BOAS PRÁTICAS DE PROGRAMAÇÃO E REPRODUTIBILIDADE DE SCRIPTS PYTHON

BRUNO IOCHINS GRISCI

V WORKSHOP DE PYTHON
PARA DADOS BIOLÓGICOS

29 DE SETEMBRO DE 2022

Quem sou eu?

Formado em Ciência da Computação

Doutorando em Computação - PPGC / INF / UFRGS

Structural Bioinformatics and Computational Biology Lab

Professor substituto - INF / UFRGS

Bioinformática, aprendizado de máquina, computação evolutiva, interpretabilidade, dados ômicos.

Motivação

Computational chemistry faces a coding crisis

BY JAMIE DURRANI | 1 JULY 2020

SOURCE: © ROYAL SOCIETY

In October last year, a team of natural product chemists discovered a glitch in a widely used piece of NMR software. Buried deep inside the code was a simple file sorting issue, which on certain operating systems led to incorrect values being predicted for chemical shifts. The finding cast uncertainty over results published in more than 150 scientific papers over a five year period.

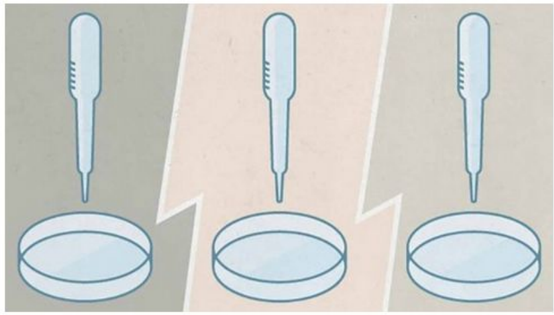
nature > special

SPECIAL | 18 OCTOBER 2018

Challenges in irreproducible research

Science moves forward by corroboration – when researchers verify others' results. Science advances faster when people waste less time pursuing false leads. No research paper can ever be considered to be the final word, but there are too many that do not stand up to further study.

There is growing alarm about results that cannot be reproduced. Explanations include increased levels of scrutiny, complexity of experiments and statistics, and pressures on researchers. Journals, scientists, institutions and funders all have a part in tackling reproducibility. Nature has taken substantive steps to improve the transparency and robustness in what we publish, and to promote awareness within the scientific community. We hope that the articles contained in this collection will help. show less



Artificial intelligence / Machine learning

AI is wrestling with a replication crisis

Tech giants dominate research but the line between real breakthrough and product showcase can be fuzzy. Some scientists have had enough.

by **Will Douglas Heaven**
November 12, 2020

Last month Nature published a damning response written by 31 scientists to a study from Google Health that had appeared in the journal earlier this year. Google was describing successful trials of an AI that looked for signs of breast cancer in medical images. But according to its critics, the Google team provided so little information about its code and how it was tested that the study amounted to nothing more than a promotion of proprietary tech.

https://www.nature.com/collections/prbfkwmwvz/

https://www.technologyreview.com/2020/11/12/1011944/artificial-intelligence-replication-crisis-science-big-tech-google-deepmind-facebook-openai/

https://www.chemistryworld.com/news/chemistrys-reproducibility-crisis-that-youve-probably-never-heard-of/4011693.article

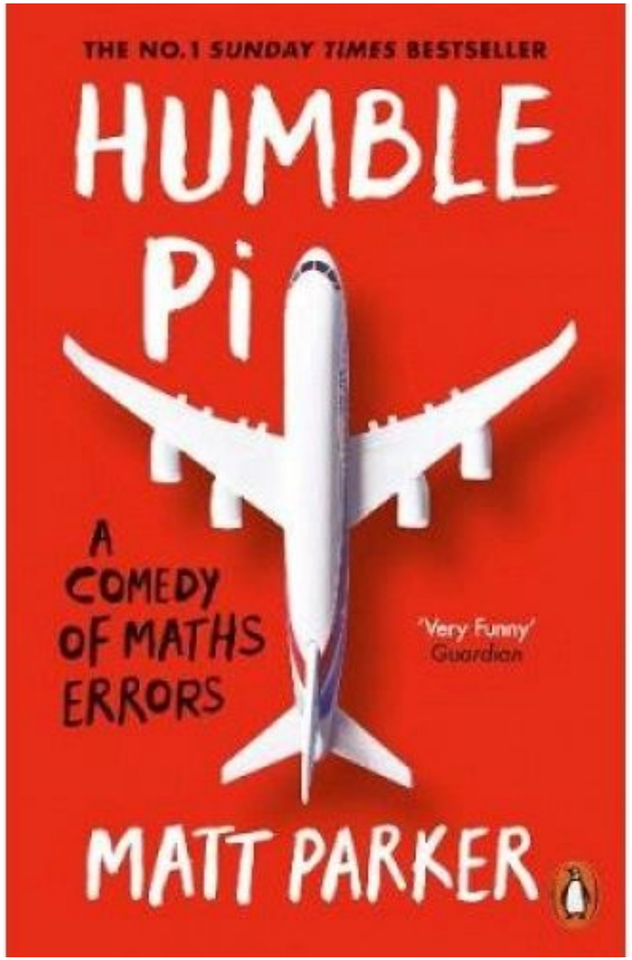
Motivação

Erros de código e más práticas não são apenas uma inconveniência.

Eles podem custar vidas também,

Therac-25: máquina de radioterapia operada por computador.

Entre 1985 e 1987, pelo menos cinco pacientes morreram por overdose de radiação devido a erros no software.



Motivação

Devido a sua periculosidade, o sistema da Therac-25 só ligava a máquina após ela passar numa série de validações.

Class3 += 1

Class3 = setup()

if Class3 == 0:

    ok

else:

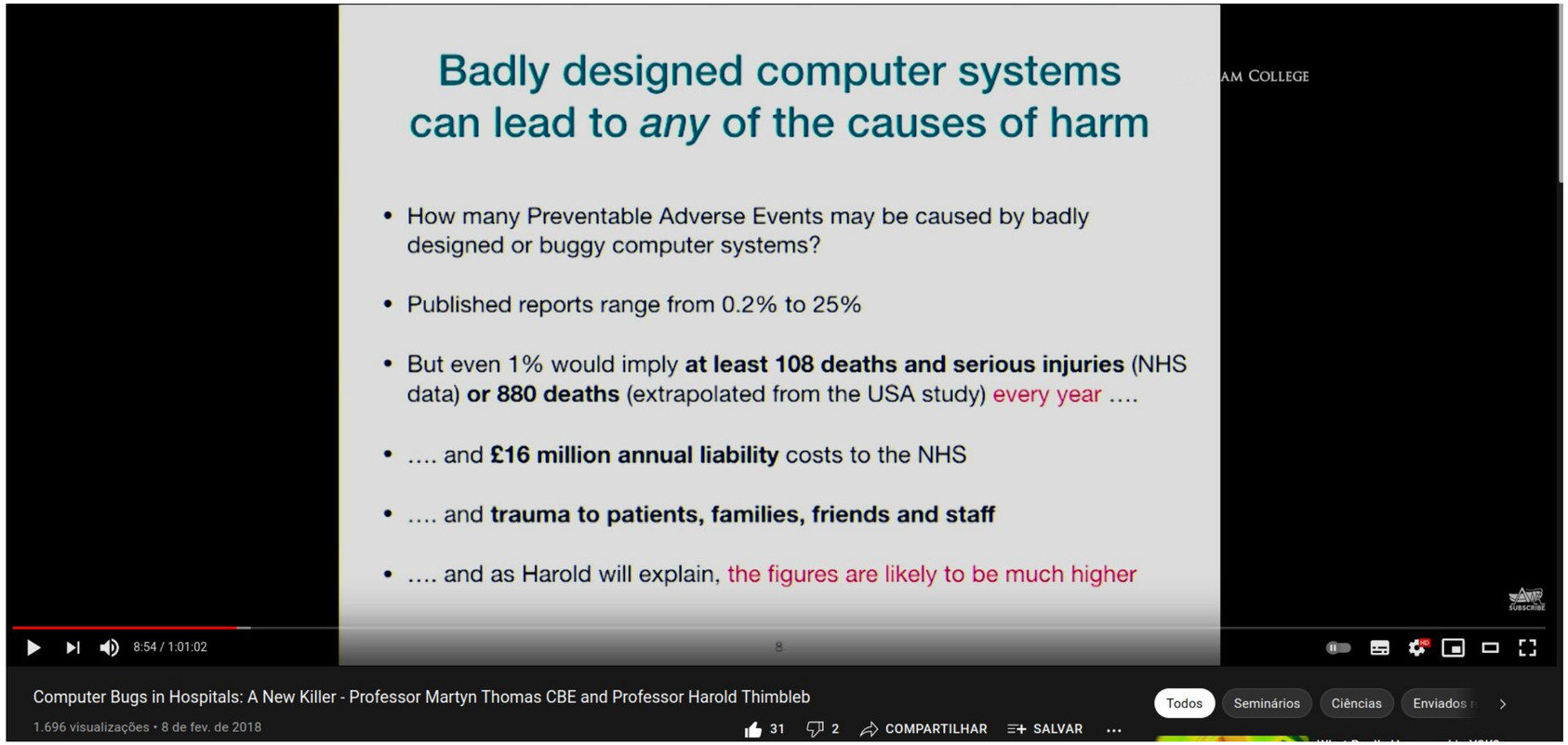
    stop

Class3 era uma variável 8-bits, a cada 256 procedimentos sem ser reiniciada, um overflow ocorria e Class3 == 0 sem passar pelas validações.

Além disso, as mensagens de erro não eram claras:

Malfunction 54, que na documentação levava a “dose input 2 error”

Motivação



Professor Martyn Thomas CBE

Professor Harold Thimbleby

https://www.gresham.ac.uk/lectures-and-events/computer-bugs-in-hospitals-a-new-killer

Objetivos

- Legibilidade
- Corretude
- Usabilidade
- Segurança
- Manutenção
- Reprodutibilidade
- Compartilhamento

Error: usar Python 2

“Python 3 é mais requisitado e inclui um sistema de tipagem. Python 2 é ultrapassado e usa uma sintaxe antiga para a função print. Enquanto Python 2 ainda é usado para gestão da configuração em DevOps, **Python 3 é o novo padrão vigente**.

O que acontece agora?

A partir de primeiro de janeiro, 2020, nenhum relatório de bugs, consertos ou mudanças será feito para Python 2, e **Python 2 não será mais apoiado**.”

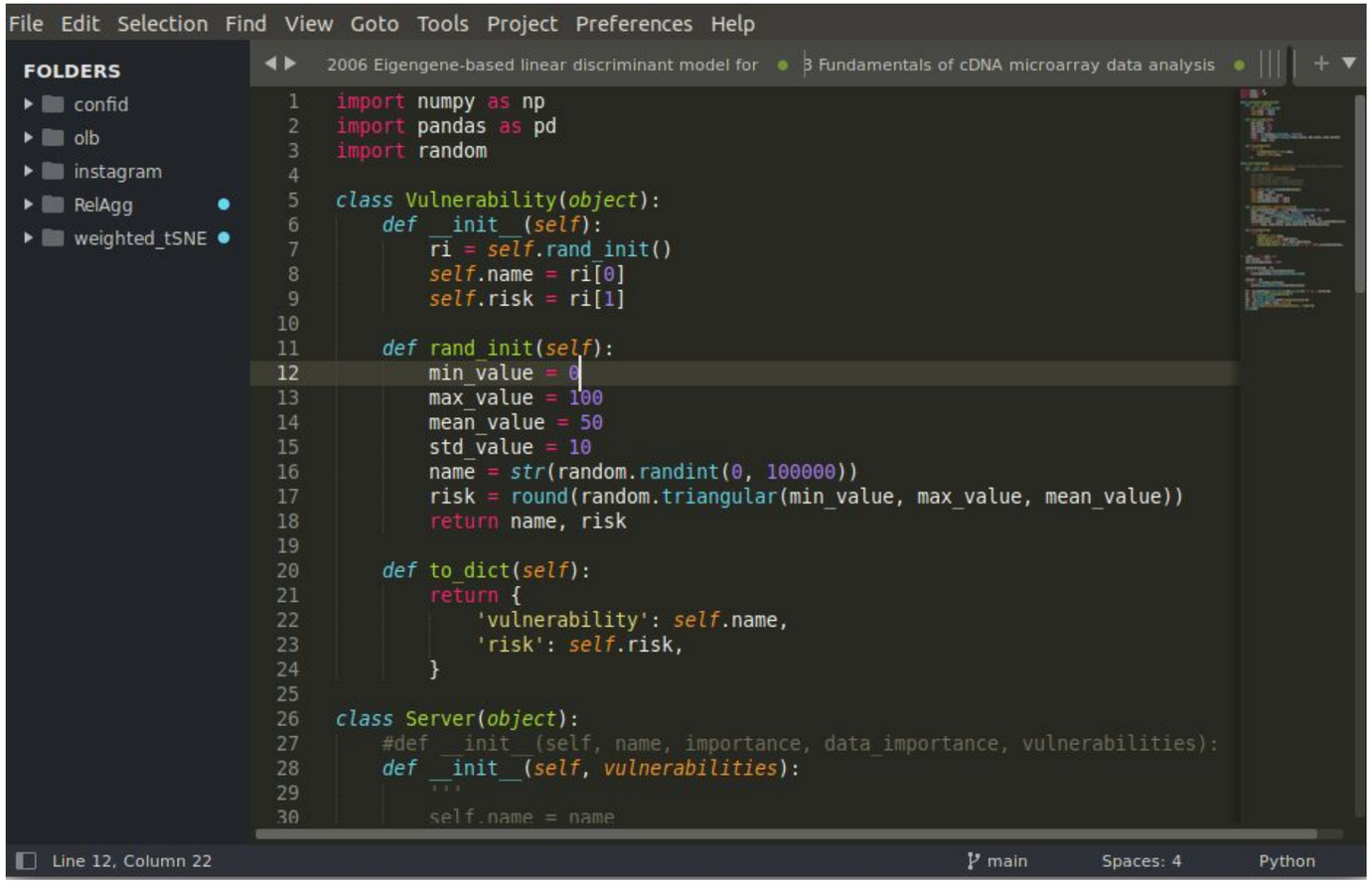
https://www.python.org/doc/sunset-python-2/

Error: não usar um bom editor de texto /IDE

O conforto e a praticidade são importantes para reduzir os erros.

Um bom programa de edição permite:

- Numeração de linha e coluna
- Destaque de linha
- Marcador de par de ({ [
- Sintaxe
- Indentação
- Autocompletar
- Temas
- Gerenciamento de arquivos



https://www.sublimetext.com/

Error: não se planejar

Antes de programar qualquer coisa é uma boa ideia pensar sobre:

- o problema
- a representação
- a solução
- as estruturas de dados
- as ferramentas necessárias
- os possíveis cenários de aplicação

Error: pular a documentação

É essencial ler a documentação disponível dos códigos e bibliotecas utilizados.

Usar apenas a intuição ou experiência para deduzir o funcionamento do código de terceiros pode levar a erros:

- unidades diferentes;
- confusão ou falha de memória;
- diferentes implementações ou definições para um mesmo algoritmo:
- ignorar casos especiais e exceções.

Error: pular a documentação

Silhouette (clustering)

From Wikipedia, the free encyclopedia

Silhouette refers to a method of interpretation and validation of consistency within clusters of data. The technique provides a succinct graphical representation of how well each object has been classified.[1]

The silhouette value is a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation). The silhouette ranges from −1 to +1, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters. If most objects have a high value, then the clustering configuration is appropriate. If many points have a low or negative value, then the clustering configuration may have too many or too few clusters.

The silhouette can be calculated with any distance metric, such as the Euclidean distance or the Manhattan distance.

Kaufman et al. introduced the term *silhouette coefficient* for the maximum value of the mean $s(i)$ over all data of the entire dataset.[3]

$$SC = \max_k \tilde{s}\,(k)$$

Where $\tilde{s}\,(k)$ represents the mean $s(i)$ over all data of the entire dataset for a specific number of clusters k.

https://en.wikipedia.org/wiki/Silhouette_(clustering)

sklearn.metrics.silhouette_score

`sklearn.metrics.` **`silhouette_score`**(*X*, *labels*, *, *metric='euclidean'*, *sample_size=None*, *random_state=None*, ***kwds*) [source]

Compute the mean Silhouette Coefficient of all samples.

The Silhouette Coefficient is calculated using the mean intra-cluster distance (`a`) and the mean nearest-cluster distance (`b`) for each sample. The Silhouette Coefficient for a sample is `(b - a) / max(a, b)`. To clarify, `b` is the distance between a sample and the nearest cluster that the sample is not a part of. Note that Silhouette Coefficient is only defined if number of labels is `2 <= n_labels <= n_samples - 1`.

This function returns the mean Silhouette Coefficient over all samples. To obtain the values for each sample, use **`silhouette_samples`**.

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html

Error: não escrever a documentação

Importante para os outros e para nós mesmos.

- Descrever casos de uso;
- Listar exceções;
- Fornecer exemplos;
- Descrever entradas e saídas;
- Listar erros e problemas comuns e suas soluções;
- Especificar casos não recomendados e limitações;
- Apontar necessidade de pacotes externos e suas versões.
- Tentar ser o mais completo e objetivo possível.

Error: não comentar

A documentação dentro do código.

Descrever entradas, saídas e objetivo de funções e métodos.

Adicionar detalhes importantes sobre variáveis.

Se comunicar com outras pessoas lendo o código (inclusive você no futuro).

Ser objetivo, mas de forma clara e com todas as informações relevantes.

Não escrever só o óbvio.

Justificar decisões.

Fornecer referências.

Documentação NÃO serve para substituir código mal feito.

Error: não comentar

```python
# A single comment

'''
A large comment.
Several lines
'''

# Using docstring

def string_reverse(str1):
    '''
    Returns the reversed String.
    https://www.datacamp.com/community/tutorials/docstrings-python

    Parameters:
        str1 (str):The string which is to be reversed.

    Returns:
        reverse(str1):The string which gets reversed.
    '''

    reverse_str1 = ''
    i = len(str1)
    while i > 0:
        reverse_str1 += str1[i - 1]
        i = i- 1
    return reverse_str1

print(string_reverse.__doc__)
help(string_reverse)

```

```
Summary or Description of the Function

    Parameters:
    argument1 (int): Description of arg1

    Returns:
    int:Returning value

    More information: https://www.w3schools.com/python/python_comments.asp
```

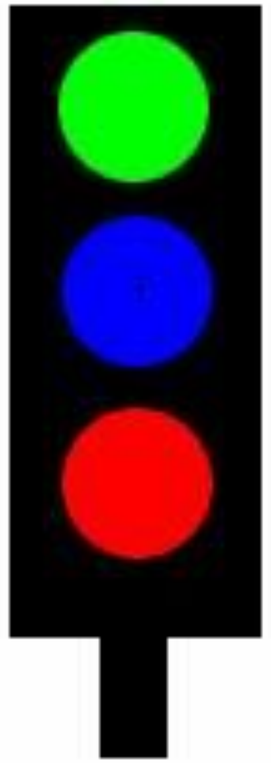
Error: ignorar convenções

Pode levar a confusão ou erro.

Devemos evitar quebrar expectativas.

Respeitar convenções facilita a legibilidade e compreensão do código por terceiros.

Essencial para trabalhos em equipe.



Error: ignorar convenções

- Variáveis, funções, métodos, pacotes, módulos: lowercase_with_underscores
- Classes e exceções: CapWordsTogether
- Métodos protegidos e funções internas: _single_leading_underscore
- Métodos privados: __double_leading_underscore
- Constantes: CAPS_WITH_UNDERSCORES
- Indentação: usar 4 espaços

$ python3 -m pip install pycodestyle

$ pycodestyle --first examples.py

https://www.python.org/dev/peps/pep-0008/

Error: nomes não significativos

O modo como nomeamos nossas variáveis determinam diretamente a legibilidade do código.

Nomes completos e descritivos podem tornar o código auto-documentado.

Ter a informação completa a respeito de uma variável durante a programação pode evitar erros e confusões.

Facilita a manutenção ou alteração do código no futuro.

Error: nomes não significativos

```
1   # variable to store the mass of a cat in kilograms
2   m = 3.6
3   mass = 3.6
4   cat_mass = 3.6
5   cat_mass_kg = 3.6
```

```
12  for i in range(rows):
13      for j in range(columns):
14          if matrix[i][j] > 0.0:
15              pass
16
17  for sample in range(rows):
18      for gene in range(columns):
19          if gene_expression[sample][gene] > 0.0:
20              pass
```

Error: não usar if __name__ == '__main__'

```python
import sys
import numpy as np

s = "Hello world"
print(s)
```

```python
import sys
import numpy as np

def main():
    s = "Hello world"
    print(s)

if __name__ == '__main__':
    main()
```

Indica ao leitor que o arquivo é um script e pode ser executado.

Evita erros de escopo com variáveis globais.

Evita problemas de importação.

Facilita a sinalização de erros pelo interpretador.

Mais rápido.

https://www.youtube.com/watch?v=g_wlZ9IhbTs

Error: codificação rígida

Escrever valores diretamente no código dificulta a leitura e entendimento.

Muito mais trabalho para editar ou mudar os valores.

Idealmente não deve ser necessário alterar o código para testar configurações diferentes.

Pode introduzir erros que não serão identificados pelo interpretador se:

- erro de digitação
- esquecer de alterar

Error: codificação rígida

```
41      def rand_init(self, vulnerabilities):
42          num_vulnerabilities = round(random.triangular(1, 20, 10))
43          name = str(random.randint(0, 100000))
44          importance = round(random.triangular(0, 5, 3))
45          data_importance = round(random.triangular(0, 5, 3))
46          vulnerabilities = random.sample(vulnerabilities, num_vulnerabilities)
47          return name, importance, data_importance, vulnerabilities
48
```

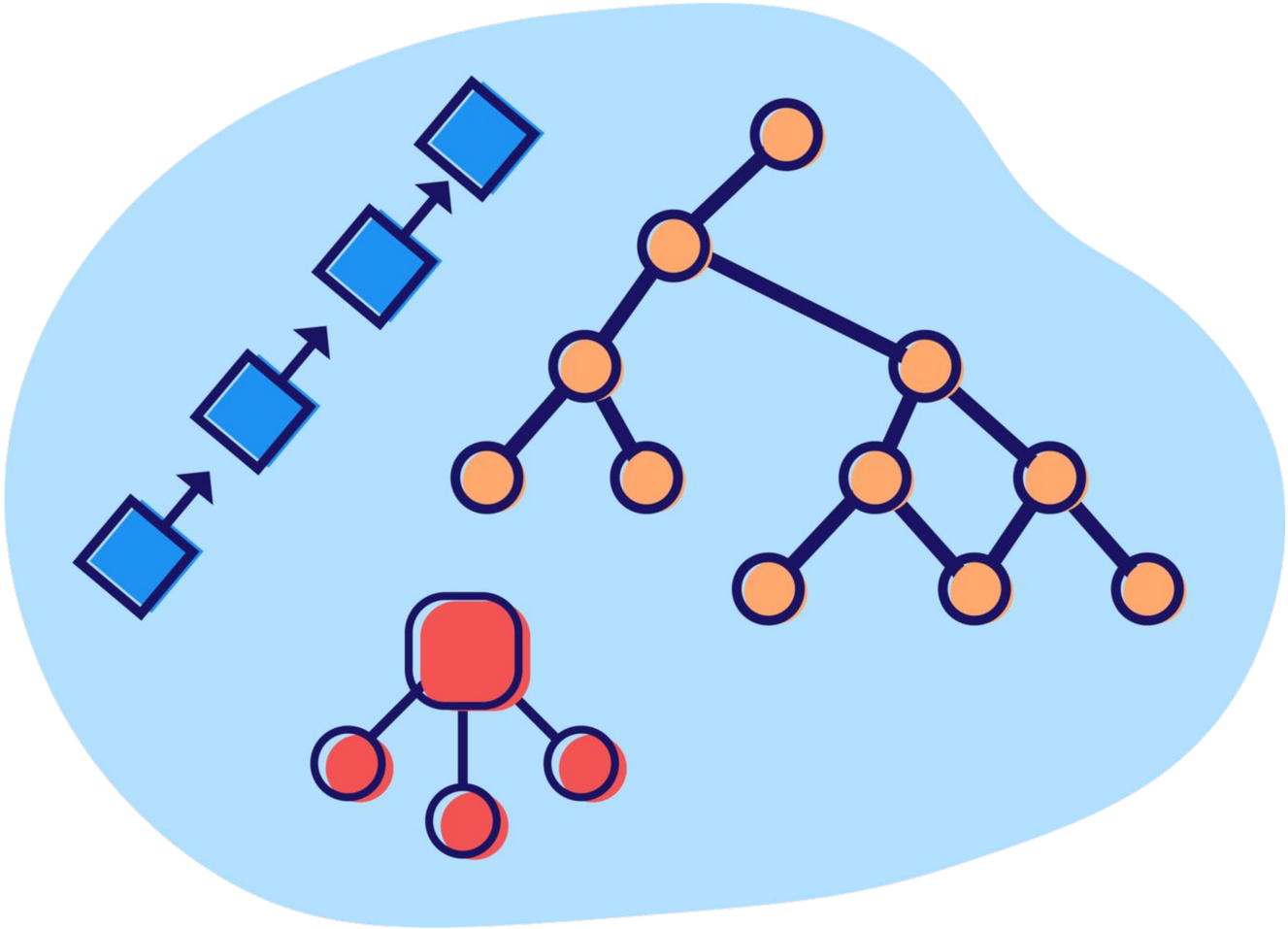
```
217     aver_func = None
218     if mean_type == 'arithmetic':
219         aver_func = np.mean
220     elif mean_type == 'geometric':
221         aver_func = gmean
222     else:
223         raise Exception('Unknown mean type: {}'.format(mean_type))
224
```

Error: estruturas de dados inadequadas

Antes de começar a programar é importante entender e planejar quais estruturas de dados serão utilizadas.

- Lists
- Tuples
- Sets
- Dictionaries

- NumPy array
- Pandas dataframe



https://towardsdatascience.com/which-python-data-structure-should-you-use-fa1edd82946c

Error: repetir código

Repetir linhas de código é a origem de todo mal.

- Dificulta a leitura e o entendimento.
- Mais difícil de encontrar e corrigir erros.
- Alterar o código se torna mais trabalhoso e complicado.
- Aumenta o risco de erros.

Error: falta de modularidade

Definir funções e classes adequadamente.

Mais fácil de encontrar o que se está procurando no código.

Melhor edição e manutenção.

Torna o trabalho em equipe mais prático.

Permite reusabilidade do código.

Facilita a documentação.

Error: não validar entradas

Usuários irão abusar do sistema acidentalmente (ou propositalmente).

É importante ter um sistema robusto para validar os valores que estão sendo alimentados ao programa.

- Casos extremos
- Zero
- Valores negativos
- Tipos errados
- Vazio
- Comprimento de strings.

Try Except

```
81
82  b = sys.argv[1]
83  try:
84      a = 100 / b
85      print (a)
86  except ZeroDivisionError:
87          print ("Zero Division Exception Raised Because User Input Was Zero." )
88  else:
89      print ("Success, no error!")
```

```
93   try:
94       a = 100
95       b = "Workshop"
96       assert a == b
97   except AssertionError:
98           print ("Assertion Exception Raised.")
99   else:
100      print ("Success!")
```

https://www.datacamp.com/community/tutorials/exception-handling-python

https://rollbar.com/blog/throwing-exceptions-in-python/#

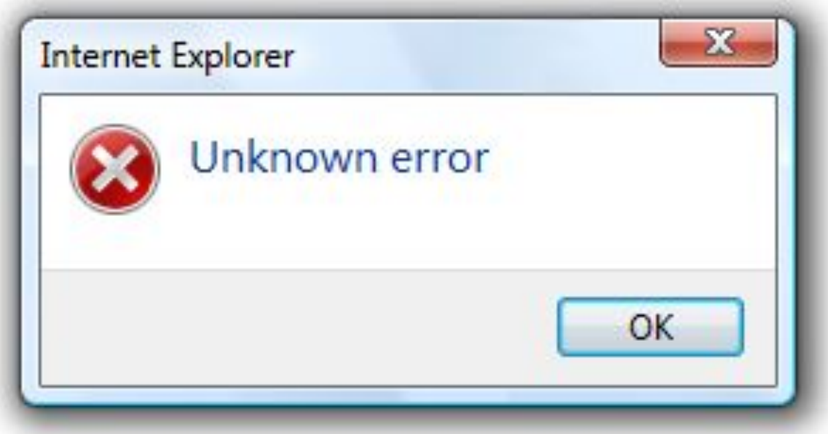
```
102   if len(dna_sequence) > 100:
103       raise Exception('DNA sequence is too large for the application, the limit is 100.')
```

Error: mensagens de erro crípticas ou ausentes

Quando algo dá errado, mensagens de erro são essenciais.

Devem comunicar claramente ao usuário a origem do problema e, se possível, propor soluções.

Não devem depender de documentação externa, mas podem referenciá-la.



Error: falta de warnings

Warnings são avisos de que algo pode estar errado ou fora do desejado na execução, sem que haja problemas na computação.

São essenciais para informar o usuário se ele está obtendo os resultados desejados.

Assim como mensagens de erro, devem ser claros, diretos e informativos.

Error: falta de warnings

```
1   import warnings
2   import sys
3
4   length_threshold = 500
5   dna_sequence = sys.argv[1]
6
7   if len(dna_sequence) > length_threshold:
8       warnings.warn('DNA sequence is larger than what is regular for this application.')
```

Error: reinventar a roda

Salvo casos específicos, é sempre melhor usar código já estabelecido e confiável.

Já foi testado, revisado e corrigido por várias pessoas.

Menos trabalho e mais rápido.

Geralmente é mais eficiente.

```
10  import con_matrix
11  import visualize
12  from microarray_reader import MAR
13
14  def convert_to_class(entry):
15      if type(entry) is not list:
16          entry = entry.tolist()
17      return entry.index(max(entry))
18
19  def get_pre_folds(k, indexes_file):
20      pre_indexes = []
21      with open(indexes_file, 'r') as f:
22          content = f.readlines()
23          content = [x.strip() for x in content]
24          content = [x.replace('TR,', '') for x in content]
25          content = [x.replace('TE,', '') for x in content]
26      for tr, te in zip(content[0::2], content[1::2]):
27          tri = np.fromstring(tr, dtype=int, sep=',')
28          tei = np.fromstring(te, dtype=int, sep=',')
29          pre_indexes.append([tri, tei])
30      if len(pre_indexes) != k:
31          raise ValueError('The number of pre computed folds does not match the informed k.')
32      return pre_indexes
33
34  def start(clf, f, training, testing=None, k=0, save_dir='', num_cores=1, num_gen=1, pre_indexes=None):
35      #READ TRAINING DATASET
36      number_samples = training.number_data_samples
37      number_features = training.number_data_features
38      number_classes = training.number_classes
39      features_labels = training.dataset_symbols
40      classes_labels = training.classes_labels
41      print(number_samples, number_features, number_classes)
42      print(classes_labels)
43
44      #K-FOLD CROSS-VALIDATION
45      k_conf_matrix = None
46      ACC = []
47      FS  = []
48      ERROR_NAMES = []
```

Error: usar libs obscuras

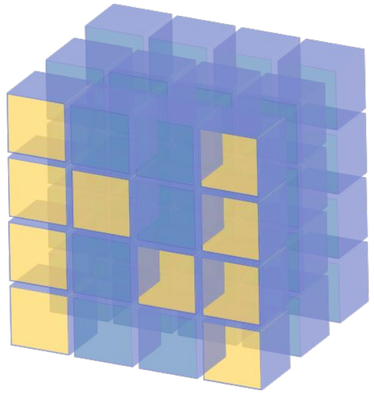
Aumentam o risco do seu código ficar sem suporte no futuro.

Documentação pode ser inacessível.

Problemas de compatibilidade.

Prejudica a reprodutibilidade do código.

Libs úteis



NumPy

scikit learn

pandas

biopython

Error: não testar

Sempre testar bem o código antes de compartilhar.

Usar exemplos representativos e adversários.

Verificar casos extremos.

Comparar versões e ambientes diferentes.

unittest: https://docs.python.org/3/library/unittest.html

Error: não compartilhar

config	added WINDOW_LEN and WINDOW selection	2 years ago
count_populations.py	added WINDOW_LEN and WINDOW selection	2 years ago
count_stay.py	fixed print of decimals	2 years ago
input.inp	application note update	2 years ago
populations.py	added WINDOW_LEN and WINDOW selection	2 years ago
setup.py	new version: ConfID 1.2.1	9 months ago

README.md

ConfID: Conformational Identifier | latest/stable 1.2.1

ConfID: an analytical method for conformational characterization of small molecules using molecular dynamics trajectories

Conformational Identifier
for drug-like molecules
ConfID

Welcome to ***ConfID***!

Conformational generation is a recurrent challenge in early phases of drug design, mostly due to the task of making sense between the number of conformers generated and their relevance for biological purposes.

In this sense, ***ConfID***, a Python-based computational tool, was designed to identify and characterize conformational populations of drug-like molecules sampled through molecular dynamics simulations.

By using molecular dynamics (MD) simulations (and assuming accurate parameters are used), ***ConfID*** can identify all conformational populations sampled in the presence of solvent and quantify their relative abundance, while harnessing the benefits of MD and calculating time-dependent properties of each conformational population identified.

- ***ConfID*** homepage: http://sbcb.inf.ufrgs.br/confid
- For installation instructions, please read INSTALL.md.
- To download ***ConfID*** from snapcraft: https://snapcraft.io/confid

Search

Breast Cancer Gene t-SNE from Scratch
Python notebook using data from Breast cancer gene expression - CuMiDa · 267 views · 7mo ago · pandas, numpy, seaborn, +1 more

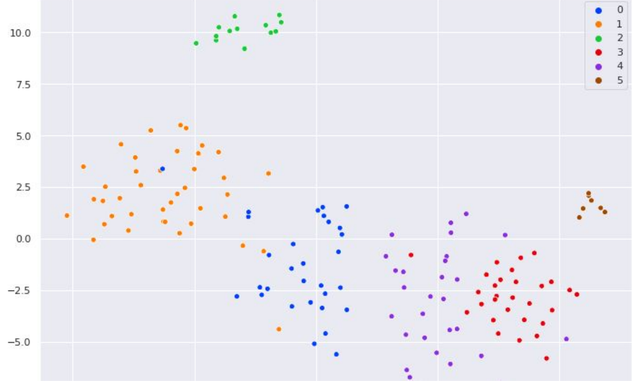
Sklearn t-SNE

```
In [6]: from sklearn.manifold import TSNE

In [7]: tsne = TSNE()
        X_embedded = tsne.fit_transform(features.values)

In [8]: sns.scatterplot(X_embedded[:,0], X_embedded[:,1], hue=labels.values, legend='full', palette=pal
        ette)

Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x7f430fcffa50>
```



Generates images from text prompts with VQGAN and CLIP (codebook sampling method).

By Katherine Crowson (https://github.com/crowsonkb, https://twitter.com/RiversHaveWings) The original BigGAN+CLIP method was by https://twitter.com/advadnoun.

Licensed under the MIT License

Show code

```
[ ] !nvidia-smi

[ ] !git clone https://github.com/openai/CLIP
    !git clone https://github.com/CompVis/taming-transformers
    !pip install ftfy regex tqdm omegaconf pytorch-lightning einops transformers
    !pip install -e ./taming-transformers

[ ] !curl -L 'https://heibox.uni-heidelberg.de/d/8088892a516d4e3baf92/files/?p=%2Fconfigs%2Fmodel.yaml&dl=1' > vqgan_imagenet_f16_1024.yaml
    !curl -L 'https://heibox.uni-heidelberg.de/d/8088892a516d4e3baf92/files/?p=%2Fckpts%2Flast.ckpt&dl=1' > vqgan_imagenet_f16_1024.ckpt
    !curl -L 'https://heibox.uni-heidelberg.de/d/a7530b09fed84f80a887/files/?p=%2Fconfigs%2Fmodel.yaml&dl=1' > vqgan_imagenet_f16_16384.yaml
    !curl -L 'https://heibox.uni-heidelberg.de/d/a7530b09fed84f80a887/files/?p=%2Fckpts%2Flast.ckpt&dl=1' > vqgan_imagenet_f16_16384.ckpt

[ ] import argparse
    import math
    import io
    ort Path
    './taming-transformers')
    ort display
    import OmegaConf
    Image
    els import cond_transformer, vqgan
    t nn, optim
    port functional as F
```

https://github.com/sbcblab/confid

https://www.kaggle.com/alincijov/breast-cancer-gene-t-sne-from-scratch

https://colab.research.google.com/drive/15UwYDsnNeldJFHJ9NdgYBYeo6xPmSelP

Error: esquecer a licença

Adicionar uma licença ao seu código deixa claro suas opções de uso.

Mais segurança para você e quem usar seu código.

Aumenta as chances de uso.

Também é possível fazer o registro de programa de computador no INPI.

- MIT License
- Apache License 2.0
- GPL v3.0

REPÚBLICA FEDERATIVA DO BRASIL
MINISTÉRIO DA ECONOMIA
INSTITUTO NACIONAL DA PROPRIEDADE INDUSTRIAL
DIRETORIA DE PATENTES, PROGRAMAS DE COMPUTADOR E TOPOGRAFIAS DE CIRCUITOS INTEGRADOS

Certificado de Registro de Programa de Computador

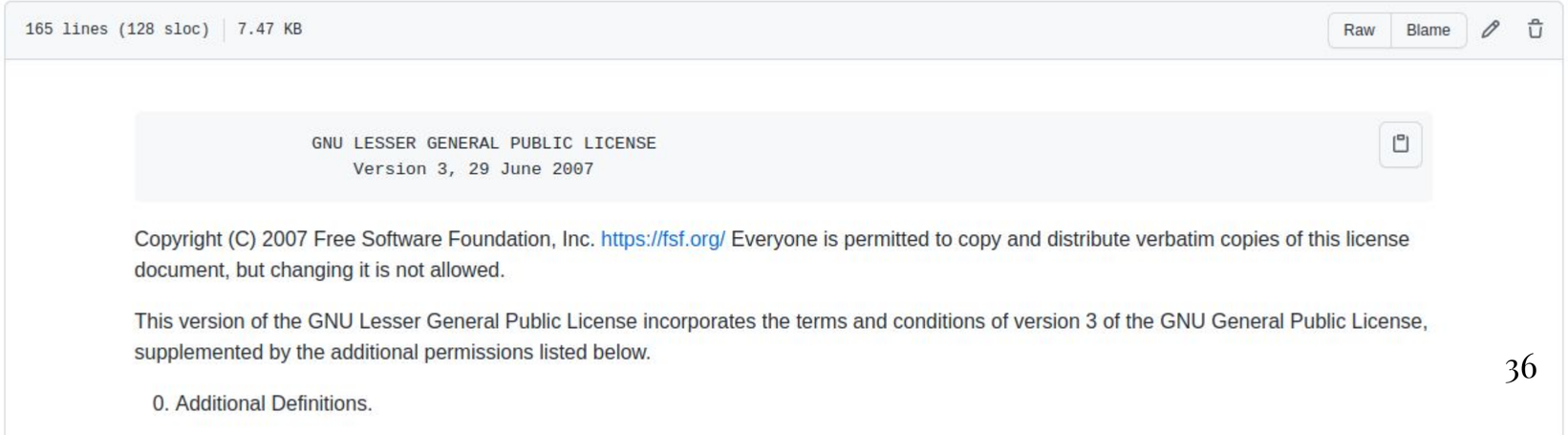
Processo Nº: **BR512019001953-9**

O Instituto Nacional da Propriedade Industrial expede o presente certificado de registro de programa de computador, válido por 50 anos a partir de 1° de janeiro subsequente à data de 03/12/2018, em conformidade com o §2°, art. 2° da Lei 9.609, de 19 de Fevereiro de 1998.

Título: N3O: A NEAT expansion for improving classification and feature selection applied to microarray data

Data de publicação: 03/12/2018

Data de criação: 03/12/2018



165 lines (128 sloc) | 7.47 KB

Raw Blame

GNU LESSER GENERAL PUBLIC LICENSE
Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. https://fsf.org/ Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

This version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented by the additional permissions listed below.

0. Additional Definitions.

https://www.gov.br/inpi/pt-br/servicos/perguntas-frequentes/programas-de-computador

Resumindo...

Erros são inevitáveis...

mas podem ser reduzidos.

Se queremos que outras pessoas usem nosso código precisamos escrevê-lo como tal.

A primeira pessoa beneficiada por um código bem organizado é você mesmo.

```
Python 3.6.9 (default, Jan 26 2021, 15:33:00)
[GCC 8.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
>>>
```

Contato

bigrisci@inf.ufrgs.br

@BrunoGrisci

https://brunogrisci.github.io/

SBCB
Structural Bioinformatics and Computational Biology Lab

https://sbcb.inf.ufrgs.br/

Obrigado!

Perguntas?

APOIO

unesp

USP ESALQ

Pós-Graduação em Microbiologia Agropecuária
Mestrado Doutorado
Jaboticabal
SÃO PAULO - BRASIL

Pós-Graduação em Agronomia
Genética e Melhoramento de Plantas
Jaboticabal
SÃO PAULO - BRASIL

IB Instituto de Biociências Unesp Botucatu

PPG Genética e Melhoramento de Plantas

Alfahelix