Руслан Чинцов,
студент факультета психологии СПбГУ
SPbR #8 (09.09.2017)

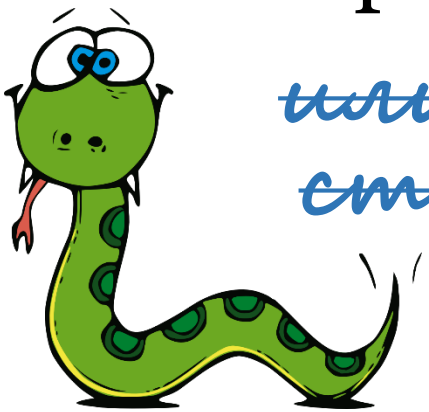
Python в R и R в Python:

гибридный пайплайн от 0 до 1

Руслан Чинцов,
студент факультета психологии СПбГУ
SPbR #8 (09.09.2017)

Python в R
и R в Python:

гибридный пайплайн от 0 до 1



~~*или как усидеть на двух стульях так, чтобы не слиплась*~~

1. Введение

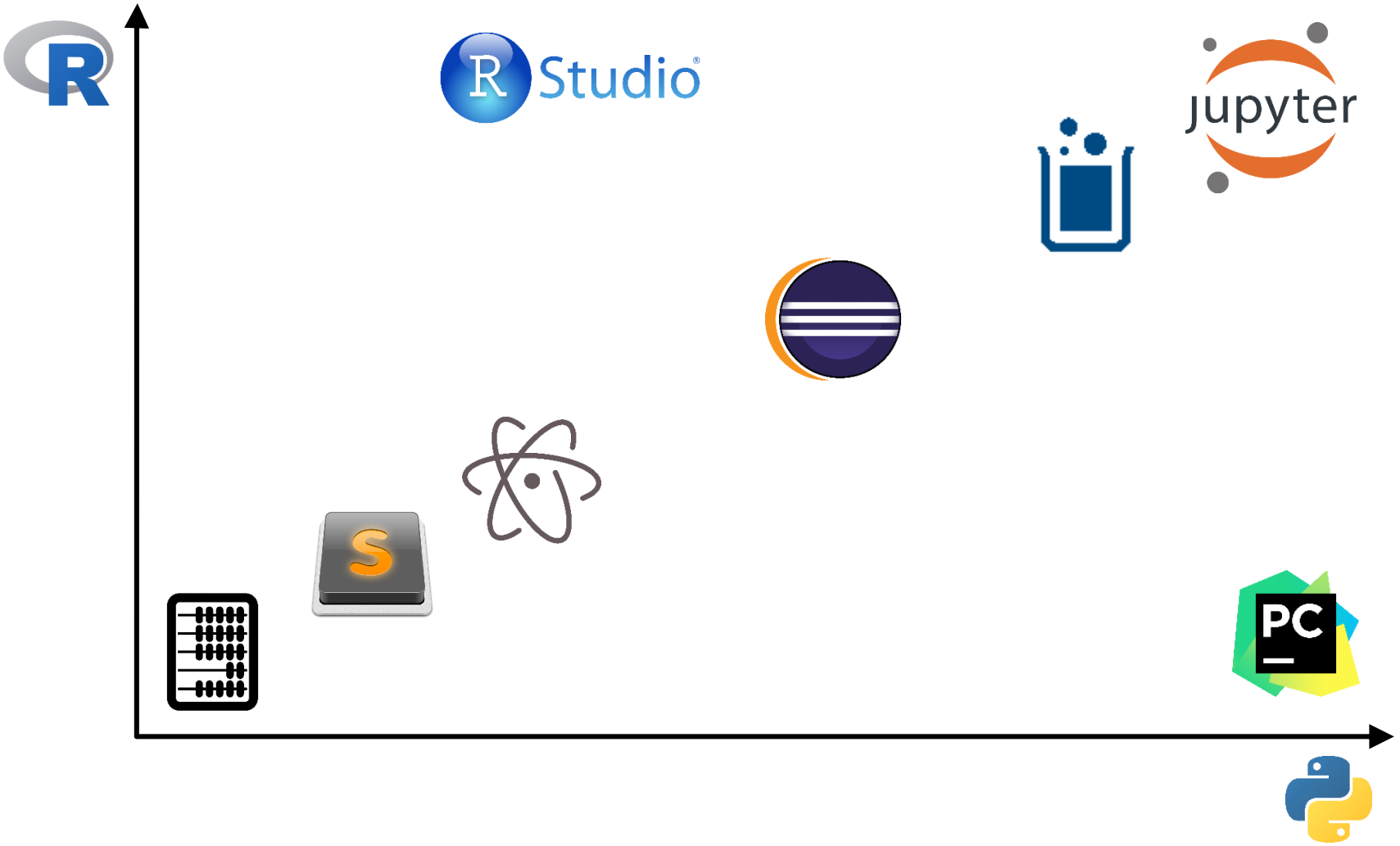
«Оно сокрыто где-то там».

Д. Браун

Зачем?

- Доступность инструментария
- Красивая математика
  - R – ученые для ученых
  - Python – программисты для программистов
  - точность vs простота имплементации
- Research vs production
- Отраслевые стандарты («так исторически сложилось»)
- Это весело

Инструменты: большая часть кода на...

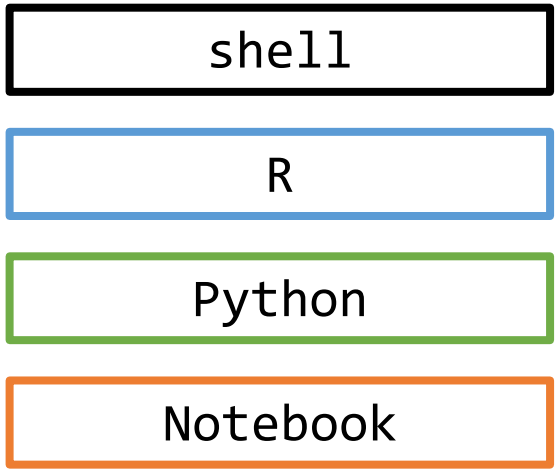


Инструменты

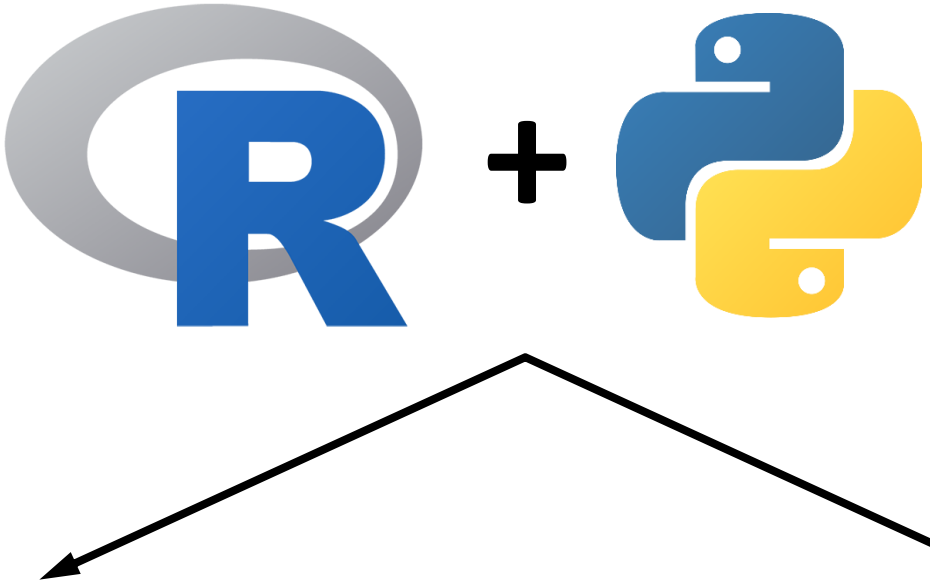
Ссылки:

- Sublime Text
- Atom Editor
- Eclipse extensions:
  - Eclipse StatET (R)
  - PyDev (Python)
- Jupiter Notebook
- Beaker Notebook
- RStudio
- PyCharm

Условные обозначения для примеров кода:



Установка и настройка



Хочу всё и сразу!

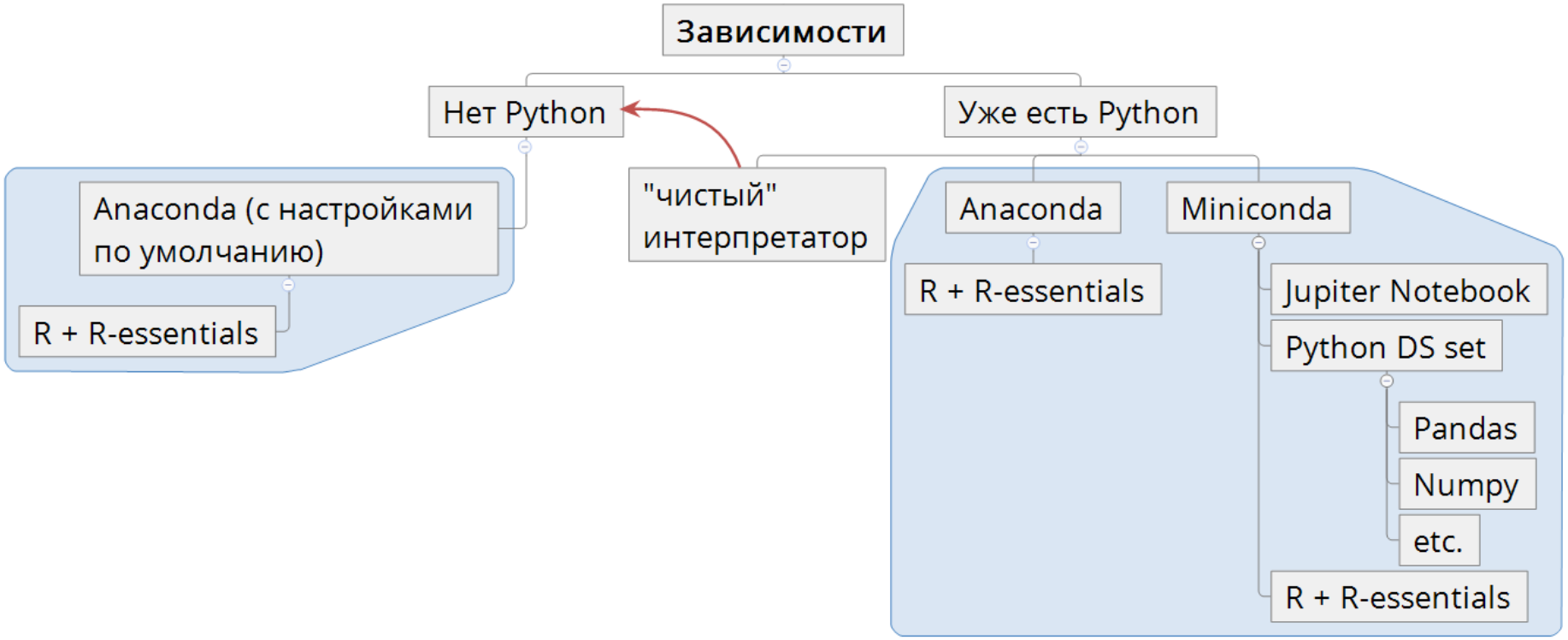
Anaconda + R + R-essentials

Хочу свой бутерброд!

Устанавливаем по отдельности R, Python, IDE и связующие инструменты

Установка и настройка

Anaconda + R + R-essentials



Используя conda:

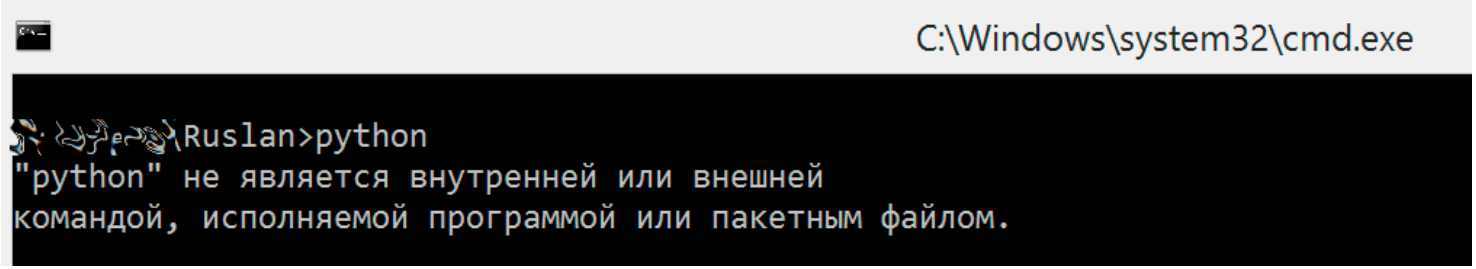
```
> conda install -c r r-essentials rpy2
```

Установка и настройка

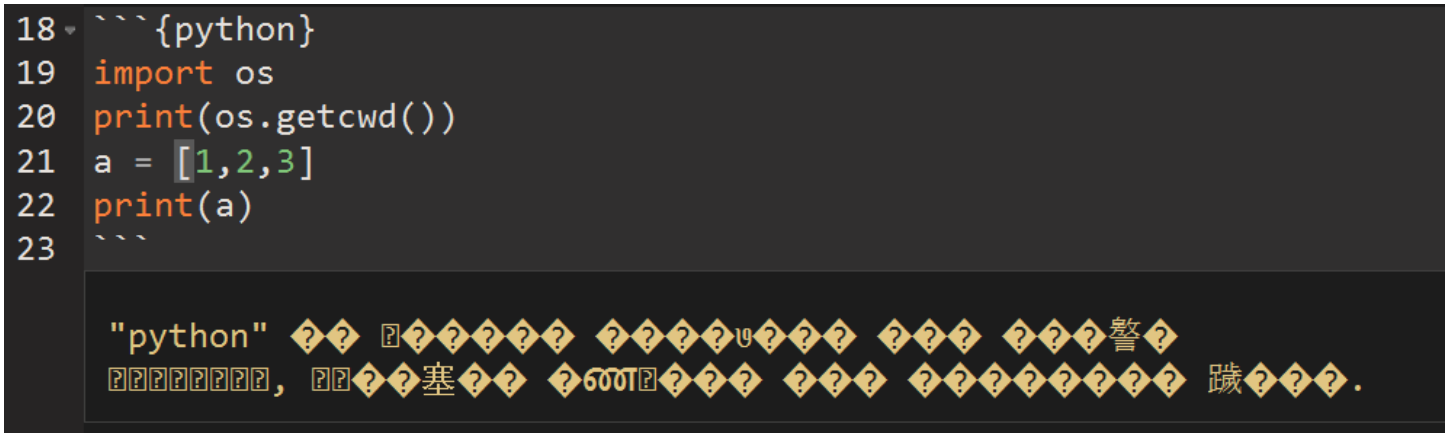
Anaconda: проблемы установки

При установке с настройками по умолчанию Anaconda не прописывает интерпретатор Python в системную переменную Path (Windows), отсюда:

- не работает вызов Python из стандартной командной строки



- невозможно запустить Python в R Notebook и R Markdown



Установка и настройка

Anaconda: решения (одно из)

- Использовать Anaconda Prompt (не решает проблему с R Notebook)
- Переустановить Anaconda с пометкой «Register Anaconda as my default Python [version]» в программе установки
- Добавить каталог с интерпретатором в переменную Path вручную. Для этого:

Windows:

добавить каталог Anaconda в переменную Path
(строку вида **[C]:\Users\[username]\Anaconda3** по инструкции, предварительно заполнив [...] своими значениями)

Linux, OS X:

править конфигурационный файл, подробности см. здесь.

2. Гибридный пайплайн

«Не мышонок, не лягушка, а неведома зверушка».

А.С. Пушкин

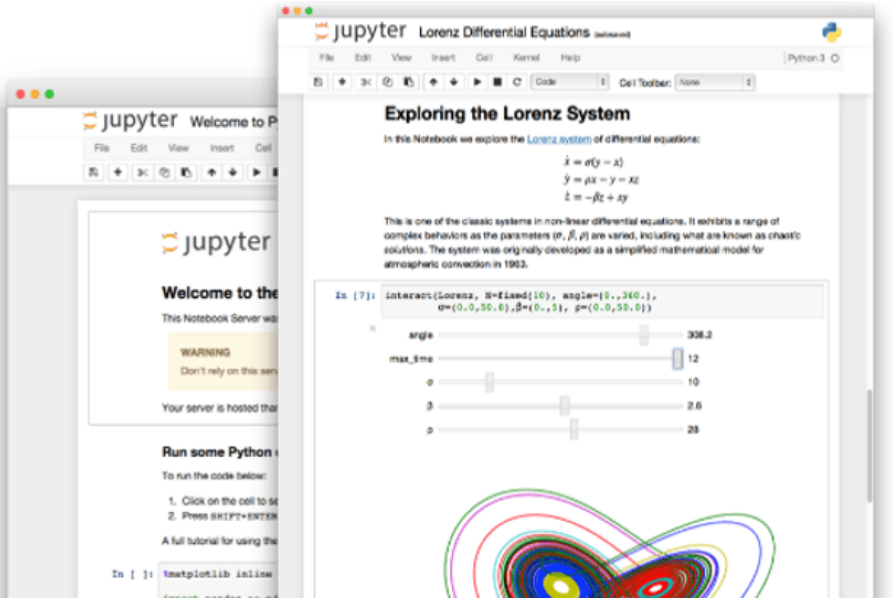
Общие принципы: объем кода

Много кода

Shell Script

~~Мало~~ *умеренно* кода

Notebook



На базе R: Script

Варианты

Консольный вызов внешнего Python script

1. Помещаем код Python в файл с расширением **.py**
2. Выполняем его с помощью системного вызова (команды **system, shell**)
3. Обрабатываем вывод или сохраняем результаты на диск

Работа с внутренними объектами, конвертация типов

Пакеты в порядке убывания полезности:

Reticulate (широкие возможности по Python-деятельности, включая конвертацию)

rPython (выполняет код Python)

XRPython (создание и операции с переменными)

PythonInR (не обновляется)

На базе R: Script

Консольный вызов

2 команды для выполнения (для разных систем)

- **shell** (command) – Windows
- **system** (command) – Unix

```
> shell('python myscript.py')
> system('python myscript.py')
```

*«The most important difference is that on a Unix-alike system launches a shell which then runs command. On Windows the command is run directly – use **shell** for an interface which runs command via a shell (by default the Windows shell cmd.exe, which has many differences from a POSIX shell)».*

***System()** documentation*

На базе R: Script

Консольный вызов

Если данных мало, передаем их через параметры

```
> system('python myscript.py var1 var2')
```

Параметры принимаем с помощью **sys.argv**, см. пример

```
import sys
args = sys.argv
```

Для расширенной работы с параметрами используем optparse

Если данных много, передаем через сохранение на диск, используя

- табличные форматы (CSV, TSV и т.д.)
- простой текст (txt)
- сериализаторы и их форматы (Feather, **RPyBridge**, JSON, HDF, netCDF и др.)

На базе R: Notebook

Чанки **{python}** использовать **не рекомендуется**:

- Сложности с поддержкой пространства имен
- Чанк не в курсе какая сейчас working directory
- Сложности с вводом-выводом переменных

Если очень хочется, **можно прямо из {r}**:

Reticulate (функциональный аналог **rpy2** в Python)

- Поддержка пространства имен Python (ничего не пропадает)
- Возможность импорта Python-объектов в пространство имен R
- Создание Python-объектов и их выполнение (функции)
- Конвертация типов (автоматически и по запросу)
- Доступ к функционалу стандартных пакетов Python
- Выполнение файлов Python Script
- и др.

На базе Python: Script

Варианты

Консольный вызов внешнего R script

1. Помещаем код R в файл с расширением **.R**
2. Выполняем его с помощью **os.system()**
3. Обрабатываем вывод или сохраняем результаты на диск

Работа с внутренними объектами

- Модуль rlike пакета rpy2 - работа с объектами без запуска R интерпретатора
- Работа с **rpy2** с подключением интерпретатора R
- Интерактивная работа с **rpy2**

Подробности см. в документации

На базе Python: Script

Консольный вызов

Если данных мало, передаем их через параметры

```
> import os
> os.system('[C]:\Program Files\R\R-
[version]\bin\Rscript.exe myscript.R var1 var2')
```

Параметры принимаем с помощью **commandArgs()**, см. пример

```
args_in <- commandArgs(trailingOnly = TRUE)
```

Если данных много, сохраняем на диск (см. на слайде про R Script)

Лайфхак:

Чтобы каждый раз не писать в **os.system** путь к **Rscript.exe**, добавить в **Path** путь **[C]:\Program Files\R\R-[version]\bin**
(см. слайд Anaconda: проблемы установки)

На базе Python: Jupiter Notebook

Преимущества:

- Самый популярный инструмент для гибридных пайплайнов
- Постоянное именное пространство R
- Удобный импорт и экспорт переменных
- Автоматическая и эксплицитная конвертации типов (+ много конверторов)
- Возможность писать в одной ячейке на R и на Python
- Возможность создавать ячейки только для R
- Возможность писать код в Jupiter R Notebook (с автодополнением и любимым Alt + “-”)
- Удовольствие от того, что можешь миксовать языки как ~~real nigga~~ настоящий исследователь

Недостатки:

- Нет менеджера переменных
- Необходимость устанавливать дополнительный R в систему
- Неочевидное управление выводом из R («спонтанная» визуализация, недостаток вывода в других случаях)

На базе Python: Jupiter Notebook

Основной синтаксис

«Магические команды» Jupiter Notebook:

`%load_ext rpy2.ipython` — Главный патимейкер, запускает интерактивный режим R.

`%%R` — Указывает, что данная ячейка содержит код только на R, пишется вначале ячейки.
Warning: присваивание “**<-**” не работает, только “**=**“.

`%%R -i X -o Y` — Импортирует в ячейку переменную X, экспортирует Y.

Строчные префиксы:

`%R` — Позволяет внутри Python ячейки выполнить строку на R.

`%Rpush` — Импортирует переменную в окружение R.

`%Rpull` — Экспортирует переменную в окружение Python.

In [1]:
```
# запускаем интерактив
%load_ext rpy2.ipython
```

Ячейка для R

In [3]:
```
%%R
r1 = c(100, 200) # создаем R-объект
print(r1) # вывод из R не работает - баг или фича?
```

Строка R с интерактивным выводом

In [5]:
```
%R r2 = r1 + 10
```

Out[5]:
```
array([ 110.,  210.])
```

Экспортируем объект r3 (его ещё не существует, а мы уже)

In [6]:
```
%%R -o r3
r3 = r2 + 100
```

In [6]:
```
r3 # Бинго! Объект доступен в Python
```

Out[6]:
```
array([ 210.,  310.])
```

In [7]:
```
%R r1 # смотрим содержимое переменной
```

Out[7]:
```
array([ 100.,  200.])
```

In [8]:
```
_7 # смотрим на вывод ячейки #7. Похоже?
```

Out[8]:
```
array([ 100.,  200.])
```

In [9]:
```
type(_7) # Оказывается это сконвертированная переменная!
```

Out[9]:
```
numpy.ndarray
```

In [10]:
```
r1 # но в namespace Python её ещё нет
```

```
---------------------------------------------------------------
NameError                                 Traceback (most
<ipython-input-10-35a853e2b2b1> in <module>()
----> 1 r1

NameError: name 'r1' is not defined
```

In [11]:
```
%Rpull r1 # экспортируем
```

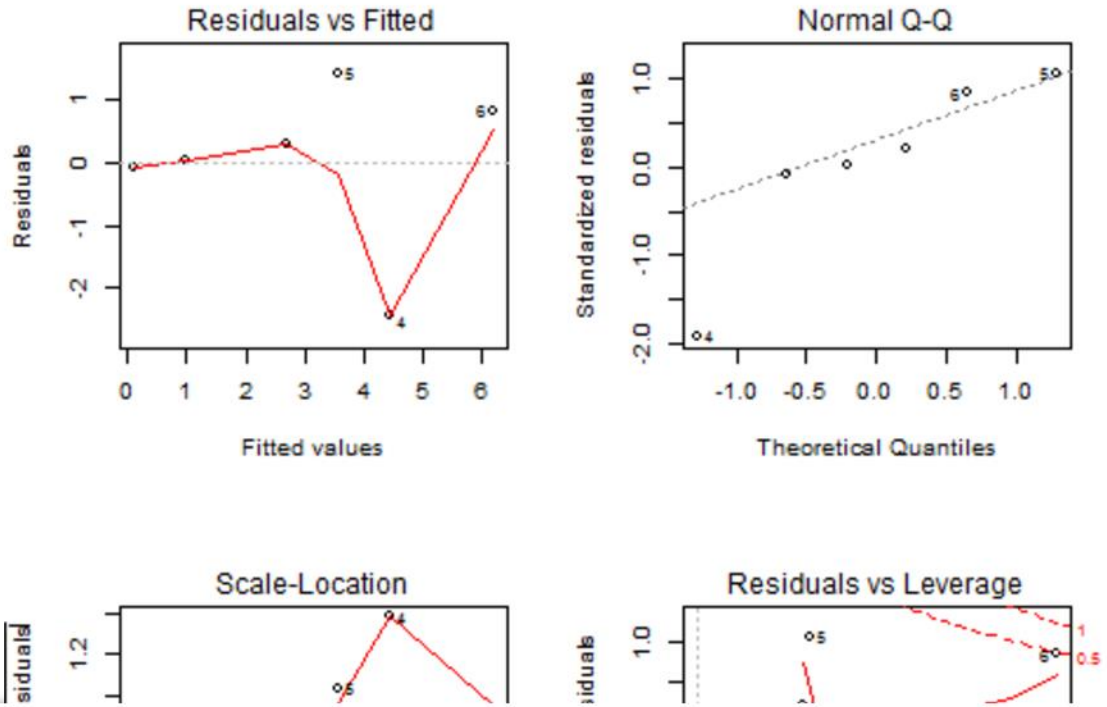
In [12]:
```
r1 # теперь есть
```

Out[12]:
```
array([ 100.,  200.])
```

In [2]:
```
import numpy as np
```

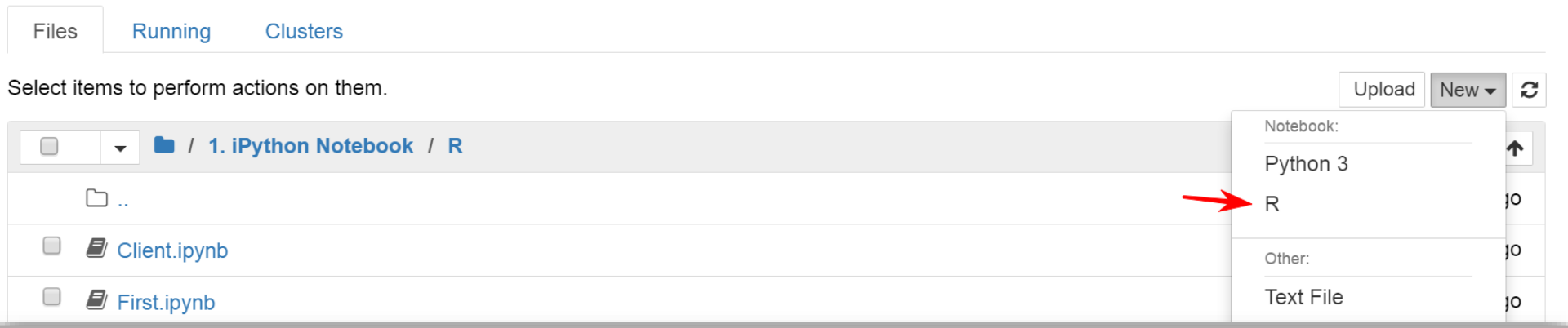
In [3]:
```
x = np.array([1,2,4,6,5,8])
y = np.array([0,1,3,2,5,7])
```

In [4]:
```
%%R -i x,y -o mycoef
xylm = lm(y~x)
mycoef = coef(xylm)
par(mfrow = c(2,2))
plot(xylm)
```



In [5]:
```
print(mycoef)
```

```
[-0.77  0.87]
```



jupyter First Last Checkpoint: 07/05/2017 (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted R

Code

In [1]:
```
library(data.table)
```

In [5]:
```
df <- data.table(data.frame(list(col1 = c(1, 2), col2 = c(3, 4))))
```

In [6]:
```
df
```

col1	col2
1	3
2	4

In [7]:
```
class(df)
```

'data.table' 'data.frame'

In [ ]:
```
type
```

type.convert
typeof

На базе Python: Jupiter Notebook

Больше хороших ссылок:

- Документация rpy2
- Туториал «Python R Collaboration» (Нонна Шахова, ODS SPb)
- Polygot Data Analysis Is Being Pragmatic
- Polyglot Data Analysis Visually Demonstrated With Python And R
- R vs Python: a false dichotomy
- Pipelining R and Python in Notebooks
- RPy2: Combining the Power of R + Python for Data Science

Warning:

Из-за постоянного развития **rpy2**, некоторый код в статьях мог устареть. Смотрите на год публикации и тестируйте. Примеры:

```
%load_ext rmagic
```
→
```
%load_ext rpy2.ipython
```

```
import pandas.rpy.common
```
→
```
from rpy2.robjects import pandas2ri
```

На базе Python: Beaker Notebook

Альтернатива, основанная на Jupiter

Преимущества:

- Поддерживает Python, R, Java Script (в планах Julia, Scala и др.)
- Можно создавать отдельные ячейки для каждого языка
- Имеет собственную кросс-языковую систему транспортировки переменных
- Проект активно развивается

Недостатки:

- Нет менеджера переменных
- Нетривиальный процесс установки
- Альфа-версия

3. Горыныч handmade

«Никогда не откладывай на завтра то,
что можно вообще не делать».

Народная пословица

Основы

Прояснение задачи

Сериализация – процесс перевода какой-либо структуры данных в последовательность битов, используется для передачи объектов по сети и для сохранения их в файлы.

Вопросы:

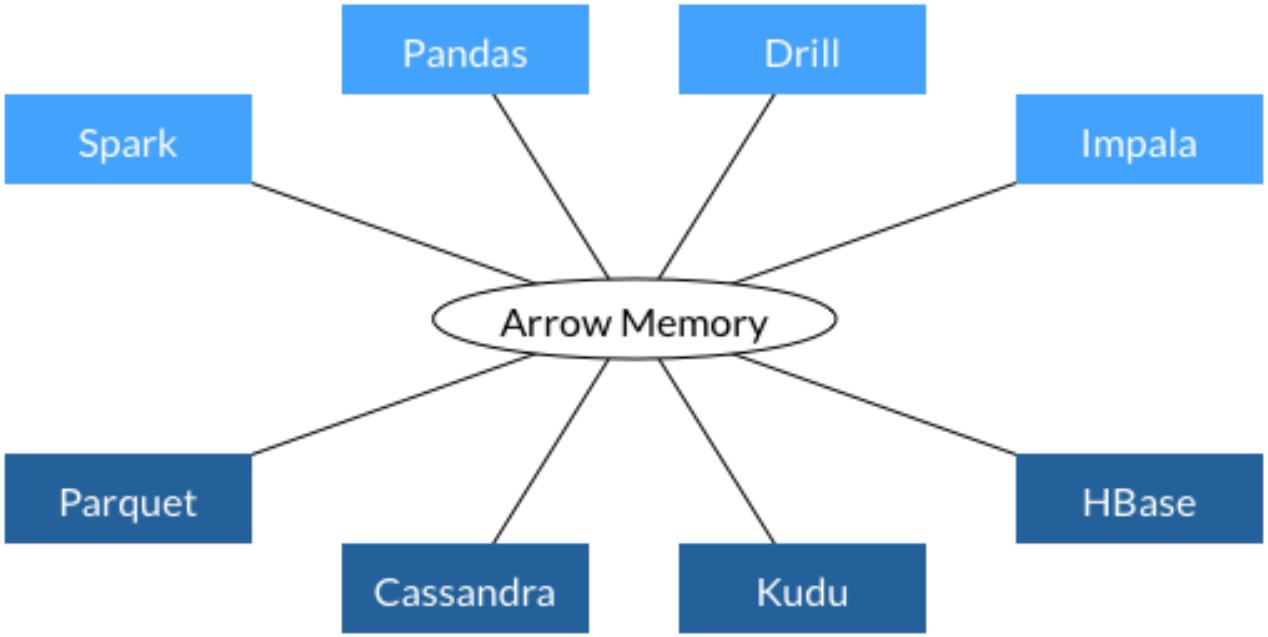
- Что собираемся хранить (тип данных)
- Где будем хранить (ОП, жесткий диск, сеть)
- В каком виде (конечный формат)

Протокол взаимодействия (передачи данных) – набор соглашений интерфейса логического уровня, которые задают единообразный способ передачи сообщений и обработки ошибок.

Примеры: Feather

Инструмент для сериализации табличных данных

- Основан на Apache Arrow Feather Format
- Имеет собственную структуру хранения данных
- Предназначен для передачи данных между платформами



Примеры: Feather

Преимущества

- Быстрый
- Поддерживает большие датасеты

Недостатки

- Только таблицы
- Ограничения в Python:
  - имена столбцов только типа **str**
- Не поддерживает (в Python):
  - Идекс строк
  - Вложенные таблицы
  - Сводные таблицы (pivot)
- Очень чувствителен к типу передаваемых данных (любит юникод и чтобы всё красиво)

```
fe.write_dataframe(df_rpb, 'dat1.fe')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-128-e120956a2134> in <module>()
----> 1 fe.write_dataframe(df_rpb, 'dat1.fe')

                Anaconda3\lib\site-packages\pyarrow\feather.py in write_feather(df, dest)
    110     writer = FeatherWriter(dest)
    111     try:
--> 112         writer.write(df)
    113     except:
    114         # Try to make sure the resource is closed

                Anaconda3\lib\site-packages\pyarrow\feather.py in write(self, df)
     94
     95                 elif inferred_type not in ['unicode', 'string']:
---> 96                     raise ValueError(msg)
     97
     98             if not isinstance(name, six.string_types):

ValueError: cannot serialize column 0 named 0 with dtype mixed-integer
```

Лайфхак: использовать `df.applymap(str)` до сериализации df

Примеры: собственная разработка

RPyBridge

Инструмент для сериализации ~~табличных данных~~ чего угодно

- Основан на JSON
- Не имеет собственной структуры хранения данных (и хорошо)
- Предназначен для передачи данных между R и Python

Преимущества:

- Всеядность кодировок (всё закончится Юникодом)
- Сохранение всех переменных разом, в один файл (**.rpba**)

Недостатки:

- Скорость десериализации на R (bottleneck где-то в **rjson**)
- Ограничение на объем датасета (большие пока не поддерживаются)
- Альфа-версия

Примеры: собственная разработка

Сопоставление типов:

```
    Python          |      R
------------------------------------
 pandas.DataFrame <-> data.frame
                  <-  data.table
------------------------------------
 numpy.matrix     <->  matrix
 numpy.array       ->
------------------------------------
 dict              =   list
 list              =   vector
------------------------------------
 int               =   int
 float             =   float
 bool              =   bool
 string            =   string
```

Фишки:

- varlist = True
- check_values = True

Примеры: собственная разработка

Что не удалось и почему

- Вложенные таблицы, сводные таблицы Python (сложности определения типов и неопределенность их реализации в R)
- Определение имени переменной через функцию в R
- Чтение больших талиц в R (ограничение на длину вектора в памяти)

Планы:

- Исправить порядок сохранения столбцов в Питон
- Реализация загрузки больших таблиц в R
- Попробовать реализовать поддержку pivot tables в Python
- Сервер: интерактивно, независимо, удобно (всё в памяти, распаковка списка переменных, удобный API)
- PyPi, CRAN

```
import feather as fe
import rpybridge as rpb
```

```
dat_fe = [[str(i*j) for j in range(1000)] for i in [k for k in range(974)] + [g for g in 'abcdefghigklmnopqrstuvwxyz']]
```

```
print(len(dat_fe), len(dat_fe[0]))
```

```
1000 1000
```

```
dat_rpb = [[i*j for j in range(1000)] for i in [k for k in range(974)] + [g for g in 'abcdefghigklmnopqrstuvwxyz']]
```

```
print(len(dat_rpb), len(dat_rpb[0]))
```

```
1000 1000
```

```
df_fe = pd.DataFrame(dat_fe)
df_rpb = pd.DataFrame(dat_rpb)

df_fe.columns = [str(i) for i in df_fe.columns]
df_rpb.columns = [str(i) for i in df_rpb.columns]
```

```
%%time
fe.write_dataframe(df_fe, 'df_fe.feather')
```

```
Wall time: 1.25 s
```

```
%%time
rpb.save(df_rpb, 'df_rpb')
```

```
Successfully saved to C:\Users\Ruslan\1. Python Notebooks\R\df_rpb.rpba
Wall time: 1.33 s
```

In [157]:
```
%%time
fe.read_dataframe('iris.fe')
```

```
Wall time: 42 ms
```

Out[157]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
5	5.4	3.9	1.7	0.4
6	4.6	3.4	1.4	0.3
7	5.0	3.4	1.5	0.2
8	4.4	2.9	1.4	0.2
9	4.9	3.1	1.5	0.1

In [158]:
```
%%time
rpb.load('iris.rpba')
```

```
Wall time: 6 ms
```

Out[158]:

	petal length (cm)	petal width (cm)	sepal length (cm)	sepal width (cm)
0	1.4	0.2	5.1	3.5
1	1.4	0.2	4.9	3.0
2	1.3	0.2	4.7	3.2
3	1.5	0.2	4.6	3.1
4	1.4	0.2	5.0	3.6
5	1.7	0.4	5.4	3.9
6	1.4	0.3	4.6	3.4

Спасибо за внимание!

r.chintsov@gmail.com

Slack ODS: @rchi