Scientific Programming in Python

Inteligencia Artificial en los Sistemas de Control Autónomo
Máster Universitario en Ingeniería Industrial

Departamento de Automática

Universidad de Alcalá
ISG

Objectives

1. Motivate the need of efficient matrix representations.
2. Introduce some Python scientific tools.
3. Handle data representations in Python.
4. Basic data visualization with Python.
5. Provide a background for scientific programming.

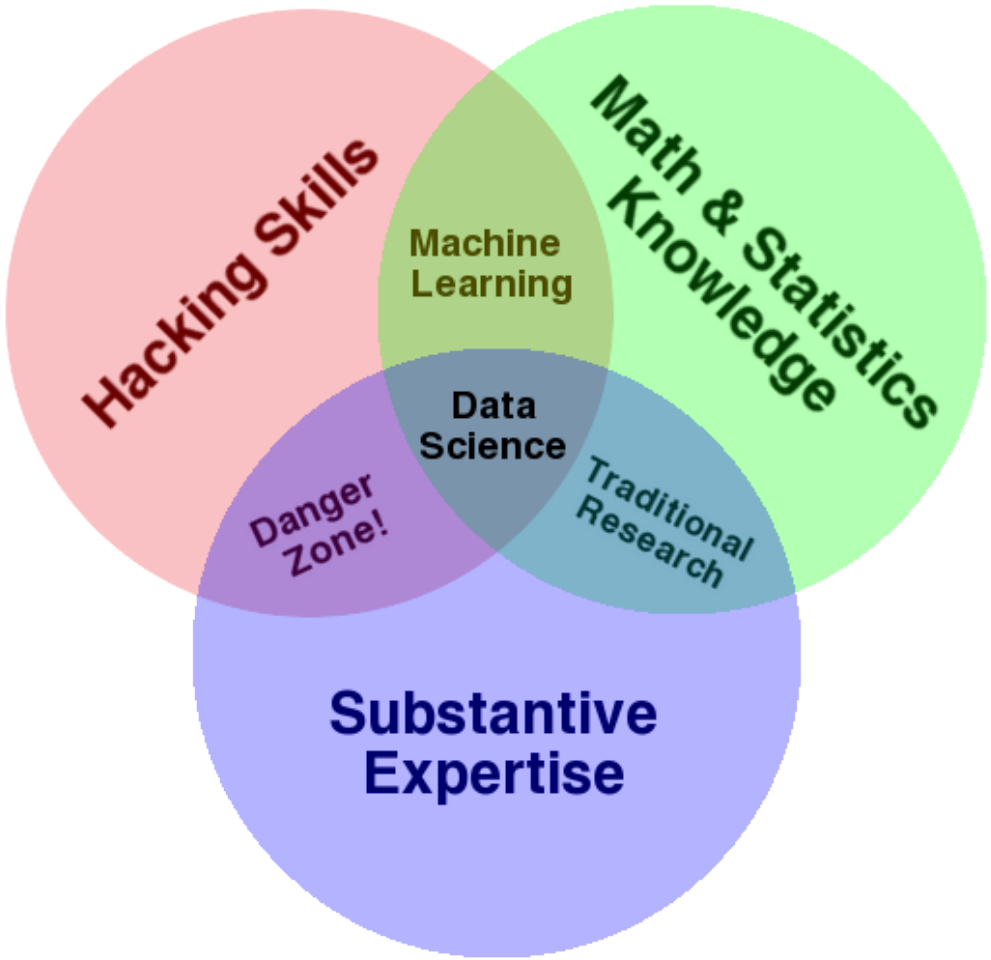
Bibliography

Jake VanderPlas. *Python Data Science Handbook*. Chapters 1, 2, 3 and 4. O'Reilly. (Link).

Table of Contents

Overview

Data Science



Data Science

The data scientist tookit (I)

Data science is about manipulating data

- Need of specialized tools
- Two main languajes: R and Python

Python is a general purpose programming language

- Easy integration
- Huge ecosystem of packages and tools

Need of data-oriented tools

- Features provided by third-party tools

Data Science

The data scientist tookit (II)

Tool	Type	Description
`iPython`	Software	Advaced Python interpreter
`Jupiter`	Software	Python notebooks (Python interpreter)
`Numpy`	Package	Efficient array operations
`Pandas`	Package	Dataframe support
`Matplotlib`	Package	Data visualization
`Seaborn`	Package	Data visualization with dataframes
`Scikit-learn`	Package	AI/ML package for Python

Data Science

Anaconda

All those tools are packaged in `Anaconda`

- Python distribution for Data Science

Anaconda provides `Spyder`

- Python IDE designed for Data Science

Other tools provided by Anaconda

- Conda: Packages management tool
- TensorFlow: Deep Learning
- Many others

Data Science

Python IDEs for Data Science (I)

iPython

iPython = Interactive Python

- Extended funcionality
- Enhanced UI
- External editor

Running iPython:

```
$ ipython
```

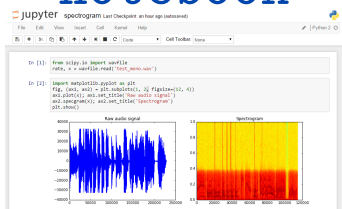
Jupyter

Python notebooks

- Web-based IDE
- Documentation
- Integration with GitHub
- Uses iPython

Running Jupyter:

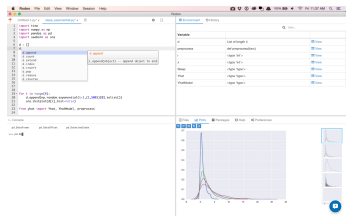
```
$ jupyter notebook
```



Rodeo

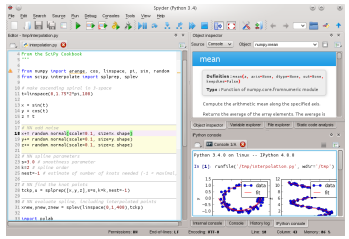
Python version of RStudio

- Good for R developers
- Not included in Anaconda
- Uses iPython



Spyder

Matlab-like IDE



Data Science

Python IDEs for Data Science (II)

Exercises

Write a Python script that shows the multiplication table of the number 5. Write the script using each one of the following environments:

1. iPython + text editor of your choice.
2. Jupiter.
   - Bonus track: Publish the notebook in GitHub.
3. Spyder.
4. Rodeo.

iPython

Basics (I)

In regular Python ...

- most objects come with a docstring attribute
- docstring accesible thorugh `help()`

iPython provides `?', a shortcut to `help()`

- `len?`, `list?`, `list.append?`
- Try to type just `?'

Easy access to source code with `??'

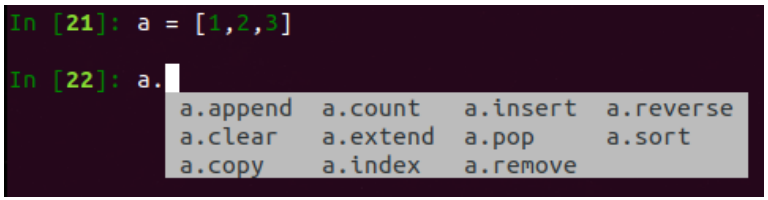
- Does not work with most buildin functions!

iPython

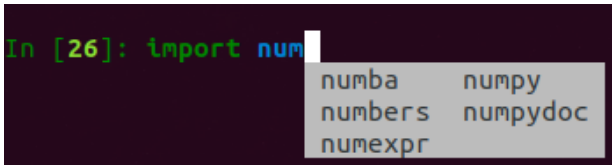
Basics (II)

Press `<tab>` to complete almost everything

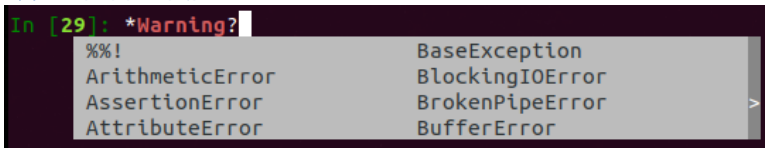
- Object contents



- Packages



- Wildcards



iPython

Basics (III): Keyboard shortcuts

Navigation

Keystroke	Action
`Ctrl-a`	Move cursor to the beginning of the line
`Ctrl-e`	Move cursor to the end of the line
`Ctrl-b`	Move cursor back one character
`Ctrl-f`	Move cursor forward one character

History

Keystroke	Action
`Ctrl-p` (↑)	Previous command
`Ctrl-n` (↓)	Next command
`Ctrl-r`	Reverse-search

Text entry

Keystroke	Action
`Ctrl-d`	Delete next character in line
`Ctrl-k`	Cut text from cursor to end of line
`Ctrl-u`	Cut text from beginning of line to cursor
`Ctrl-y`	Yank (paste) previously cut text

iPython

iPython magic commands

Magic commands: iPython extension of Python syntax

- Not valid in regular Python
- Provides handly features
- Widely used in DS and ML

Two flavours

- % prefix: Line magics - single line
- % % prefix: Cell magics - several lines

Help available

- `%magic`: Magic commands
- `%lsmagic`: List of magic commands

iPython

Pasting code blocks: %paste and %cpaste

Pasting code in Python is troublesome

- %paste: Paste one time
- %%cpaste: Paste several times

```
def donothing(x):
    return x
```

%paste

```
In [20]: %paste
   def donothing(x):
    return x

## -- End pasted text --
```

%cpaste

```
In [25]: %cpaste
Pasting code; enter '--' alone on the line
to stop or use Ctrl-D.
:       def donothing(x):
            return x:
:--
```

iPython

Running external code: %run and %timeit

%run: Execute script

- Many optional arguments
- Checkout %run?

```
In [40]: %run donothing.py

In [41]: donothing(10)
Out[41]: 10
```

%timeit: Computes execution time

- Executes a single line
- Automatic adjustment of runs
- Shows basic statistics

```
In [33]: %timeit [n ** 2 for n in range(200)]
71.6 µs ± 1.84 µs per loop
(mean ± std. dev. of 7 runs, 10000 loops each)

In [34]: %timeit [n ** 2 for n in range(2000)]
753 µs ± 16.2 µs per loop
(mean ± std. dev. of 7 runs, 1000 loops each)
```

%%timeit: Several lines

iPython

Input and output history (I)

iPython stores its history as objects

- `In`: Input commands
  - List storing commands
- `Out`: Commands output
  - Dictionary storing outputs
  - Not all commands have outputs

```
In [1]: import math
In [2]: math.sin(2)
Out[2]: 0.9092974268256817
In [3]: math.cos(2)
Out[3]: -0.4161468365471424
In [4]: Out[2] ** 2 + Out[3] ** 2
Out[4]: 1.0
```

iPython

Input and output history (II)

Fast access to history: Underscore (_)

- Variable containing the last output
- Example: `print(_)`

Double and triple underscores

- Example: `print(__)`
- Example: `print(___)`

Trick: Shortcut to access (_n)

- Out[n] = _n, with n=number
- Example: `print(_2)`

Magic command to show history

- `%history`

Supressing command output (;)

- Example: `4 * 2;`

iPython

iPython shell commands

iPython provides easy interaction with the shell

- Execution of shell commands from iPython
- Use prefix `!'
- Example: `!ls`, `!pwd`

Save shell output in Python variables

- Example: `files = !ls`

Use Python variables in shell

- Example: `!echo {files}`

iPython

Automagic

Problems with some shell commands

```
In [23]: !pwd
/repositorios/pythonCourse
In [24]: !cd ..
In [25]: !pwd
/repositorios/pythonCourse
```

Some magic commands here to help

- `%cd`, `%ls`, `%mkdir`, `%pwd`, ...

Those magics are regularly used ...

- ... so common that % is no longer required (automagic)
- Working with iPython is almost like working with a Unix-like shell

Automagic commands

`cat`, `cp`, `env`, `ls`, `man`, `mkdir`, `more`, `mb`, `pwd`, `rm` and `rmdir`