меня даже утверждают, что являются выходцами из хакерской среды на Perl. Однако все они хотели бы иметь в своем арсенале быстрый компилируемый язык, такой как C или Fortran, поскольку нынешние языки, перечисленные выше, ужасно медленные.

Цитируя группу разработчиков,

> *«нам нужен общедоступный язык с либеральной лицензией. Нам нужна скорость C с динамизмом Ruby. Нам нужен язык с гомоиконностью и настоящими макрокомандами, как в Lisp, но с очевидной и понятной математической записью выражений, как в MATLAB. Нам нужно нечто столь же удобное для общего программирования, что и Python, столь же простое для статистических расчетов, что и R, столь же естественное для обработки строк, что и Perl, столь же мощное для решения линейных алгебраических задач, что и MATLAB, столь же хорошее для склеивания программ, как и командный процессор. Нечто простое в изучении и одно-*

[1] *Git* – распределенная система управления версиями исходного программного кода. GitHub – крупнейшая веб-служба для хостинга ИТ-проектов и их совместной разработки, которую также называют «социальной сетью для разработчиков». – *Прим. перев.*

временно радующее самых серьезных хакеров. Язык должен быть интерактивным и компилируемым.

(*Мы ведь уже сказали, что он должен быть быстрым, как C?*)»
http://julialang.org/blog/2012/02/why-we-created-julia

Достижение поставленной цели и разработка языка с самого начала стали возможными благодаря появлению компиляции на основе низкоуровневой виртуальной машины (*Low-Level Virtual Machine*, *LLVM*), которая сделала двуязычный подход в большой степени избыточным.

Язык Julia разрабатывался таким образом, чтобы быть похожим на другие языки сценариев, и поэтому должен быть прост в изучении для любого, кто знаком с Python, R и MATLAB. Синтаксически он наиболее близок к MATLAB, однако, что важно, не является его клоном-заменителем: у них имеется целый ряд принципиальных отличий, к которым мы обратимся позже.

Важно не потерять голову и не рассматривать Julia как соперника Python и R. На самом деле эти языки с успехом дополняют друг друга, и мы покажем это на примерах. Конечно, Julia не предполагался как такая замена, но определенные особенности его работы делают его идеальным для использования в научном сообществе.

Роль в науке о данных и в области больших данных

Julia изначально разрабатывался с прицелом на научные расчеты. Несмотря на то что термин «наука о данных» был введен еще в 1970-х, ему было отведено видное место лишь в 2001 году в статье Уильяма С. Кливленда *Data Science: An Action Plan for Expanding the Technical Areas of the Field of Statistics* («Наука о данных: План действий по расширению технических областей в статистике»). Почти параллельно с разработкой языка Julia развивалась наука о данных и росла потребность в специалистах в этой области.

Что такое наука о данных?

Вот одно из определений:

> ***Наука о данных*** *– это практическая дисциплина, которая занимается изучением методов обобщаемого извлечения знаний из данных. Она включает различные составляющие и основывается на методах и теориях из многих областей знаний, включая обработку сигналов, математику, вероятностные модели, машинное и статистическое обучение, программирование, технологии данных, распознавание образов, теорию обучения, визуальный анализ, моделирование неопределенности, организацию хранилищ данных, а также высокоэффективные вычисления с целью извлечения смысла из данных и создания продуктов обработки данных.*

Если это звучит знакомо, так и должно быть! Именно в этом заключались конкретные цели, поставленные на начальном этапе проектирования Julia. Чтобы заполнить вакуум, большинство аналитиков данных (data scientists) обратились к Python и в меньшей степени к R. Одну из основных причин роста популярности Python и R можно объяснить непосредственно интересом к науке о данных.

И потому основная задача этой книги заключается в том, чтобы показать вам как начинающему аналитику данных, почему вам следует рассмотреть возможность использования Julia и, если мы вас убедим, – как это сделать.

Помимо науки о данных другие «новички» – это большие данные и облачные вычисления. Большие данные первоначально входили в сферу влияния Java в основном из-за внедрения платформы Hadoop/HDFS, которая, будучи написанной на Java, сделала удобным программирование алгоритмов MapReduce на этом языке либо на любом другом, который выполняется в виртуальной машине Java (JVM), что в итоге ведет к раздутому до непристойности объему шаблонного программного кода.

Однако с внедрением механизма YARN и потоковой обработки в Hadoop парадигма обработки больших данных стала открытой для более широкого разнообразия подходов. Python начинает считаться альтернативой Java, но, если присмотреться, Julia тоже имеет превосходные перспективы в этой категории.

Сопоставление с другими языками

Язык Julia известен своим быстродействием. Домашняя страница основного вебсайта Julia, по состоянию на июль 2014 года, содержала ссылки на эталонные тесты (бенчмарки) относительно языка C, которые показаны в табл. 1.1 (чем меньше, тем лучше производительность; C = 1.0).

Таблица 1.1

	Fortran	Julia	Python	R	MATLAB	Octave	Mathematica	Java Script	Go
fib	0.26	0.91	30.37	411.31	1992.0	3211.81	64.46	2.18	1.0
mandel	0.86	0.85	14.19	106.97	64.58	316.95	6.07	3.49	2.36
pi_sum	0.80	1.00	16.33	15.42	1.29	237.41	1.32	0.84	1.41
rand_mat_stat	0.64	1.66	13.52	10.84	6.61	14.98	4.52	3.28	8.12
rand_mat_mul	0.96	1.01	3.41	3.98	1.10	3.41	1.16	14.60	8.51

Таблица 1.2 показывает результаты аналогичных эталонных испытаний по состоянию на июль 2016 года.

Таблица 1.2

	Fortran gcc 5.1.1	Julia 0.4.0	Python 3.4.3	R 3.2.2	MATLAB R2015b	Octave 4.0.0	Mathematica 10.2.0	JavaScript V8 3.28.71.19	Go go1.5
fib	0.70	2.11	77.76	533.52	26.89	9324.35	118.53	3.36	1.86
parse_int	5.05	1.45	17.02	45.73	802.52	9581.44	15.02	6.06	1.20
quicksort	1.31	1.15	32.89	264.54	4.92	1866.01	43.23	2.70	1.29
mandel	0.81	0.79	15.32	53.16	7.58	451.81	5.13	0.66	1.11
pi_sum	1.00	1.00	21.99	9.56	1.00	299.31	1.69	1.01	1.00
rand_mat_stat	1.45	1.66	17.93	14.56	14.52	30.93	5.95	2.30	2.96
rand_mat_mul	3.48	1.02	1.14	1.57	1.12	1.12	1.30	15.07	1.42

Эталонные испытания пользуются печальной известностью, нередко вводя в заблуждение; действительно, перефразируя известное высказывание, есть ложь, подлая ложь и эталонные испытания.

Сайт языка Julia прилагает все усилия, чтобы четко сформулировать параметры этих испытаний, предоставляя подробную информацию об используемом типе процессора рабочей станции, тактовой частоте ЦП, объеме ОЗУ и т. д. – и развернутой операционной системе. Для каждого теста предлагаются версия программного обеспечения и любые сторонние пакеты или библиотеки; например, для теста `rand_mat` в Python используется библиотека `NumPy`, а в C, Fortran и Julia соответственно – библиотека `OpenBLAS`.

Для проверки производительности языка Julia специально отведен раздел вебсайта http://julialang.org/benchmarks/.

Исходный код для всех тестов доступен на GitHub. Это не только код на Julia, но также на C, MATLAB, Python и т. д. Более того, этот список пополняется примерами на дополнительных языках, и там можно найти бенчмарки для выполнения испытаний на Scala и Lua: https://Github.com/JuliaLang/julia/tree/master/test/perf/micro.

Эта таблица имеет практическую пользу и в другом плане: в ней перечислены все основные языки, с которыми Julia сопоставляется. Тут нет никаких неожиданностей – разве что только диапазон времени выполнения.

1. *Python*. Он стал de facto языком науки о данных, а его диапазон доступных модулей поистине впечатляет. Обе версии – 2 и 3 – широко используются; последняя из двух не является супермножеством предыдущей и приблизительно на 10% медленнее. В целом производительность Julia на порядок выше, чем у Python, поэтому часто устоявшийся программный код на Python компилируют или переписывают на C.
2. *R*. Начав свою карьеру в качестве общедоступной версии коммерческого статистического пакета S+ (® TIBCO Software Inc.), он в значительной мере заменил его в проектах в области статистики, имея большой набор вспомогательных пакетов. Это однопотоковый язык, что и объясняет неутешительное время выполнения, с непрямым распараллеливанием задач вычисления. Но в R есть очень хорошие графические возможности и пакеты визуализации данных.
3. *MATLAB/Octave*. MATLAB – это коммерческий продукт (® MathWorks) для матричных операций; отсюда разумное время выполнения последних двух эталонных испытаний. Тем не менее по другим испытаниям показатели очень продолжительные. GNU Octave – это бесплатный клон MATLAB. Он был разработан в целях совместимости, а не эффективности, что и объясняет более длительное время выполнения.
4. *Mathematica*. Еще один коммерческий продукт (® Wolfram Research) для математических задач общего назначения. У него нет никакого очевидного клона, несмотря на то что платформа Sage является общедоступной и использует Python в качестве вычислительного движка; поэтому его показатели подобны Python.

5. *JavaScript* и *Go*. Они связаны между собой, поскольку оба используют движок Google V8. Перед выполнением программы этот движок компилирует исходный код в собственный машинный код; отсюда превосходные показатели производительности, однако оба языка в большей степени предназначены для разработки веб-приложений.

Таким образом, Julia, похоже, является идеальным языком для решения задач в области науки о данных. Важно понимать, что многие встроенные в R и Python функции не реализованы исходно, а написаны на C. Производительность Julia примерно такая же, что и у C, поэтому языку Julia не удастся быть эффективнее R или Python, если значительная часть работы, которую вы делаете на R или Python, сводится к вызовам встроенных функций без выполнения любой явной итерации или рекурсии.

Зато когда вы начинаете выполнять заказную работу, Julia непременно вступит в свои права. Это совершенный язык для опытных разработчиков на R или Python, которые пытаются создать продвинутые инструменты в пределах этих языков. Как правило, альтернативой Julia является C; R предлагает эту возможность посредством *Rcpp*, а Python – посредством *Cython*[1].

Для сотрудничества между Julia, с одной стороны, и R и/или Python, с другой, имеется больше возможностей, чем для соперничества, пусть и не все разделяют это мнение.

Характеристики

Julia – это свободный общедоступный (в соответствии с лицензией MIT) язык программирования, исходный код которого имеется на сайте GitHub.

Ветеранам программирования он представляется чем-то похожим на MATLAB. Блоки инструкций, создаваемые операторами `for`, `while` и `if`, завершаются ключевым словом `end`, а не `endfor`, `endwhile` и `endif`, или с использованием знакомой синтаксической конструкции с фигурными скобками {}. Однако это не клон MATLAB, и исходный код, написанный для MATLAB, в Julia выполняться не будет.

Ниже приводятся некоторые характеристики языка Julia:

- разработан для параллелизма и распределенных вычислений (многоядерных и кластерных);
- функции C вызываются напрямую (нет необходимости в оберточном коде или специальных программных интерфейсах);
- мощные подобные командному процессору возможности по управлению другими процессами;
- Lisp-подобные макрокоманды и другие средства метапрограммирования;
- быстродействие и компактность определяемых пользователем типов такие же, что и у встроенных;

---

[1] Cython – язык программирования, упрощающий написание модулей C/C++ для Python. Пакет Rcpp обеспечивает функции на R и классы на C++, которые обеспечивают бесшовную интеграцию R с C++. – *По материалам Википедии.*

- *динамический (JIT)* компилятор на основе низкоуровневой виртуальной машины (LLVM), который позволяет Julia приближаться и часто соответствовать производительности C/C++;
- обширная библиотека математических функций (написанных на Julia);
- интегрированные зрелые, лучшие в своем роде библиотеки C и Fortran для линейной алгебры, генерации случайных чисел, *быстрого преобразования Фурье* (*БПФ*, *FFT)* и обработки строк.

Ядро Julia реализовано на C и C++, а синтаксический анализатор языка – на Scheme; платформа LLVM-компилятора используется для динамической (JIT) трансляции байт-кода в машинные коды во время исполнения программы.

Стандартная библиотека написана непосредственно на Julia при помощи библиотеки `libuv`[1] программной платформы Node.js в целях эффективного межплатформенного ввода-вывода.

Julia имеет развитые выразительные средства для конструирования и описания объектов, которые к тому же могут факультативно использоваться для объявлений типов. Он дает возможность определять поведение функции по многим сочетаниям типов аргументов посредством множественной диспетчеризации, которая является основополагающим элементом в конструкции языка.

Julia может использовать программный код на других языках программирования, напрямую обращаясь к подпрограммам на C или Fortran, хранящимся в динамически загружаемых библиотеках (shared libraries) либо в библиотеках DLL, известных под общим названием *динамических библиотек общего пользования*. Эта отличительная особенность синтаксиса языка будет более подробно обсуждаться в дальнейшем.

Кроме того, имеется возможность посредством пакета *PyCall* взаимодействовать с языком Python, который используется в пакете *IJulia*, реализующем среду программирования на основе записных книжек.

Начало работы

Начать программировать на Julia очень просто. Первым делом следует заглянуть на основной веб-сайт языка: http://julialang.org/. Он не перегружен графикой. На домашней странице вы найдете логотип Julia и ряд ссылок на основные разделы сайта.

Документация по Julia всеобъемлющая и находится в разделе документов docs: http://docs.julialang.org/. Там есть дальнейшие ссылки на руководство по языку Julia, по стандартной библиотеке и системе программных пакетов. Их все мы обсудим позже. Кроме того, документация может быть загружена в виде файла PDF, zip-файла страниц HTML или файла в формате ePub.

---

[1] libuv – библиотека с мультиплатформенной поддержкой и акцентом на асинхронном вводе-выводе. Была разработана в первую очередь для использования программной платформой Node.js (https://github.com/libuv/libuv). – *Прим. перев.*

Исходные тексты Julia

Сейчас мы обратимся к материалам для скачивания. Перейдя в раздел downloads, можно получить ссылки на 32- и 64-разрядные стандартные дистрибутивы для Windows, Mac OS X, CentOS и Ubuntu в стабильной версии либо в виде ночного снимка разработки. Поэтому большинству начинающих пользователей требуется просто скачать файл и выполнить стандартную процедуру инсталляции.

В Windows это делается путем запуска на исполнение загруженного с сайта `.exe`-файла, который извлечет Julia в папку `C:\Users\[ИМЯ_ПОЛЬЗОВАТЕЛЯ]\AppData\Local\Julia-[ВЕРСИЯ]` (хотя место установки можно изменить, например на `C:\`). В папке установки будет находиться ярлык `julia`, указывающий на расположенный в подпапке `bin` исполнимый файл. Ярлык можно вынести на Рабочий стол и/или использовать для запуска консоли Julia. Когда инсталляция завершится, следует создать переменную системного окружения под названием `JULIA_HOME` и присвоить ей значение каталога `\bin` под папкой, куда был установлен Julia. Важно указать именно каталог `/bin`, а не просто каталог `JULIA`. Затем следует добавить значение `;%JULIA_HOME%` к вашей переменной системного окружения `PATH` с тем, чтобы можно было запускать скрипты из любого каталога.

В Mac OS X пользователи должны щелкнуть по скачанному `.dmg`-файлу, чтобы запустить на исполнение образ диска, и перетащить *значок* `app` в папку `Applications`. В Mac OS X вам будут предложены варианты продолжения операции, поскольку источник был скачан из Интернета и потому не считается безопасным.

Процесс деинсталляции также прост. В Windows следует удалить папку `julia`, в Mac OS X – `Julia.app`. Чтобы выполнить «чистую» деинсталляцию, потребуется также удалить несколько скрытых файлов/папок, и этот вопрос будет рассмотрен после обсуждения системы пакетов.

В Linux/Ubuntu все немного сложнее, поскольку необходимо добавить ссылку на *персональный архив пакетов* (*PPA*) для вашей системы. Для того чтобы исполнить следующие команды, вы должны обладать корневой привилегией:

```
sudo apt-get add-repository ppa:staticfloat/juliareleases
sudo add-apt-repository ppa:staticfloat/julia-deps
sudo apt-get update
sudo apt-get install julia
```

> **Скачивание кода примеров**
> Вы можете скачать файлы с кодом примеров из этой книги на сайте http://www.dmkpress.com/.

Релизы предоставлены Эллиотом Саба; также имеется отдельный PPA для ночных снимков: `ppa:staticfloat/julianightlies`.

Персональный архив пакетов PPA добавляется всего один раз, и для установки обновлений нужно лишь исполнить следующую команду:

```
sudo apt-get update
```

Сборка из исходников

Операционная система Ubuntu является составной частью семейства дистрибутивов Linux под эгидой операционной системы Debian; другими представителями этого семейства является сама Debian, а также *дистрибутив Linux Mint в редакции Debian* (*LMDE*) и Knoppix, **которые базируются на Debia**n. Все могут устанавливать пакеты в формате DEB и использовать упомянутую выше команду `apt-get`.

Еще одно ключевое семейство Linux основано на дистрибутиве Red Hat: Red Hat Enterprise, CentOS, Fedora, Scientific Linux и т. д. Они используют другой механизм управления пакетами, основанный на файлах пакетов в формате RPM. Есть также дистрибутивы на основе операционных систем SUSE, Mandriva и Slackware.

Их исчерпывающий список можно найти на странице в Wikipedia: http://en.wikipedia.org/wiki/List_of_Linux_distributions.

Опять же в разделе материалов для скачивания на `julialang.org` имеется соответствующая ссылка. Julia использует GitHub в качестве репозитория для своего исходного дистрибутива, а также для разнообразных пакетов Julia. Далее будет рассмотрена процедура инсталляции среды разработки в операционной системе CentOS, которая представляет собой популярную редакцию сообщества Red Hat.

Инсталляция в CentOS

Операционную систему CentOS можно скачать в виде образа ISO с http://www.centos.org и записать на DVD. Ее можно инсталлировать как замену существующей системе Windows или запускать наряду с Windows в двухзагрузочной конфигурации.

В стандартном варианте CentOS поставляется без команды `git`; после инсталляции операционной системы первая задача будет состоять в том, чтобы ее установить. Для этого и других процессов установки используется менеджер пакетов `yum` (Yellowdog Updater and Modified).

Вам потребуется наличие корневой привилегии или полномочия суперпользователя, поэтому, как правило, в эмуляторе терминала вы наберете `su`:

```
su
(набрать свой пароль)
yum update
yum install git
```

Yum доставит исходные коды Git репозитория Red Hat, выведет список всего того, что необходимо инсталлировать, и выдаст подсказку с приглашением нажать `Y/N` для продолжения.

После установки Git нужно взять исходники Julia на GitHub, выполнив для этого следующую команду:

```
git clone git://Github.com/JuliaLang/julia.git
```

(Вместо `git://` также можно использовать `https://`, если вы находитесь позади сетевого экрана.)

```
git clone git://Github.com/JuliaLang/julia.git
Cloning into 'julia'...
remote: Counting objects: 97173, done.
remote: Compressing objects: 100% (24020/2)
```

В результате в текущем расположении будет создана подпапка с именем `julia` со всеми исходниками и документацией.

Чтобы выполнить сборку, потребуются инструменты разработки, которые обычно в стандартном дистрибутиве CentOS не присутствуют – в частности, компиляторы GCC, g++ и gfortran (соответственно для C/C++, GNU C++ и Fortran).

Их можно инсталлировать следующим образом:

```
sudo yum install gcc
sudo yum install g++
sudo yum install gfortran
```

Или, что более удобно, в комплекте для групповой инсталляции:

```
sudo yum groupinstall 'Development tools'
```

Другие инструменты (которые обычно присутствуют), такие как GNU Make, Perl и patch, также необходимы, но команда групповой инсталляции `groupinstall` позаботится и о них, если они отсутствуют. Мы обнаружили, что инсталляция на Fedora 19 на деле закончилась неудачно, потому что макропроцессор M4[1] не был найден; но опять же достаточно указать команду `yum install m4`, и процесс возобновится с того самого места, где инсталляция перестала работать.

Итак, чтобы продолжить работу со сборкой, переходим в клонированную папку `julia` и выполняем команду компиляции `make`. Обращаем внимание завзятых Linux-программистов, занимающихся сборкой из открытых исходников, на то, что этап конфигурирования не требуется! Все необходимые компоненты предположительно уже имеются (иначе процесс компиляции дал бы сбой), и исполняемый файл создается в папке `julia`, а значит, этап `make install` отсутствует.

Процесс сборки может занять массу времени и выдать много предупреждающих сообщений касательно отдельных исходных файлов, но по окончании в папке сборки появится файл с именем `julia`. Он представляет собой символическую ссылку на исполняемый файл, фактически находящийся в папке `usr/bin`.

Таким образом, если все инструменты на месте, то процесс, как правило, выглядит следующим образом:

```
[malcolm@localhost] cd ~
[malcolm@localhost] mkdir julia
[malcolm@localhost] cd julia
[malcolm@localhost julia] git clone git://github.com/JuliaLang/julia.git
[malcolm@localhost julia] cd julia
[malcolm@localhost julia] make
```

---

[1] Макропроцессор m4 предназначен для макрогенерации на предварительном проходе в различных языках. Макрогенерация означает копирование входного символьного потока в выходной с подстановкой макросов по мере их появления. – *По материалам Википедии.*