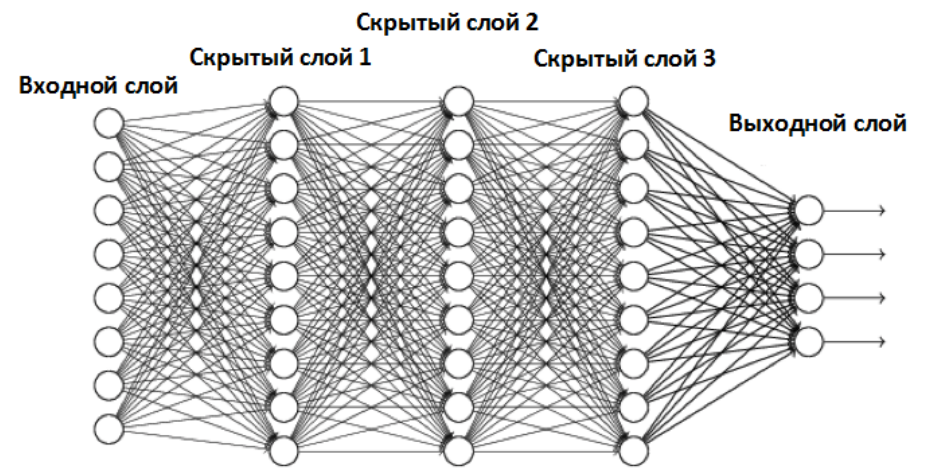


[www.machinelearningmastery.ru](http://www.machinelearningmastery.ru)

Машинное обучение, нейронные сети,  
искусственный интеллект

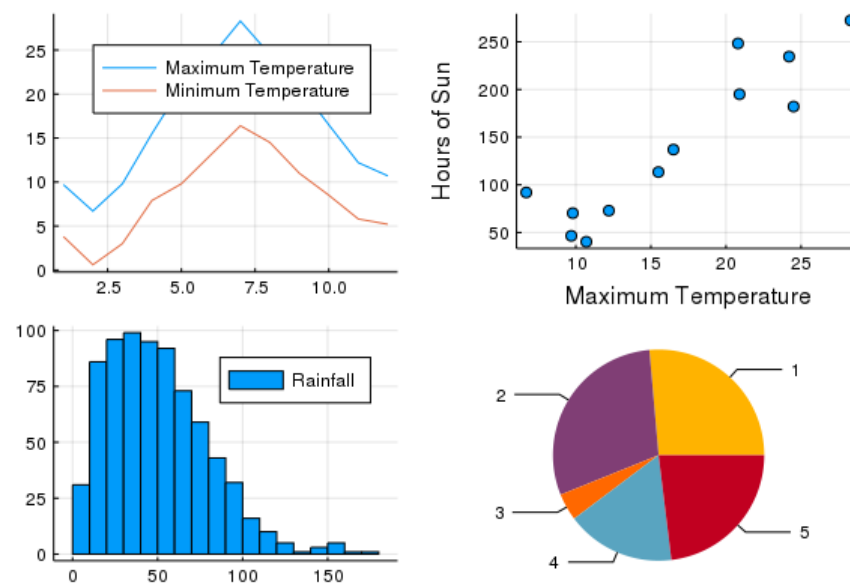


[Home](#) / Начало визуализации данных с Julia и JuliaBox

Home

## Начало визуализации данных с Julia и JuliaBox

🕒 Дата публикации Sep 2, 2019



(Первоначально опубликовано в [CodeEd](#) - увидеть ниже)

## [Визуализация данных с помощью Julia и JuliaBox](#)

[Это замечательно, если вам приходится иметь дело с большими наборами данных, которые требуют большой обработки. Юля тоже намного меньше ...](#)

[projectcodeed.blogspot.com](http://projectcodeed.blogspot.com)

Юлия - относительно новый язык для анализа данных. Он имеет синтаксис высокого уровня и прост в использовании и понимании. Некоторые назвали это новым Питоном.

Однако, в отличие от Python, это язык соответствия, а это означает, что, хотя его так же легко писать, как и Python, он работает намного быстрее, потому что он конвертируется в низкоуровневый код, который легче понять компьютеру. Это замечательно, если вам приходится иметь дело с большими наборами данных, которые требуют большой обработки.



Джулия также гораздо менее суетлива относительно того, как устроена программа, чем Python. (Python - один из немногих языков, который вынуждает программиста компоновать код особым образом, с определенными частями, правильно выделенными фиксированным количеством пробелов или табуляций. Это облегчает чтение кода, но может быть немного сложным, если у вас нет хороший редактор.)

Julia обладает всеми функциями, которые можно ожидать от современного языка программирования, но здесь мы рассмотрим возможности визуализации данных Julia. Они впечатляют и просты в использовании.

# JuliaBox

Вам не нужно устанавливать Julia, чтобы следовать этой статье или создавать собственные визуализации данных Julia. Эти примеры были записаны в блокноте Jupyter с использованием бесплатной учетной записи на **JuliaBox** \*\*,



JuliaBox - это онлайн-среда для создания ноутбуков Jupyter. Он прост в использовании и бесплатен (есть ограничения, но вы можете купить больше вычислительной мощности, если вам это нужно, но для наших целей бесплатный аккаунт идеально подходит).

С помощью бесплатной учетной записи JuliaBox \*\* вы можете создавать блокноты Jupyter, которые запускают код Julia, исполняют их, загружают и экспортируют в виде HTML. Вы можете сохранить свои визуализации как стандартные PNG-файлы и скачать их, чтобы включить в ваши документы.

Кроме того, у JuliaBox уже есть библиотеки, которые мы хотим использовать, поэтому, опять же, установка не требуется.

Записные книжки Jupyter просты в использовании, но, если вы хотите поближе познакомиться с Jupyter или с Юлией, в папке с учебниками будут некоторые полезные руководства, которые вы найдете, когда начнете с JuliaBox.

Если вы хотите поработать с примерами из этой статьи, вы можете загрузить файлы данных и записную книжку Jupyter для использования со своей учетной записью JuliaBox. Я поставлю ссылки в конце статьи.

## Сюжеты

Джулия, как и большинство других языков, использует библиотеки кода для конкретных специализированных целей. Тот, который нас изначально интересует, называется *Сюжеты*, Это дает нам возможность создавать визуализации данных.

Итак, первая часть кода, который нам нужно выполнить, это:

```
using Plots
```

Когда вы вводите это в ячейку кода в своей записной книжке и нажимаете ctrl / enter, чтобы выполнить его, он говорит Джулии загрузить библиотеку, которую мы будем использовать для создания наших визуализаций.

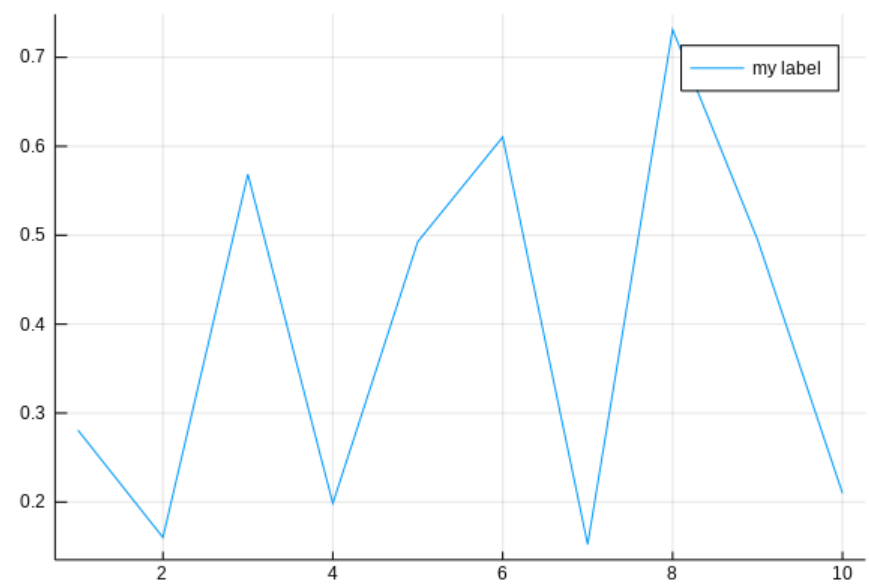
Когда вы запускаете ячейку в блокноте в первый раз, выполнение может занять некоторое время. Это связано с тем, что код Julia компилируется на лету при первом запуске. Последующие прогоны кода выполняются намного быстрее. (Это еще одно преимущество JuliaBox - их вычислительная мощность, вероятно, лучше, чем у вас или у меня, поэтому время компиляции быстрее.)

## Ваш первый график

Я обычно помещаю разные фрагменты кода в новые ячейки ноутбука. Это означает, что мне нужно только запустить нужный мне код, а не весь блокнот. Я предлагаю вам сделать то же самое, поэтому в новой ячейке я набрал следующий код:

```
x = 1:10; y = rand(10); # These are the plotting data
plot(x,y, label="my label")
```

Запустив его, вы получите следующий график:



Впечатляет. Позвольте мне объяснить, что происходит.

```
x = 1:10; y = rand (10); # Это данные построения
```

Этот бит кода создает два бита данных, один называется *Икси* другие *У*, *Икс* дается значение диапазона чисел от 1 до 10, в то время как *У* задается диапазон из 10 псевдослучайных чисел (каждое будет иметь значение от 0 до 1). Итак, здесь у нас есть основа графика: ось *x*, которая находится в диапазоне от 1 до 10, и значения *y* для каждой из точек на оси *x*.

Следующий бит прост.

```
сюжет (x, y, label = "my label")
```

Этот код вызывает функцию для построения графика, и все, что мы делаем, это присваиваем ему значения *x* и *y* - и, в качестве дополнительного, мы также присваиваем ему метку.

Это было легко, но, конечно, мы действительно хотим визуализировать некоторые реальные данные.

У меня есть несколько таблиц, которые я использовал для других статей. Это набор данных о погоде в Лондоне, Великобритания, за последние несколько десятилетий. Я получил его из публичных таблиц, предоставленных Метеорологическим бюро Великобритании.

Данные регистрируют максимальную температуру, минимальную температуру, количество осадков и количество солнечных часов, зарегистрированных в каждом месяце. У меня есть два файла, один полный набор данных, а другой только для 2018 года. Они представлены в формате CSV, например, вы можете импортировать их в электронную таблицу.

Для работы с этими файлами нам нужна другая библиотека, которая позволяет нам читать файлы CSV.

Мы можем видеть библиотеку, на которую ссылаются в следующем фрагменте кода, то есть «используя CSV», а следующая строка фактически считывает данные в переменную *d*,

```
using CSV
d = CSV.read("london2018.csv")
```

В результате выполнения кода мы получили таблицу данных, которая выглядит следующим образом:

	Year	Month	Tmax	Tmin	Rain	Sun
	Int64	Int64	Float64	Float64	Float64	Float64
1	2018	1	9.7	3.8	58.0	46.5
2	2018	2	6.7	0.6	29.0	92.0
3	2018	3	9.8	3.0	81.2	70.3
4	2018	4	15.5	7.9	65.2	113.4
5	2018	5	20.8	9.8	58.4	248.3
6	2018	6	24.2	13.1	0.4	234.5
7	2018	7	28.3	16.4	14.8	272.5
8	2018	8	24.5	14.5	48.2	182.1
9	2018	9	20.9	11.0	29.4	195.0
10	2018	10	16.5	8.5	61.0	137.0
11	2018	11	12.2	5.8	73.8	72.9
12	2018	12	10.7	5.2	60.6	40.3

# Графики из значимых данных

Данные, которые мы скачали, сформированы из таблицы с 6 столбцами: *Год*, *Месяц*, *Tmax* (максимальная температура), *Tmin* (минимальная температура), *дождь* (количество осадков в миллиметрах) и *солнце* (количество часов солнечного сияния).

Это подмножество данных (только для 2018 года), поэтому *Год* столбец имеет одинаковое значение во всех строках.

## Гистограмма

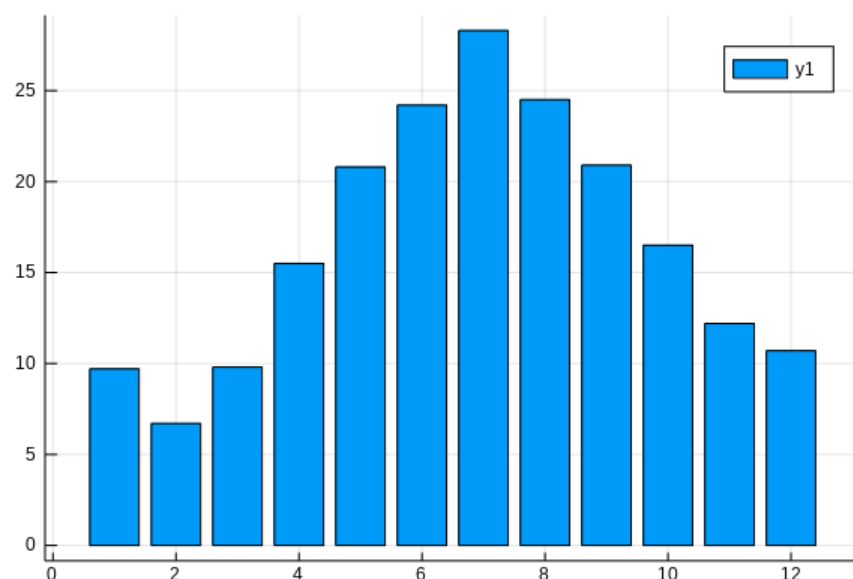
Итак, у нас есть данные за каждый месяц 2018. Если бы мы хотели построить максимальную температуру каждого месяца на гистограмме, мы бы сделали это:

```
bar(d.Month, d.Tmax)
```

*bar* это функция, которая рисует гистограмму (что еще?), и мы предоставляем столбцы для осей X и Y. Мы делаем это, используя имя таблицы данных, а затем имя столбца. Два имени разделены точкой.

Здесь у нас есть столбец *Месяц* как ось x и *Tmax* в качестве оси y - мы показываем максимальную зарегистрированную температуру для каждого из 12 месяцев в таблице.

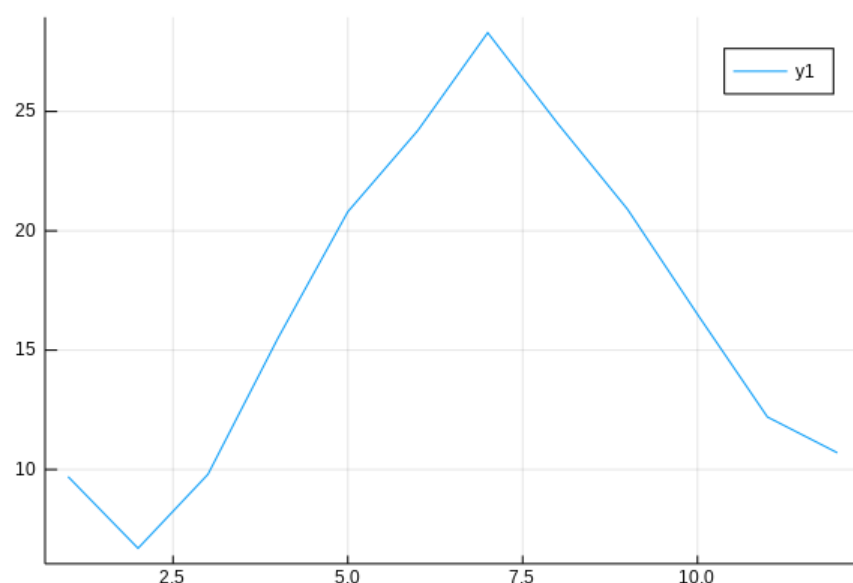
Поместите это в новую ячейку кода и запустите, и вы будете приятно удивлены (я надеюсь), увидев этот график:



## Линейные графики

Если вы хотите создать линейный график, вы делаете то же самое, но используете функцию *plot*

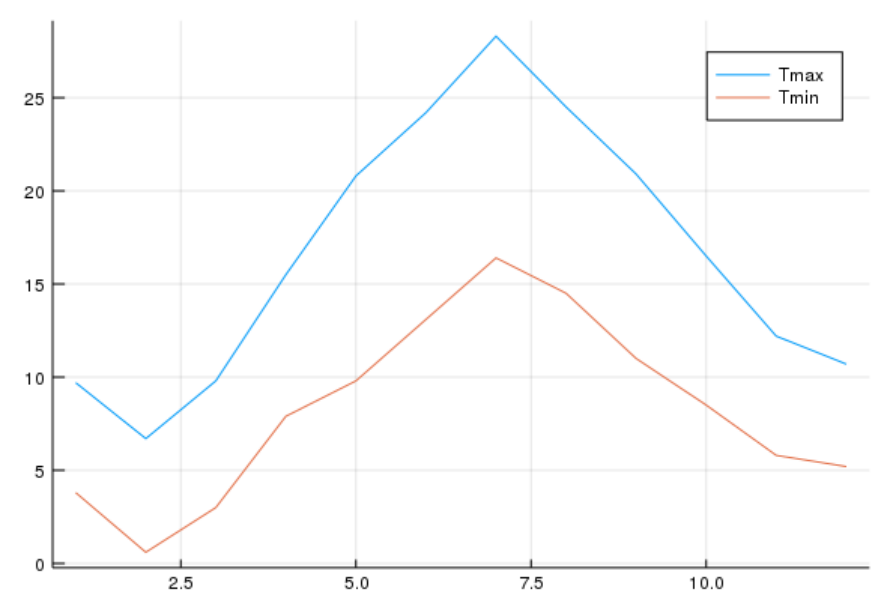
```
plot(d.Month, d.Tmax)
```



И, если вы хотите построить максимальную и минимальную температуры на одном графике, вы можете сделать это:

```
plot(d.Month, [d.Tmax, d.Tmin], label=["Tmax", "Tmin"])
```

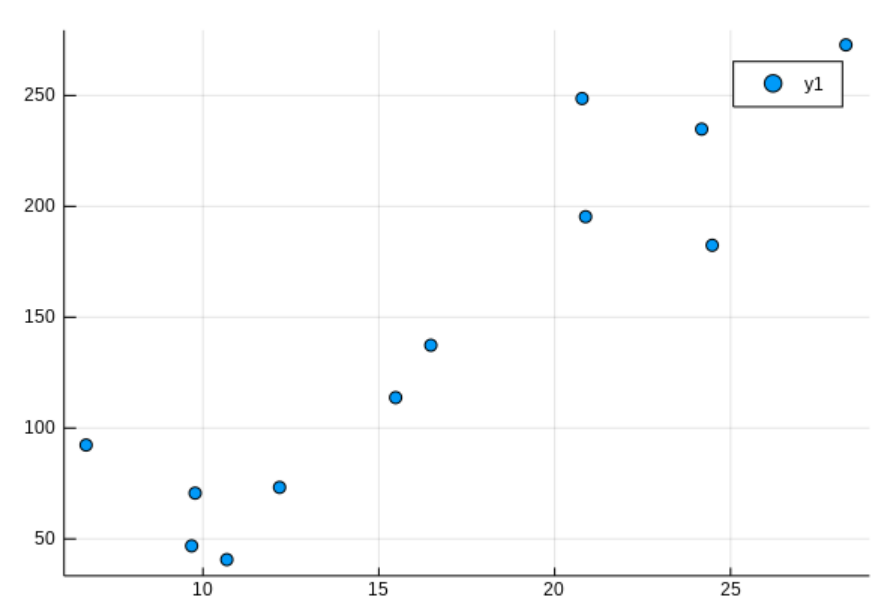
Обратите внимание, что два значения, `d.Tmax` и `d.Tmin`, сгруппированы в квадратные скобки и разделены запятой. Это обозначение для вектора или выделенного размерного массива. Кроме того, мы добавили метки для линий, и они сгруппированы таким же образом. Мы получаем график как это:



## Точечная диаграмма

Или как насчет точечной диаграммы? Точечная диаграмма часто используется, чтобы увидеть, может ли образец быть обнаружен в данных. Здесь мы наносим максимальную температуру на солнце. Как и следовало ожидать, существует закономерность: видимая корреляция - чем больше часов солнечного света, тем выше температура.

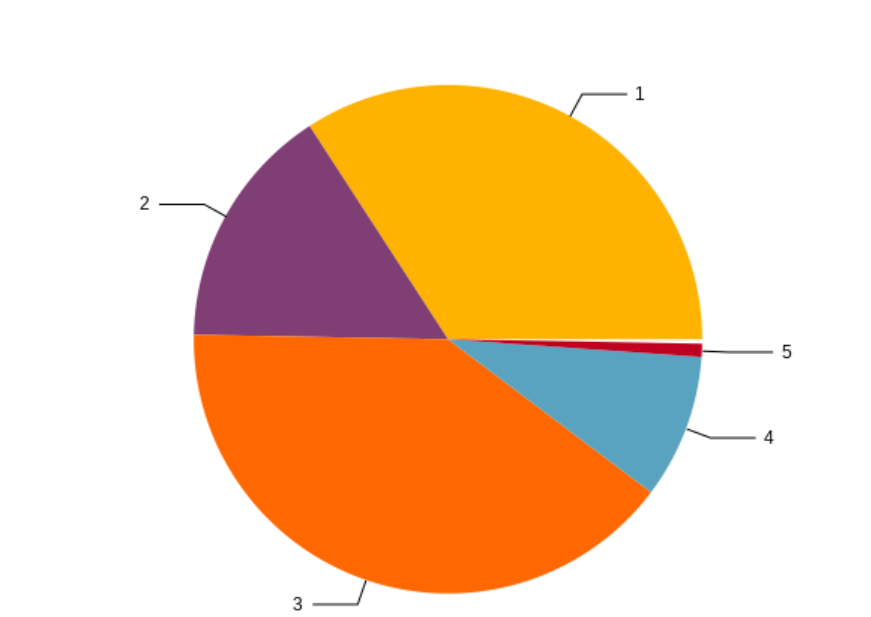
```
scatter(d.Tmax, d.Sun)
```



## Круговая диаграмма

Данные, которые у нас есть, на самом деле не поддаются изображению в виде круговой диаграммы, поэтому мы собираемся снова сгенерировать несколько случайных данных - 5 случайных чисел.

```
x = 1:5; y = rand(5); # These are the plotting data
pie(x,y)
```



# Гистограмма

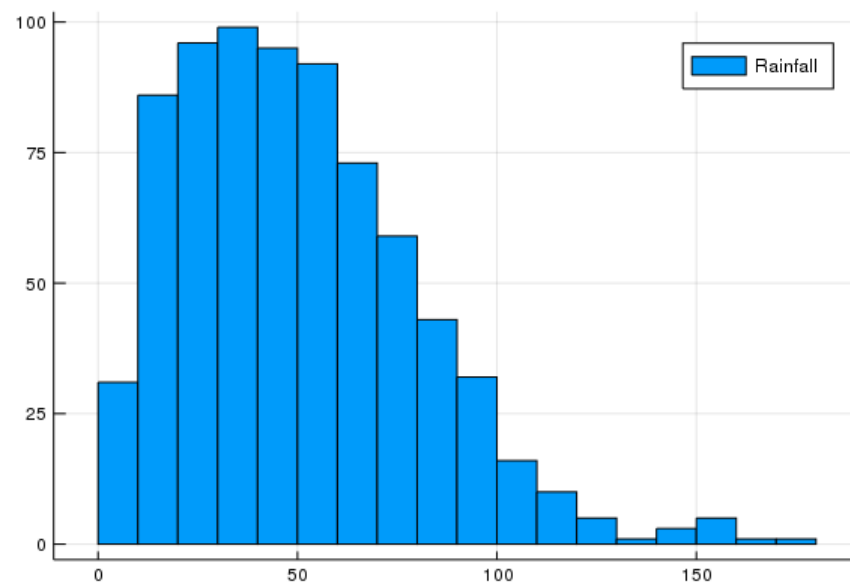
Теперь мы собираемся загрузить немного больше данных:

```
d2 = CSV.read("londonweather.csv")
```

Это похоже на таблицу данных, которую мы использовали, но больше, поскольку она охватывает данные за несколько десятилетий, а не за один год. Это дает нам много данных об осадках, чтобы мы могли видеть распределение уровней осадков, которые происходят в Лондоне в течение длительного периода.

```
histogram(d2.Rain, label="Rainfall")
```

Вот результат.



## Сохранение диаграммы

Все эти диаграммы очень хорошо видны в среде JuliaBox, но чтобы быть полезными, мы должны иметь возможность загружать их, чтобы использовать их в наших документах.

Первый шаг - сохранить график следующим образом:

```
histogram(d2.Rain, label="Rainfall")
savefig("myhistogram.png")
```

Когда этот код запущен, диаграмма не будет отображаться, но будет сохранена с указанным именем файла. И вы можете найти его в списке файлов JuliaBox. Выберите его, нажав на галочку рядом с именем файла, и над списком файлов появится список параметров. Один из вариантов будет «скачать», нажмите на него, и он будет загружен на ваш компьютер.

## Вывод

Я надеюсь, что это было полезно - мы рассмотрели основные графики, доступные в Julia Plots, и я предположил, что использование JuliaBox - это простой способ создания и загрузки графиков. У Джулии есть гораздо больше, чем мы видели в этой короткой статье, и гораздо больше, что вы можете сделать и с помощью Plots, но я надеюсь, что вы узнали, что это введение вызвало у вас аппетит, чтобы узнать больше.

*\*\* Мне грустно говорить, что сегодня (23 октября 2019 года) я узнал, что JuliaBox больше не будет предоставлять бесплатные аккаунты после конца этого месяца. Академические счета от 7 долларов в месяц и неакадемические от 14 долларов.*

## Загрузки


Щелкните правой кнопкой мыши три ссылки ниже, чтобы загрузить файлы, а затем загрузить их на JuliaBox. Затем в вашем списке файлов откройте `plotweatherjulia.ipynb`,

Блокнот находится здесь: [plotweatherjulia.ipynb](#)

и файлы данных здесь: [london2018.csv](#) а также [londonweather.csv](#)

---

Первоначально опубликовано на [CodeEd](#) 2 сентября 2019 г.

 [Оригинальная статья](#)

- [Фреймворки и библиотеки \(большая подборка ссылок для разных языков программирования\)](#)
- [Список бесплатных книг по машинному обучению, доступных для скачивания](#)
- [Список блогов и информационных бюллетеней по науке о данных и машинному обучению](#)
- [Список \(в основном\) бесплатных курсов машинного обучения, доступных в Интернете](#)