Язык программирования Julia

Рыжков Александр

ММП ВМК МГУ

8 октября 2014 года

Содержание

О языке Julia

- Высокоуровневый
- Высокопроизводительный
- Поддержка всех операционных систем
- Простой и удобный синтаксис
- Огромное количество подключаемых пакетов

Сферы применения

На текущий момент Julia может быть использована для:

- Анализа и обработки изображений
- Классификации текстов и тематического моделирования
- Оптимизации функций
- Анализа статистических моделей
- Работы с вероятностными распределениями
- Прогнозирования временных рядов
- Создания и публикации интерактивных моделей
- ...

Базовые операции

- Синтаксис языка совмещает в себе MatLab и Python
- Есть возможность писать имена переменных и знаки операций в юникоде
- Простое и удобное создание однострочных функций
- Создание многомерных массивов через однострочные конструкции с циклами
- Выполнение функций командной строки

Графические возможности

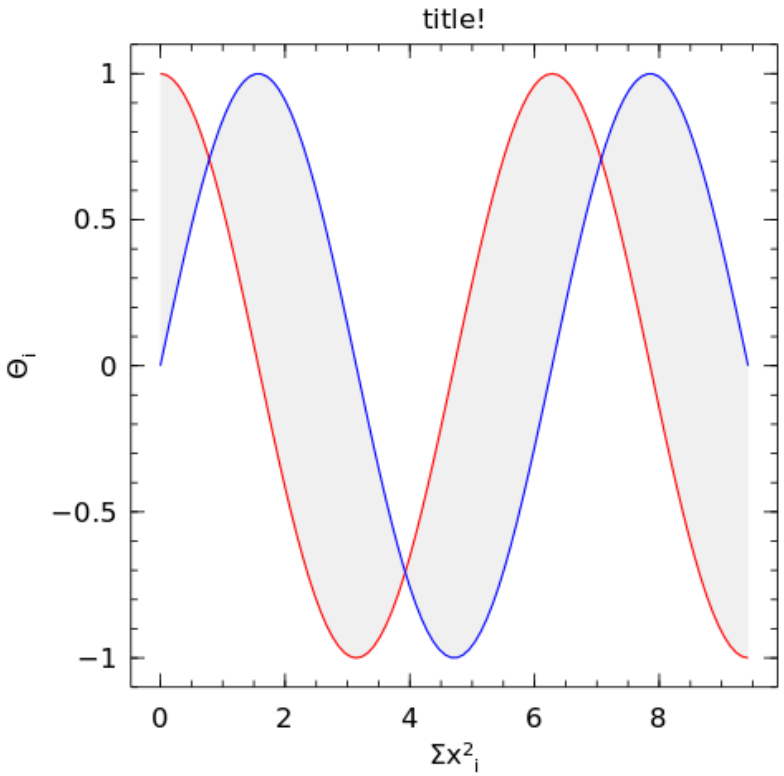
Пакеты, которые могут использоваться для визуализации:

- PyPlot — интерфейс *matplotlib* под Julia
- Winston — библиотека для отрисовки графиков, похожих на графики MatLab, с дополнительным набором удобных функций
- Gadfly — продвинутая библиотека для визуализации различных датасетов, создающая интерактивный график внутри ячейки iJulia notebook

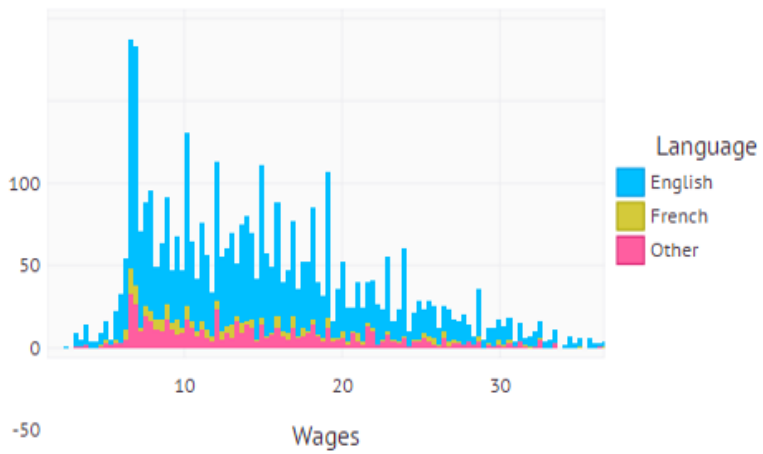
Графические возможности

Примеры графиков:

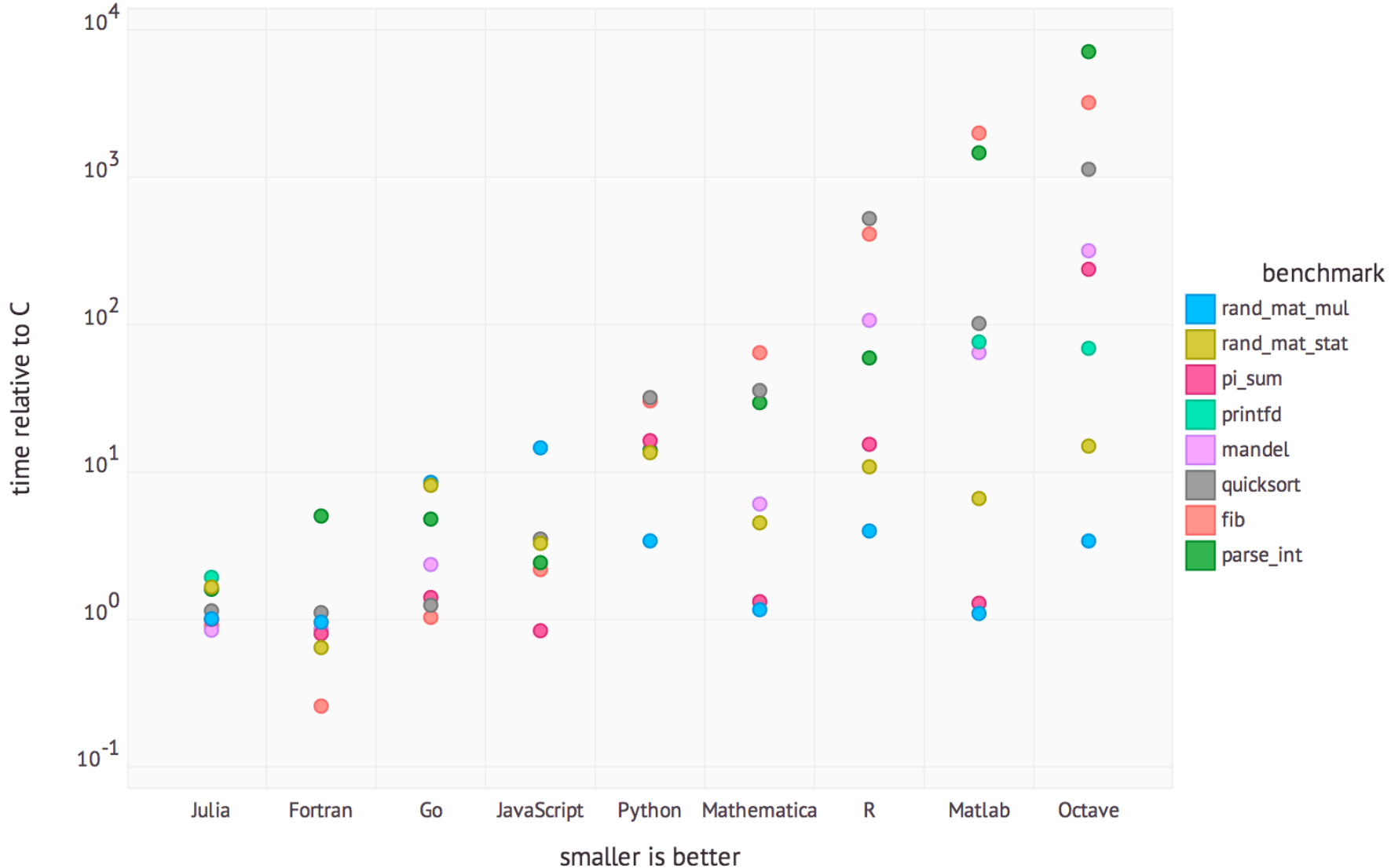
(a) Winston



(b) Gadfly



Быстродействие



Быстродействие

Основные факторы **снижения** производительности:

- Использование глобальных переменных
- Неявное приведение типов
- Использование функций без предвариательной компиляции

Написание макросов

Макросы в Julia могут быть использованы для:

- Упрощения написания программы
- Использования заготовленных алгоритмов
- Выполнения частых действий

Использование Python и C

Взаимодействие с Python:

- Вариант 1 — сложный:

```
using PyCall
pyeval("2+2") #=> 4
pyeval("str(5)") #=> "5"

# doing things by hand, for fun :)
math = pyimport(:math) #=> PyObject <module 'math'>
pycall(math["sin"],Float64,1) #=> 0.8414709848078965
```

- Вариант 2 — простой:

```
using PyCall
@pyimport pylab
x = linspace(0,2*pi,1000); y = sin(3*x + 4*cos(2*x));
pylab.plot(x, y; color="red", linewidth=2.0, linestyle="--")
pylab.show()
```

Использование Python и C

Взаимодействие с языком C:

```
function getenv(var::String)
  val = ccall( (:getenv, "libc"),
               Ptr{Uint8}, (Ptr{Uint8},), var)
  if val == C_NULL
    error("getenv: undefined variable: ", var)
  end
  bytestring(val)
end
```

Выполнение разных действий в зависимости от ОС:

```
@linux? (
          begin
              some_complicated_thing(a)
          end
        : begin
              some_different_thing(a)
          end
        )
```

Постановка задачи

Что за символ изображен на картинке?
Примеры изображений:

Задача классификации на $26 + 26 + 10 = 62$ класса!

Результаты

Текущие результаты работы алгоритмов:

Язык	Алгоритм	Результат	Время работы
Julia	KNN + CV	0.42408	$\approx$ 13 минут
Julia	RF (500 деревьев)	0.49640	$\approx$ 25 минут
Python	KNN + CV	0.42408	$\approx$ 4 минуты
Python	RF (500 деревьев)	0.48331	$\approx$ 8.5 минут

В случае использования советов по повышению быстродействия программы, алгоритмы на Julia работают примерно на $5 - 7\%$ быстрее реализаций на Python (только в случае предварительной компиляции).
Стоит также отметить, что замеры проводились на виртуальной машине: гостевая ОС — Lubuntu 14.04, домашняя ОС — Windows 7.

Выводы

Преимущества Julia:

- Простой и удобный синтаксис
- Наличие JIT-компиляции
- Увеличение числа специализированных пакетов
- Быстрый рост сообщества

Недостатки Julia:

- Необходимо предварительно запускать функции для того, чтобы произошла компиляция
- Проблемы с работой iJulia в Windows
- Недостаточная популярность в сообществе ML

Литература и ссылки

- *http://julialang.org/*
- *https://github.com/JuliaLang/julia*
- *http://juliacon.org/*
- *http://math.mit.edu/~stevenj/Julia-cheatsheet.pdf*
- *http://bogumilkaminski.pl/files/julia_express.pdf*
- *http://julia.readthedocs.org/en/release-0.2/packages/packagelist/*

Визуализация

Полная история создания и развития проекта Julia на GitHub.com:

http://www.youtube.com/watch?v=rbJ611WEX78

Вопросы

Спасибо за внимание!

Вопросы?