



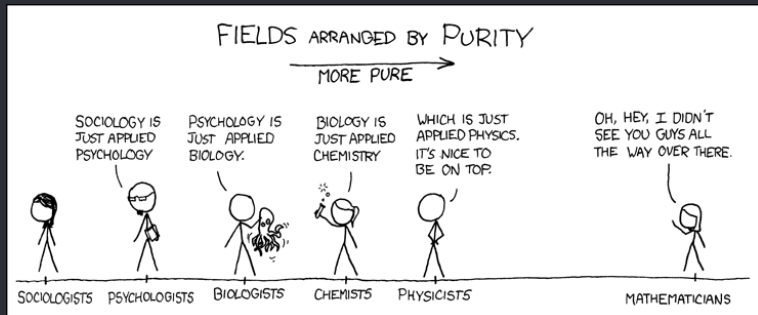
# Do cartão perfurado aos supercomputadores: Fortran moderno na prática (e no Python)

.....

**PythonBrasil 2019**

**Melissa Weber Mendonça**

# Quem sou eu?



# O que é Computação Científica?

Computação Científica não é só Data Science!!!!1!1ONE

```
\includegraphics{angry.gif}
```

# O que é Computação Científica?

“Computação Científica é a coleção de ferramentas, técnicas e teorias necessárias para a resolução computacional de modelos matemáticos de problemas nas Ciências e na Engenharia.” [1]

# O que é Computação Científica?

“Computação Científica é a coleção de ferramentas, técnicas e teorias necessárias para a resolução computacional de modelos matemáticos de problemas nas Ciências e na Engenharia.” [1]

- Previsão/modelagem de sistemas complexos em grande escala;

# O que é Computação Científica?

“Computação Científica é a coleção de ferramentas, técnicas e teorias necessárias para a resolução computacional de modelos matemáticos de problemas nas Ciências e na Engenharia.” [1]

- Previsão/modelagem de sistemas complexos em grande escala;
- Simulação de experimentos caros/perigosos;

# O que é Computação Científica?

“Computação Científica é a coleção de ferramentas, técnicas e teorias necessárias para a resolução computacional de modelos matemáticos de problemas nas Ciências e na Engenharia.” [1]

- Previsão/modelagem de sistemas complexos em grande escala;
- Simulação de experimentos caros/perigosos;
- Soluções interdisciplinares de problemas das Ciências e da Engenharia.

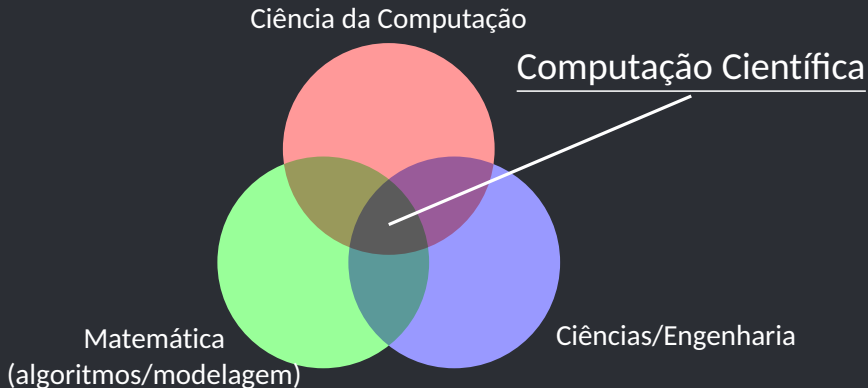
# O que é Computação Científica?

“Computação Científica é a coleção de ferramentas, técnicas e teorias necessárias para a resolução computacional de modelos matemáticos de problemas nas Ciências e na Engenharia.” [1]

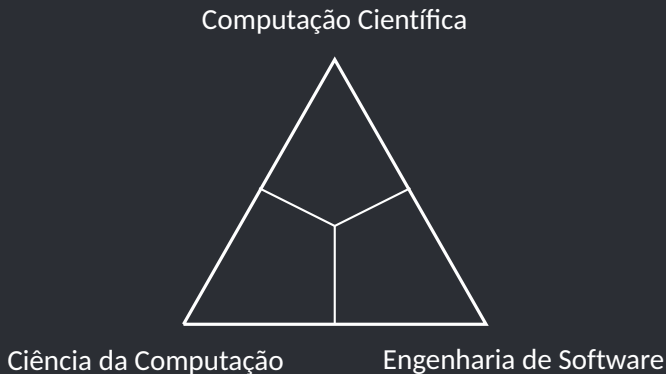
- Previsão/modelagem de sistemas complexos em grande escala;
- Simulação de experimentos caros/perigosos;
- Soluções interdisciplinares de problemas das Ciências e da Engenharia.
- “A Computação Científica é hoje o “terceiro pilar da ciência”, ao lado da análise teórica e dos experimentos para a descoberta científica.” [2]



# O que é Computação Científica?



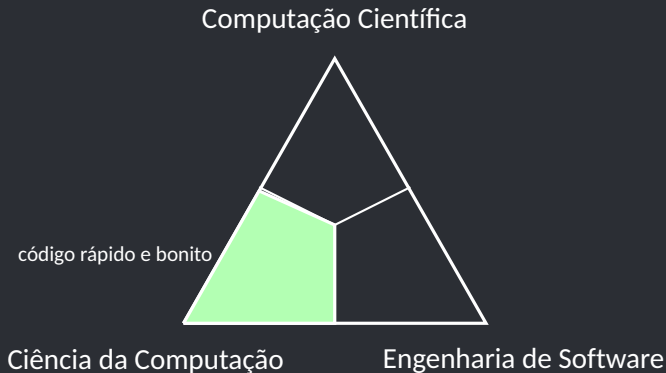
# Mas, na prática...



---

Baseado na imagem [daqui](#)

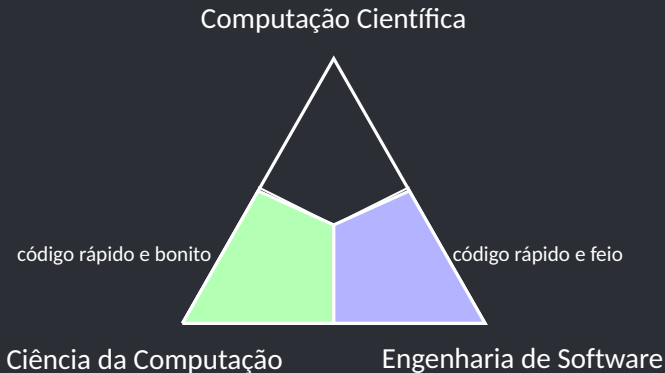
## Mas, na prática...



---

Baseado na imagem [daqui](#)

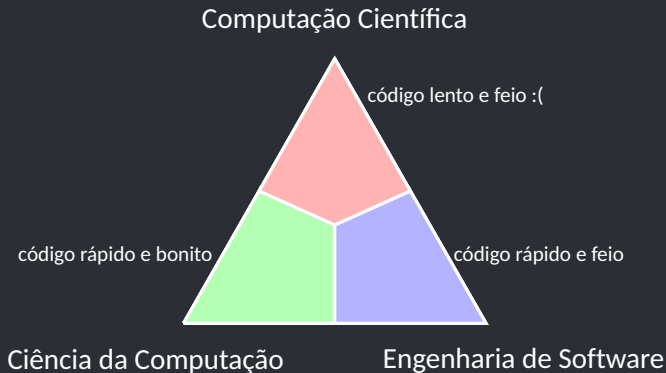
## Mas, na prática...



---

Baseado na imagem [daqui](#)

## Mas, na prática...



---

Baseado na imagem [daqui](#)

# Tipos de Problemas

- Solução Numérica de equações diferenciais: finanças, bioinformática, meteorologia, economia, física
- Estimação de parâmetros/Problemas inversos: estatística, Data Science, processamento de imagens, biomedicina, ajuste de curvas
- Simulações: física, engenharia, geociências, ciência de materiais, dinâmica de fluidos
- Otimização e Álgebra Linear Numérica

# A importância da Álgebra Linear

## Dimensão

Número de variáveis de um problema.

Exemplo:

$$\begin{cases} 2x + 3y = 1 \\ x - 2y = 1 \end{cases}$$

# A importância da Álgebra Linear

## Dimensão

Número de variáveis de um problema.

Exemplo:

$$\begin{cases} 2x + 3y = 1 \\ x - 2y = 1 \end{cases} \Leftrightarrow \begin{pmatrix} 2 & 3 \\ 1 & -2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$



# A importância da Álgebra Linear

## Dimensão

Número de variáveis de um problema.

Exemplo:

$$\begin{cases} 2x + 3y = 1 \\ x - 2y = 1 \end{cases} \Leftrightarrow \begin{pmatrix} 2 & 3 \\ 1 & -2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \Leftrightarrow Ax = b.$$

# A importância da Álgebra Linear

A Álgebra Linear estuda espaços vetoriais e matrizes para resolver problemas do tipo

$$Ax = b$$

$$(A \in \mathbb{R}^{m \times n}, x \in \mathbb{R}^n, b \in \mathbb{R}^m)$$

# A importância da Álgebra Linear

A Álgebra Linear estuda espaços vetoriais e matrizes para resolver problemas do tipo

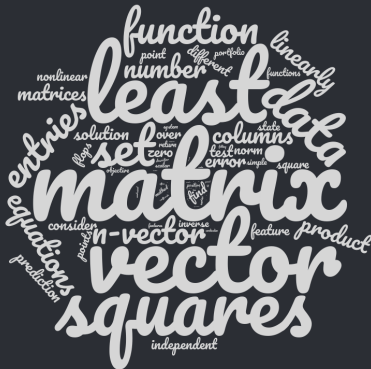
$$Ax = b$$

$$(A \in \mathbb{R}^{m \times n}, x \in \mathbb{R}^n, b \in \mathbb{R}^m)$$

## FUN FACT:

A solução de todos os problemas mencionados previamente dependem da solução de sistemas lineares!

# O que é Álgebra Linear Numérica?



Estudo de algoritmos para realizar cálculos computacionais de álgebra linear, principalmente operações em matrizes e vetores.

## BLAS (1979)

*Basic Linear Algebra Subprograms* (BLAS): especificação de rotinas de baixo nível para operações comuns de álgebra linear.

- ATLAS
- MKL
- OpenBLAS (GotoBLAS)

LAPACK, LINPACK, GNU Octave, Mathematica, MATLAB, NumPy/SciPy, R, e Julia usam a BLAS. (Gráficos, Compressão de áudio/vídeo, Compressão de arquivos, Jogos...)

# Operações Básicas da BLAS

- Level 1 (1979): tipicamente  $O(n)$ . Operações vetoriais em arrays: **produto interno**, norma de matriz ( $axpy$ ):

$$y \leftarrow \alpha x + y$$

- Level 2 (1984-1988): tempo quadrático. Operações matriz-vetor ( $gemv$ ):

$$y \leftarrow \alpha Ax + \beta y$$

- Level 3 (1990): tempo cúbico. Operações matriz-matriz ( $gemm$ ):

$$C \leftarrow \alpha AB + \beta C$$

# LAPACK (antiga LINPACK)

Benchmarks LINPACK: TOP500 (que elenca e detalha os melhores sistemas computacionais não-distribuídos do mundo) é construída medindo a velocidade de resolução de um sistema linear  $n$  por  $n$  denso  $Ax = b$ .



Prof. Jack Dongarra

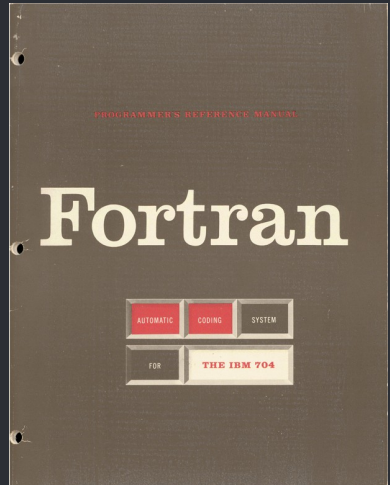
---

Foto da [Universidade do Tennessee, Knoxville](#)

# FORTRAN/Fortran

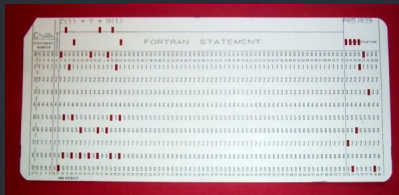
"I don't know what the programming language of the year 2000 will look like, but I know it will be called FORTRAN."

—Tony Hoare, ganhador do Turing Award de 1980, in 1982






# Como assim "Fortran Moderno?!"



- FORTRAN 66/77, Fortran 90/95/2003/2008/2018 - *backwards compatible*
  - Fortran 90: Formato livre, recursão, alocação dinâmica de memória, ponteiros, módulos;
  - Fortran 2003: OO, interoperabilidade com C;
  - Fortran 2008: Submódulos, **Coarrays**, block, tipos derivados;

## Características: FORmula TRANslation

- A estrutura de dados primária é o vetor/matriz
- Sintaxe de slicing nativa, masked arrays, vetorização\*
- É difícil escrever código lento! (exemplos [aqui](#) e [aqui](#))
- Fortran é a única linguagem compilada **concorrente** nativamente
- Legibilidade: feito para computação científica
- Compiladores muito eficientes (alguns "roubando") ;
- Código maduro, bibliotecas, interoperabilidade
- Arquiteturas variadas
- **Exemplos**

# Exemplos

- Regressão Linear
- `simple_pi.f90`

The screenshot shows the GitHub repository page for 'scipy/scipy'. The repository is in the 'master' branch. The 'optimize' directory is selected, showing a list of files: 'README', 'lbfgsb.f', 'lbfgsb.pyf', 'linpack.f', and 'timer.f'. The 'README' file is expanded, showing the following content:

```
From the website for the L-BFGS-B code (from at
http://www.ece.northwestern.edu/~nocedal/lbfgsb.html):

***
L-BFGS-B is a limited-memory quasi-Newton code for bound-constrained
optimization, i.e. for problems where the only constraints are of the
form  $l \leq x \leq u$ .
***

This is a Python wrapper (using F2Py) written by David M. Cooke
<cookedm@physics.mcmaster.ca> and released as version 0.9 on April 9, 2004.
The wrapper was slightly modified by Jonas Paalasmaa for the 3.0 version
in March 2012.

License of L-BFGS-B (Fortran code)
=====

The version included here (in lbfgsb.f) is 3.0 (released April 25, 2011). It was
written by Ciyou Zhu, Richard Byrd, and Jorge Nocedal <nocedal@ece.nwu.edu>. It
carries the following condition for use:
```

# Futuro do Fortran

Ainda muito saudável em computação científica e HPC!

Alguns projetos:

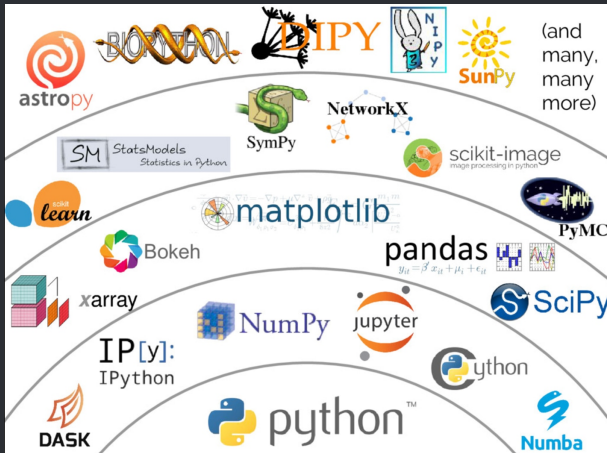
- CHARMM (Dinâmica molecular)
- Code Saturne (Dinâmica de Fluidos Computacional)
- NEMO (Oceanografia)
- QUANTUMESPRESSO (Modelagem de Materiais)
- SPECFEM3D (Propagação de ondas sísmicas)
- WRF (Previsão do Tempo)

# HPC

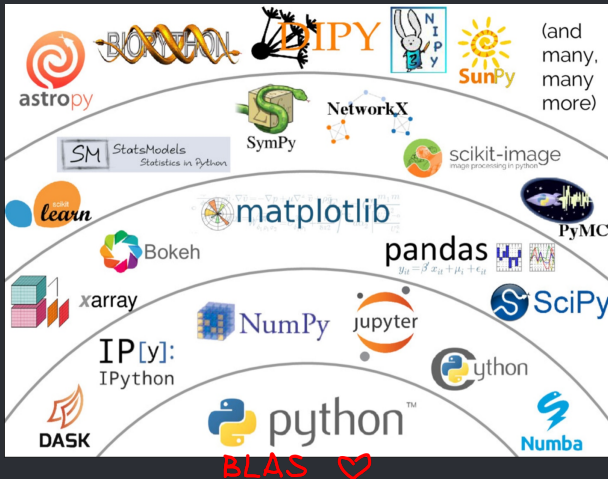
Em uma **pesquisa\*** de usuários de Fortran na 2014 Supercomputing Convention, 100% dos entrevistados disse que acreditavam que ainda estariam usando Fortran em 5 anos.

- C++ e Fortran moderno, com OpenMPI
- CUDA
- Clay Christiansen, em “The Innovators Dillema” diz que tecnologias disruptivas aparecem quando surge um novo mercado/grupo de usuários
- Loop fission/fusion, multithreading, coarrays não são *nice to have*; são **essenciais!**
- Exemplos: <http://www.archer.ac.uk/status/codes/>

# E o Python? (Imagem de [3])



## E o Python? (Imagem de [3])



# Cython/Numba/Pypy

É suficiente usar um JIT, ou acelerar o Python?



# Cython/Numba/Pypy

É suficiente usar um JIT, ou acelerar o Python?

- Perda de modularidade/interoperabilidade (código monolítico)

# Cython/Numba/Pypy

É suficiente usar um JIT, ou acelerar o Python?

- Perda de modularidade/interoperabilidade (código monolítico)
- Reescrever algoritmos especializados do zero?

# Cython/Numba/Pypy

É suficiente usar um JIT, ou acelerar o Python?

- Perda de modularidade/interoperabilidade (código monolítico)
- Reescrever algoritmos especializados do zero?
- Numba
- Cython
- Pypy

# Cython/Numba/Pypy

É suficiente usar um JIT, ou acelerar o Python?

- Perda de modularidade/interoperabilidade (código monolítico)
- Reescrever algoritmos especializados do zero?
- **Numba**
  - Fortran (SciPy) + Numba + Python ainda tem context switching
  - Não faz otimização interprocedural
  - Determinar quando o código deve usar um JIT nem sempre é óbvio
- Cython
- Pypy

# Cython/Numba/Pypy

É suficiente usar um JIT, ou acelerar o Python?

- Perda de modularidade/interoperabilidade (código monolítico)
- Reescrever algoritmos especializados do zero?
- Numba
- Cython
  - Linguagem compilada: compila Python em C.
  - Grande parte da SciPy, pandas e scikit-learn estão escritas em Cython.
  - Pode ser estaticamente tipada, e é aí que o desempenho melhora (loops em C)
  - Pode desligar o GIL
- Pypy

# Cython/Numba/Pypy

É suficiente usar um JIT, ou acelerar o Python?

- Perda de modularidade/interoperabilidade (código monolítico)
- Reescrever algoritmos especializados do zero?
- Numba
- Cython
- Pypy
  - JIT (RPython)
  - GIL: existem workarounds, mas não tão eficientes
  - Grande parte da SciPy, pandas e scikit-learn estão escritas em C(ython).

# f2py

- O f2py cria módulos que podem ser importados no Python.
- Fortran source → signature file → extension module → módulo no Python
- Veja no exemplo!

# Dask

- Dask é uma biblioteca de computação paralela
- Dynamic task scheduling + coleções “Big Data”
- Dataframes, Bags e Arrays (usam RAM + disco)
- Compatível com numpy, scipy, scikit-learn e pandas
- Álgebra linear ainda sendo desenvolvida

Exemplo: (youtube)



## E o C?

“C is pretty much a model of the computer, where you can glue on as much linear algebra as you want, provided you are prepared to pretend that it resembles the computer.

Fortran is pretty much a model of linear algebra, where you can glue on as many computer details as you want, provided you are prepared to pretend that they resemble linear algebra.

(...)

**Whether you prefer to pretend that the computer is made of linear algebra or that the linear algebra is made of computer, depends mostly on where you are coming from to begin with.”**

— Resposta em [7]

## E o Julia?

- Solução para o problema das duas linguagens?

## E o Julia?

- Solução para o problema das duas linguagens?
- Não é um Python mais rápido, e sim um C++ mais produtivo

## E o Julia?

- Solução para o problema das duas linguagens?
- Não é um Python mais rápido, e sim um C++ mais produtivo
- Mais próximo de um AOT do que de um JIT

## E o Julia?

- Solução para o problema das duas linguagens?
- Não é um Python mais rápido, e sim um C++ mais produtivo
- Mais próximo de um AOT do que de um JIT
- Multiple Dispatch + Estabilidade de Tipos => Velocidade + Legibilidade

## E o Julia?

- Solução para o problema das duas linguagens?
- Não é um Python mais rápido, e sim um C++ mais produtivo
- Mais próximo de um AOT do que de um JIT
- Multiple Dispatch + Estabilidade de Tipos => Velocidade + Legibilidade
- Interoperabilidade com Fortran, C, Python (LLVM)

## E o Julia?

- Solução para o problema das duas linguagens?
- Não é um Python mais rápido, e sim um C++ mais produtivo
- Mais próximo de um AOT do que de um JIT
- Multiple Dispatch + Estabilidade de Tipos => Velocidade + Legibilidade
- Interoperabilidade com Fortran, C, Python (LLVM)
- *A maior parte das pessoas não consegue escrever “Fortran ótimo”. Cientistas não são engenheiros de software.*

## E o Julia?

- Solução para o problema das duas linguagens?
- Não é um Python mais rápido, e sim um C++ mais produtivo
- Mais próximo de um AOT do que de um JIT
- Multiple Dispatch + Estabilidade de Tipos => Velocidade + Legibilidade
- Interoperabilidade com Fortran, C, Python (LLVM)
- *A maior parte das pessoas não consegue escrever “Fortran ótimo”. Cientistas não são engenheiros de software.*

Para scripts pequenos o benefício pode não ser óbvio; Para desenvolvedores de pacotes de computação científica, pode ser revolucionário!



# Desafios Técnicos

# Desafios Técnicos

- Obter o desempenho dos benchmarks nem sempre é fácil!

# Desafios Técnicos

- Obter o desempenho dos benchmarks nem sempre é fácil!
- “Better software, better research”: será que a velocidade é sempre o mais importante?

# Desafios Técnicos

- Obter o desempenho dos benchmarks nem sempre é fácil!
- “Better software, better research”: será que a velocidade é sempre o mais importante?
- Poderio computacional não é tudo: algoritmos podem ser mais importantes que a implementação

## Desafios menos técnicos

- Padrões e linguagens novas nem sempre são adotados rapidamente

## Desafios menos técnicos

- Padrões e linguagens novas nem sempre são adotados rapidamente
- *Research Software Engineers*

## Desafios menos técnicos

- Padrões e linguagens novas nem sempre são adotados rapidamente
- *Research Software Engineers*
- Desempenho numérico em grande escala é um problema difícil e particular

## Desafios menos técnicos

- Padrões e linguagens novas nem sempre são adotados rapidamente
- *Research Software Engineers*
- Desempenho numérico em grande escala é um problema difícil e particular
- É difícil convencer pessoas a trabalharem em projetos open source porque o desenvolvimento de software não é visto como um produto de pesquisa válido para acadêmicos. “Toda grande biblioteca matemática open source é construída sobre as cinzas da carreira acadêmica de alguém”.



# Obrigada!

twitter/github: melissawm

repo: [github.com/melissawm/pybr2019](https://github.com/melissawm/pybr2019)

# Bibliografia

- [1] Gene H. Golub and James Ortega. *Scientific Computing and Differential Equations — An Introduction to Numerical Methods*. Academic Press, 1992.
- [2] *What is Scientific Computing?* [Technische Universität Kaiserslautern](#)
- [3] *Python Scientific Computing* [DataCamp](#)
- [4] *Scientific Computing's Future: Can any coding language top a 1950's behemoth?* [Ars Technica](#)
- [5] *Why Numba and Cython are not substitutes for Julia* [Stochastic Lifestyle Blog](#)
- [6] *Formula Translation in Blitz++, NumPy and Modern Fortran: A Case Study of the Language Choice Tradeoffs*
- [7] *For scientific computing, what are the advantages and disadvantages of using C as compared to using FORTRAN 90?*

# Fortran Compilers

- GFortran
- Flang: LLVM
- Intel ifort: SIMD vetorização e threading, OpenMP
- Lahey/Fujitsu
- NAG
- IBM XL Fortran: SIMD, OpenMP, Arquitetura Power9
- Arm Fortran Compiler

► back

# Exemplo da Thais Viana

Correlação de Pearson:

$$\frac{\text{cov}(x, y)}{\sqrt{\text{var}(x) \cdot \text{var}(y)}}$$

► back

## Exemplo da Thais Viana

Correlação de Pearson:

$$\frac{\text{cov}(x, y)}{\sqrt{\text{var}(x) \cdot \text{var}(y)}} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}}$$

► back

## Exemplo da Thais Viana

Correlação de Pearson:

$$\begin{aligned}\frac{\text{cov}(x, y)}{\sqrt{\text{var}(x) \cdot \text{var}(y)}} &= \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}} \\ &= \frac{\langle x - \bar{x}, y - \bar{y} \rangle}{\|x - \bar{x}\| \|y - \bar{y}\|}.\end{aligned}$$

► back