# Introdução

Thiago Teixeira Santos
<thiago.santos@embrapa.br>

17 de novembro de 2017

## 1 Introdução

Pesquisadores empregam, cotidianamente, técnicas computacionais para simulação, visualização e análise de dados e teste de hipóteses. Assumindo uma visão Popperiana[1] da ciência, a computação auxilia o cientista no desenvolvimento de modelos com boa capacidade de predição, isto é, capazes não só de explicar os dados observados no passado, mas também prever fenômenos futuros (DHAR, 2013).

[1] Karl Popper defendia que o critério principal para se avaliar um modelo era sua capacidade de prever fenômenos futuros, ou seja, seu **poder de predição**. Ver Popper (1963) ou Dhar (2013) para um visão moderna envolvendo computação e ciência dos dados.

O presente material se originou em um tutorial sobre **computação científica com Python** regularmente oferecido pelo autor na Embrapa Informática Agropecuária, na forma de um curso introdutório com um dia de duração. Texto e tutorial são inspirados nas *Python scientific lecture notes*, editadas por Haenel, Gouillart e Varoquaux (2013)[2] e fortemente recomendadas ao leitor interessado no tema. O presente tutorial não tem a pretensão de ser uma referência definitiva, nem prover uma cobertura exaustiva de toda a funcionalidade existente. Seu objetivo é introduzir os recursos encontrados no ambiente Python que podem atender às necessidades de diversos pesquisadores em suas atividades, envolvendo computação.

[2] Alguns exemplos utilizados no presente texto são modificações de exemplos contidos nas *Python scientific lecture notes*, mas obedecem extritamente os termos da licença Creative Commons Atribuição 3.0 (CC BY 3.0) que garante o direito de remix, transformação e criação a partir do material para qualquer fim.

### 1.1 Organização components e materiais

Este texto é organizado da seguinte forma: o restante da introdução (Seção 1) apresenta brevemente o conceito de ambiente de computação científica e algumas soluções existentes, finalizando com informações sobre a instalação do ambiente científico Python. A Seção 2 apresenta rudimentos mínimos da linguagem necessários ao pesquisador. A Seção 3 apresenta o interpretador IPython como ambiente interativo para computação e as vantagens de sua funcionalidade notebook para pesquisa. A Seção 4 introduz a representação utilizada para matrizes de dados (arrays multidimensionais). A Seção 5 apresenta recursos de plotting para visualização de dados. Finalmente, a Seção 6 introduz alguns dos componentes da biblioteca científica SciPy Library e a Seção 7 encerra o texto, apresentando mais referências e novas ferramentas promissoras no ambiente Python.

O conteúdo deste texto, incluindo o código fonte dos exemplos, pode ser obtido na forma de IPython notebooks na web[3] .

[3] Scipy-intro no GitHub.

## 1.2 Computação científica

Com o enorme aumento no volume de dados disponíveis aos cientistas devido à informatização e a novos sensores, o desenvolvimento de modelos preditivos vem sendo chamado de **ciência dos dados** (HEY et al., 2009; DHAR, 2013. O cientista necessita então de um ambiente no qual um vasto conjunto de ferramentas computacionais e estatísticas estejam disponíveis, auxiliando-o no desenvolvimento de resultados replicáveis, ou seja, um ambiente de **computação científica**.

Tal ambiente deveria prover fácil acesso a implementações de diversas ferramentas matemáticas em álgebra linear, estatística, processamento de sinais, visualização científica (plotting de gráficos 2D e 3D), cálculo diferencial e integral, otimização e probabilidade entre muitas outras. Contudo, o uso de tal ambiente deveria implicar em um tempo de codificação curto, permitindo que a exploração de ideias e a avaliação de modelos sejam realizadas rapidamente, além de produzir código e documentação legíveis para o cientista e seus pares, favorecendo a **reprodutibilidade** dos resultados (VANDEWALLE et al., 2009).

- Há um conjunto vasto de ferramentas que deve estar à disposição do pesquisador
- Pesquisadores **não** desejamos re-implementar:
  - *plotting*;
  - a Transformada de Fourier;
  - mínimos quadrados;
  - a solução de um sistema linear $\mathtt{A}\mathbf{x} = \mathbf{b}$.
- Código e comentários deveriam ser legíveis, favorecendo **reprodutibilidade**
- Tempo de desenvolvimento deve ser **curto** pois explorar ideias deve ser **rápido**
- Linguagem e ambiente não devem disputar o foco do pesquisador com o problema científico que está sendo tratado.

## 1.3 Soluções existentes

### 1.3.1 Linguagens compiladas como C/C++ e Fortran

A linguagem Fortran foi criada na década de 1950 como uma alternativa à linguagem *assembly* para computação científica e análise numérica (seu nome é derivado da expressão *Formula Translating System*). Ao longo de quase 60 anos, inúmeras bibliotecas de computação científica foram produzidas para a linguagem, como por exemplo as bibliotecas de álgebra linear, Basic Linear Algebra Subprograms (BLAS) e Linear Algebra Package (LAPACK). Com a criação da linguagem C em 1972 e da linguagem C++ em 1985, parte da comunidade científica passou a utilizá-las, e bibliotecas de computação científica foram desenvolvidas especialmente para elas, como por exemplo a GNU Scientific Library (GSL).

Embora capazes de produzir código extremamente eficiente quanto ao uso da capacidade dos processadores, essas linguagens apresentam tempos de desenvolvimento longos, marcados por ciclos de edição, compilação e execução de código e atividades de gerenciamento de memória e depuração. Mesmo operações simples, como a leitura de dados a partir de um arquivo, podem requerer diversas linhas de código. Outra desvantagem vem do fato de não fornecerem um modo

**interativo** de programação, no qual pequenos trechos de código podem ser executados e seus resultados examinados prontamente pelo pesquisador.

- Vantagens:
  - execução *rápida*;
    - há rotinas científicas em código extremamente eficiente e otimizadas
  - *Exemplo*: BLAS (operações com vetores e matrizes)
- Desvantagens:
  - não são interativas;
  - gerenciamento de memória deve ser realizado pelo programador;
  - tempo de desenvolvimento é **longo**
    - ciclo de *edição-compilação-execução*
    - depuração
    - mesmo operações simples podem requisitar muitas linhas de código

### 1.3.2 MATLAB

MATLAB, nome derivado da expressão *matrix laboratory*, é um ambiente de computação científica que se originou dos esforços de Cleve Moler, professor da Universidade do Novo México, que desejava introduzir as ferramentas de álgebra linear disponíveis nas bibliotecas LAPACK e EISPACK aos seus alunos, mas sem que estes necessitassem aprender Fortran. A MATLAB foi lançada como produto comercial em 1984. Desde então, diversas bibliotecas de computação científica, chamadas de *toolboxes*, passaram a ser fornecidas para MATLAB, em áreas tão diversas como processamento de imagens, bioinformática, econometria e sistemas de controle.

As duas principais desvantagens de MATLAB são:

1. seu elevado custo, o que torna a redistribuição de código entre pesquisadores mais difícil, comprometendo a reprodutibilidade dos experimentos;
2. sua linguagem, bastante limitada frente a linguagens de programação como Fortran, C ou Python.

- Vantagens:
  - rica coleção de rotinas científicas;
  - ambiente de computação interativo;
  - suporte comercial.
- Desvantagens:
  - não é livre ou gratuita e
  - linguagem é pobre e restritiva para usuários mais avançados.

### 1.3.3 Scilab, Octave e R

Scilab e Octave são ambientes de computação científica de código aberto que surgiram como alternativas ao MATLAB. R é um ambiente para computação estatística que surgiu como uma alternativa ao ambiente S, tornando-se uma das plataformas mais populares atualmente em estatística e análise de dados.

- Vantagens:
  - código aberto;
  - bastante avançadas em certos domínios.
    - *Exemplo*: R para estatística
- Desvantagens:
  - menos recursos disponíveis se comparadas ao MATLAB;
  - linguagem pobre e restritiva no caso dos clones MATLAB (Scilab e Octave).

### 1.3.4 Python

Nos últimos anos, um ambiente para computação científica emergiu em torno da linguagem de programação Python. Seu nascimento pode ser traçado a partir do desenvolvimento do módulo Numarray, criado pela comunidade de astrofísica para a representação de matrizes *n*-dimensionais de modo similar a ambientes como MATLAB e Octave. Seu sucessor, o módulo **NumPy**, tornou-se o padrão para a representação de dados na forma de *arrays n*-dimensionais em Python. Originou-se daí uma pilha de software (*software stack*) dedicada à computação científica em Python, a **SciPy Stack**, bem como uma comunidade dedicada de desenvolvedores, reunidos regularmente em conferências SciPy em todo o mundo.

Uma das principais vantagens do ambiente SciPy é o uso da linguagem Python (PEREZ et al., 2011, PERKEL, 2015). Clara e elegante, a linguagem não tem uma curva íngreme de aprendizado como C ou Fortran. Recursos avançados como gerenciamento automático de memória (*garbage collection*), tipagem dinâmica e introspecção facilitam e aceleram o desenvolvimento de código. Expressiva e madura, Python é uma linguagem usada no desenvolvimento de produtos e serviços no mundo todo e conta com uma extensa lista de módulos para programação de propósito geral, além dos componentes de computação científica.

A Tabela 1 compara Python a outras seis linguagens em relação ao desempenho, à interatividade, à gratuidade e à disponibilidade de código. Recentemente, o uso de notebooks IPython (parte do ambiente Python de computação científica) foi destacado pela revista Nature como ferramenta para registro e reprodução de análises de dados (SHEN, 2014). Na primeira linha da Tabela 1, o sinal "+" é utilizado para sinalizar melhor desempenho ("+++" melhor, "+" pior). A classificação de desempenho é baseada no comparativo realizado pela equipe da linguagem Julia.

- Vantagens:
  - bibliotecas científicas muito ricas;
  - linguagem poderosa e elegante;
  - diversas bibliotecas dedicadas a domínios além da computação científica:
    - servidores Web,
    - rede,
    - bancos de dados,
    - persistência de dados
  - código-aberto;
  - uma **gigantesca comunidade de usuários**
- Desvantagens:
  - carece de alguns algoritmos que podem ser encontrados em *toolboxes* especializadas do MATLAB.

Tabela 1. Comparativo entre várias opções para computação científica.

| | Desempenho | Ambiente interativo | Gratuidade | Código aberto |
|---|---|---|---|---|
| C/C++ | +++ | Não | Sim | Sim |
| Fortran | +++ | Não | Sim | Sim |
| MATLAB | + | Sim | Não | Não |
| Octave | + | Sim | Sim | Sim |
| R | + | Sim | Sim | Sim |
| Python | ++ | Sim | Sim | Sim |


Pick up Python, Nature 518:7537

## 1.4 Componentes de um ambiente Python para computação científica

A comunidade SciPy assume que há um **núcleo duro** (*core*) de módulos que constitui um ambiente SciPy de computação científica. Fazem parte de tal núcleo a própria **linguagem Python**, o módulo **NumPy** para *arrays n*-dimensionais, o interpretador de comandos **IPython** e a **biblioteca SciPy**, um conjunto de módulos que provê uma vasta gama de rotinas científicas em estatística,

otimização, interpolação e álgebra linear, entre outras. Também fazem parte do núcleo os módulos **Matplotlib** e **SymPy**. Matplotlib fornece rotinas para a produção de gráficos 2D e 3D para visualização científica. SymPy é um sistema para álgebra computacional que fornece ferramentas para cálculo diferencial e integral, manipulação de polinômios e equações diferenciais, entre outras.

Diversos outros módulos científicos estão disponíveis fora do core. Um dos mais populares é o módulo de aprendizado de máquina **Scikit-learn**. Tendo seu desenvolvimento liderado por pesquisadores ligados ao Institut National de Recherche en Informatique et en Automatique (INRIA), França, Scikit-learn provê ferramentas em regressão, classificação, *clustering*, redução de dimensionalidade e validação cruzada (*cross-validation*). O presente tutorial irá introduzir, ao leitor, a linguagem Python, o ambiente IPython e os módulos NumPy, SciPy e Matplotlib. Por serem módulos fundamentais da computação científica em Python, muito pode ser realizado apenas com tais componentes e a compreensão de sua funcionalidade básica auxilia no trabalho com outros módulos, como a Scikit-learn.

- **IPython** - interpretador de comandos
- **Numpy** - fornece uma implementação eficiente para *arrays n-dimensionais*
- **Biblioteca SciPy** - diversas rotinas cientificas, entre elas:
  - otimização,
  - estatística,
  - interpolação,
  - álgebra linear.
- **Matplotlib** - visualização científica (*plotting*)
- **SymPy** - computação simbólica
- **Scikit-learn** - aprendizado de máquina (*machine learning*)
- **Mayavi** - visualização de dados tridimensionais

## 1.5 Instalação do ambiente de computação científica Python

### 1.5.1 Sistemas Windows, Mac e Linux

Usuários de sistemas operacionais Microsoft Windows, Apple Mac e Linux podem considerar pacotes de instalação como o **Anaconda** da Continuum Analytics ou o **Canopy** da Enthought, Inc. Outra alternativa é a instalação individual de cada um dos pacotes da *SciPy Stack*, de acordo com as instruções atualizadas encontras em SciPy.org.

### 1.5.2 Pacotes Linux

Usuários Linux podem preferir a utilização do sistema gerenciador de pacotes de sua distribuição. Por exemplo, em sistemas Debian/Ubuntu a instação pode ser realizada com o uso do sistema `apt`:

```
$ sudo apt-get install ipython ipython-qtconsole ipython-notebook

$ sudo apt-get install python-scipy python-matplotlib python-matplotlib-data
```

### 1.5.3 Instalação com a ferramenta `pip`

Usuários que já tenham um ambiente Python funcional em seus sistemas podem optar pela ferramenta `pip` para instalar e gerenciar os módulos de computação científica:

```
$ pip install ipython scipy matplotlib
```

## 2 Nosso objetivo

As organizações que obtêm o máximo da **ciência dos dados** são aquelas que posuem uma **infraestrutura** que faz com que a **experimentação** com diversos **problemas**, **técnicas** e **fontes de dados** seja **fácil e eficiente**, e aquelas nas quais há uma **forte colaboração** entre especialistas em **análise de dados** (estatísticos, cientistas da computação, engenheiros) e especialistas no **domínio de aplicação**.

-- adaptado de Pedro Domingos, *A few useful things to know about machine learning*