III WORKSHOP DE PYTHON PARA DADOS BIOLÓGICOS

PYTHON EM ESTUDOS EVOLUTIVOS:

DE SIMULAÇÕES À ANÁLISE DE DADOS

- 10 DE SETEMBRO, 2020 -

ANDRÉA T. THOMAZ

UNIVERSITY OF BRITISH COLUMBIA – CANADA

UNIVERSIDADE DEL ROSARIO - COLOMBIA

(DEATTHOMAZ@GMAIL.COM)

ORGANIZAÇÃO DA PALESTRA

- O que são simulações e quando utilizá-las?
- Uso de Python em simulações:
  1) Scripts caseiros
  2) Com outros softwares
  3) Sumarizar grandes quantidades de resultados

BIOLOGIA

COMO TU ACHA

QUE VAI SER…

BIOLOGIA

E A REALIDADE!!

AGRADECIMENTOS

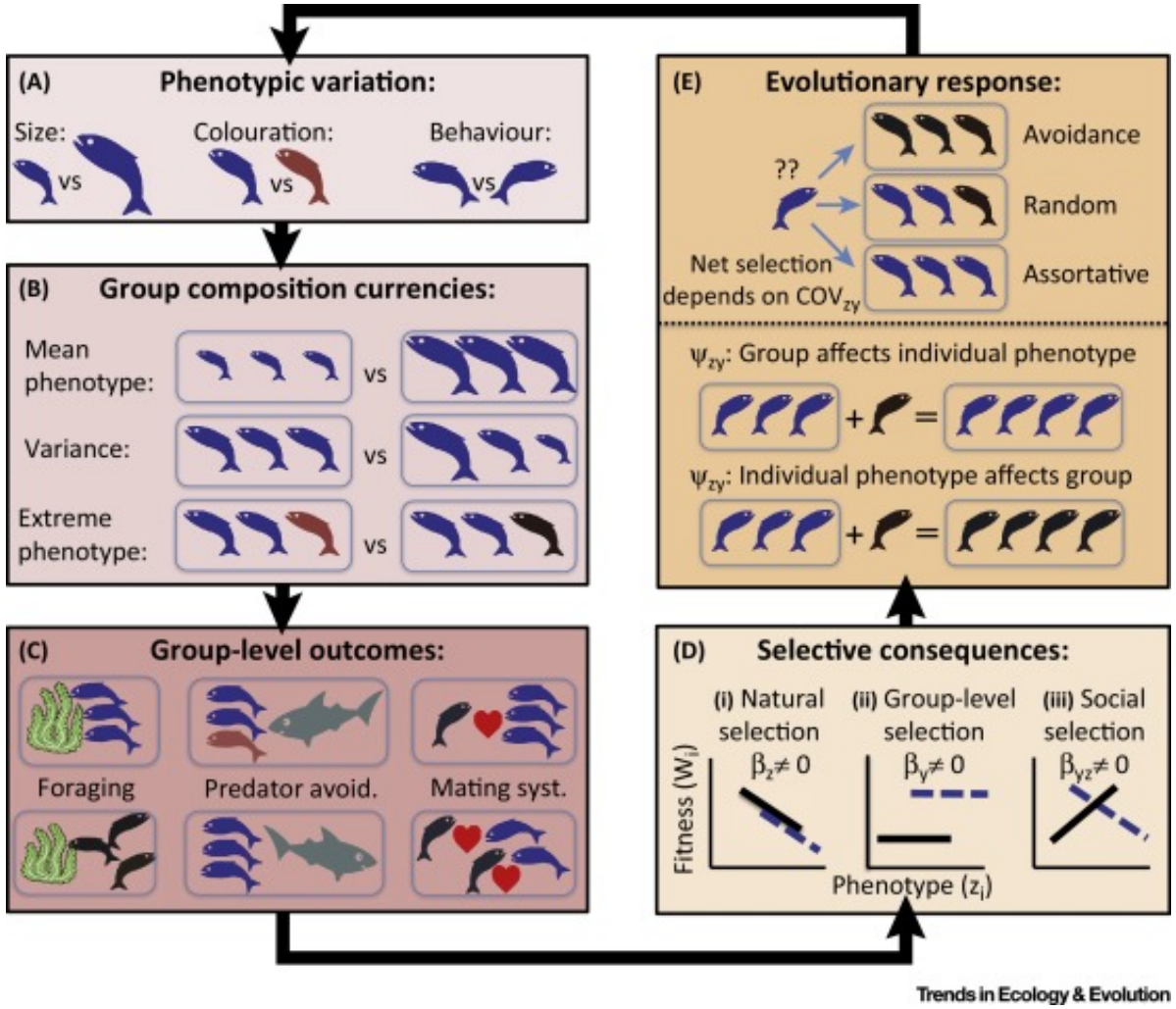
Tom Booker
UBC
(https://github.com/TBooker)

Mike Whitlock
UBC

- Renato Augusto Corrêa dos Santos
- Joyce Prado (ESALQ) e Tiago Carvalho (U. Javeriana)

O QUE SÃO SIMULAÇÕES E QUANDO UTILIZÁ-LAS?

COMPLEXIDADE BIOLÓGICA



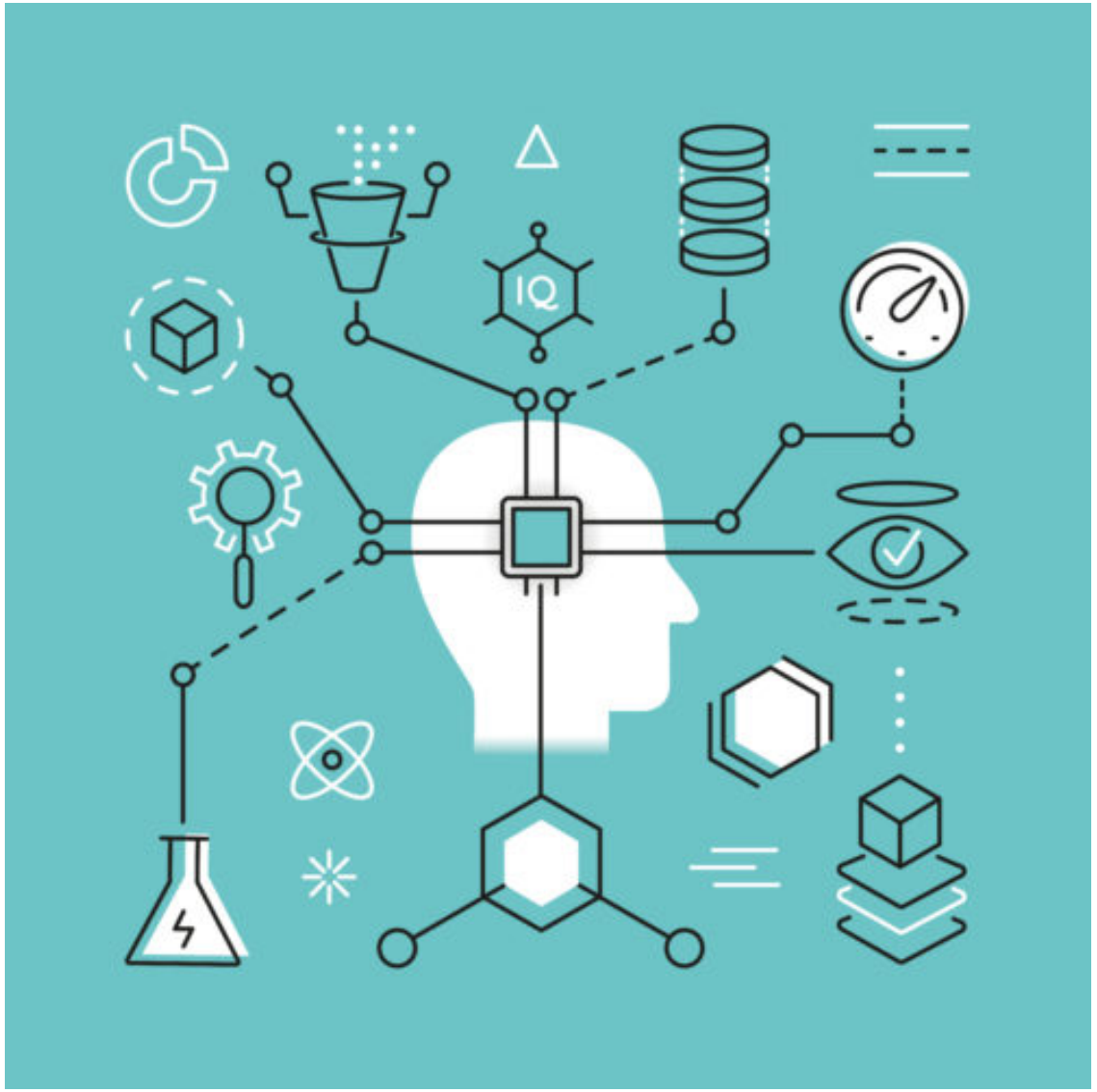
(Farine et al., 2015)

COMO PODEMOS ALCANÇAR UM MELHOR ENTENDIMENTO DESSA COMPLEXIDADE BIOLÓGICA?

Abstrações,

simplificações

Através do uso de modelos!



(Fonte: https://www.litmos.com/es-LA/blog/articles/mental-models-learning-design)

MODELOS MENTAIS

MODELOS MENTAIS

- Raciocinar depende de prever as possibilidades dado um ponto de início
- Levando em consideração as potenciais variáveis, criamos modelos mentais para cada possibilidade e derivamos uma conclusão
- Abilidade de utilizar exemplos/informações prévias (*priors*) para refutar inferências inválidas proporciona a fundação para racionalidade

Raciocínio é uma simulação do mundo dado nosso conhecimento (parâmetros)

(Johnson-Laird, 2010 - PNAS)

O QUE SÃO SIMULAÇÕES?

Imitações da realidade de maneira simplificada
(certo número de parâmetros)
de um processo ou sistema ao longo do tempo

(Arenas, 2012 - Plos Comp. Biol.)

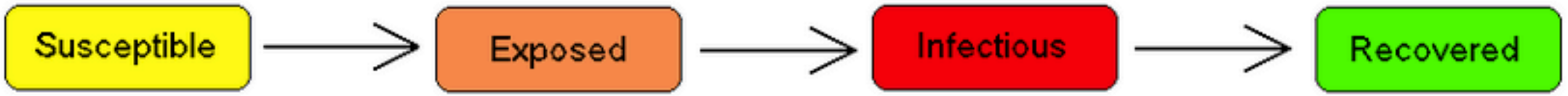
(https://covid19-projections.com/)

PROJEÇÕES COVID-19

YYG / covid19-projections.com SEIR Simulator

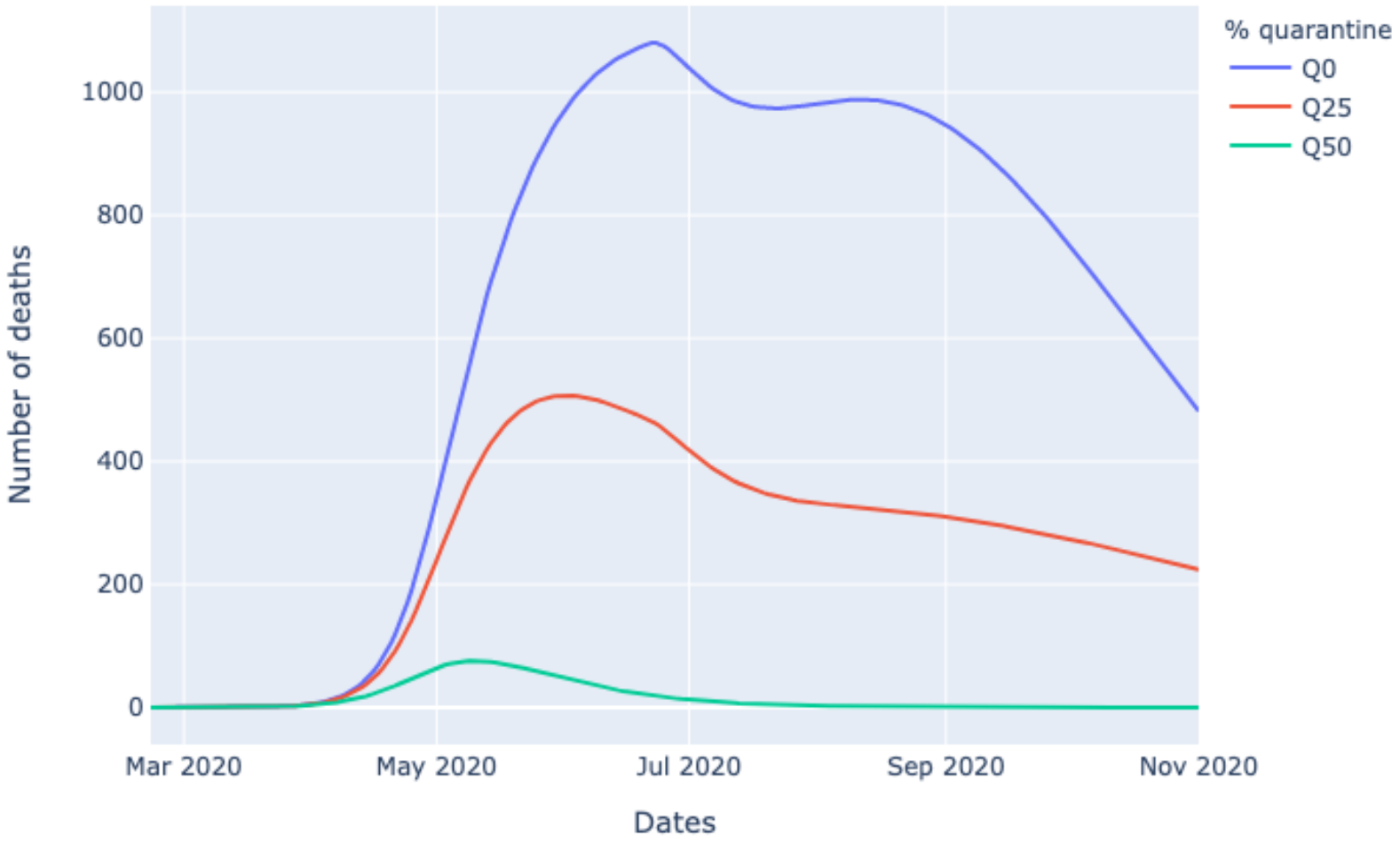
We present the underlying SEIR model simulator behind the YYG / covid19-projections.com model, as well a summarized set of parameters that helped generate the projections.

If your system supports Python, you can generate your own simulations in under 5 minutes. No prior Python experience is needed to run this tool. Get started here.



(https://covid19-projections.com/)

git clone https://github.com/youyanggu/yyg-seir-simulator.git



SIMULAÇÃO DE MORTES CAUSADAS POR COVID-19 NO BRASIL

EM FUNÇÃO DA % POPULAÇÃO EM QUARENTENA

QUANDO UTILIZAR SIMULAÇÕES?

- discernir entre possibilidades (hipóteses)
- comparar e verificar métodos e ferramentas analíticas
- estimar parâmetros
- entender a interação entre processos

(Arenas, 2012 - Plos Comp. Biol.)

QUANDO UTILIZAR SIMULAÇÕES?

Para problemas simples... soluções simples
(Ex: teste-T)

- Matemática muito complicada (pelo menos para mim!)
- Testar a estocasticidade do sistema ("*noise*": pelo processo ou pela amostragem)

- Geralmente requer muitas réplicas para abranger a distribuição total do parâmetro amostral -

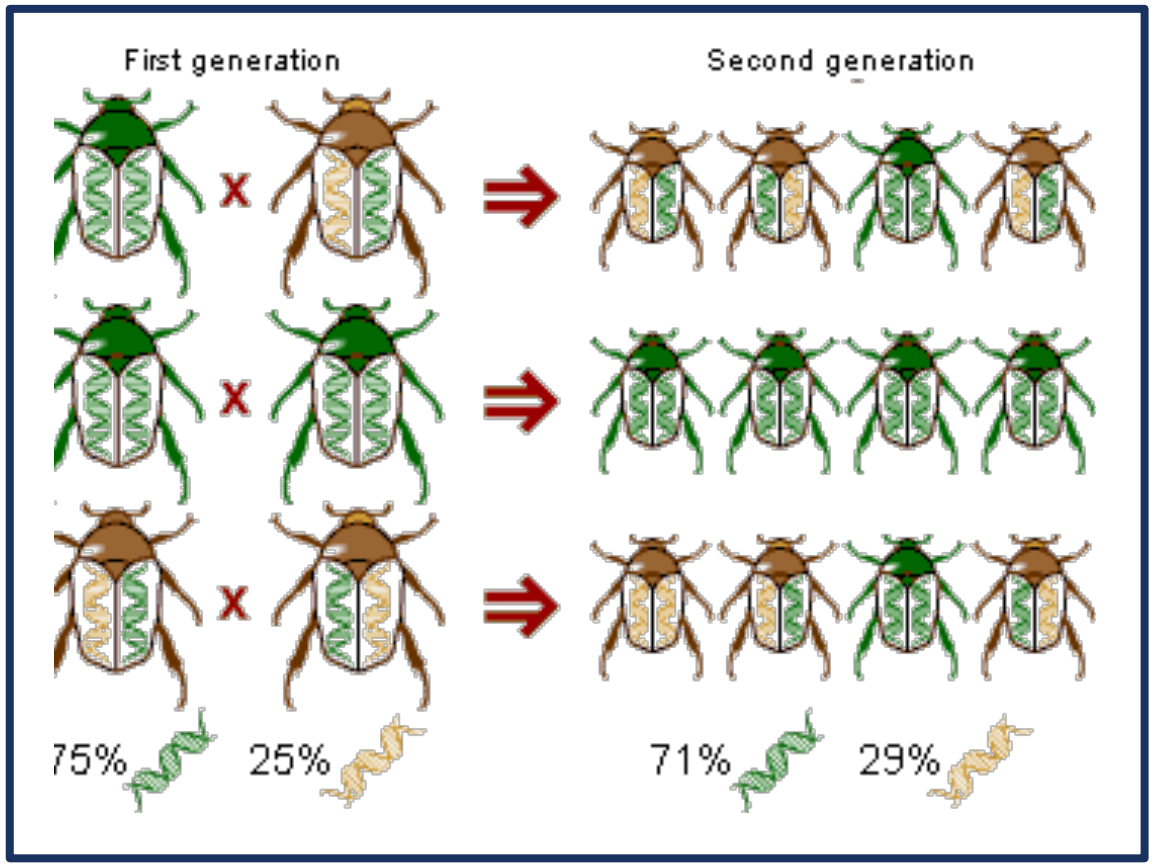
USO DE PYTHON EM SIMULAÇÕES BIOLÓGICAS:

1) **SCRIPTS CASEIROS**

2) COM OUTROS SOFTWARES

3) SUMARIZAR GRANDES QUANTIDADES DE RESULTADOS

DERIVA GENÉTICA



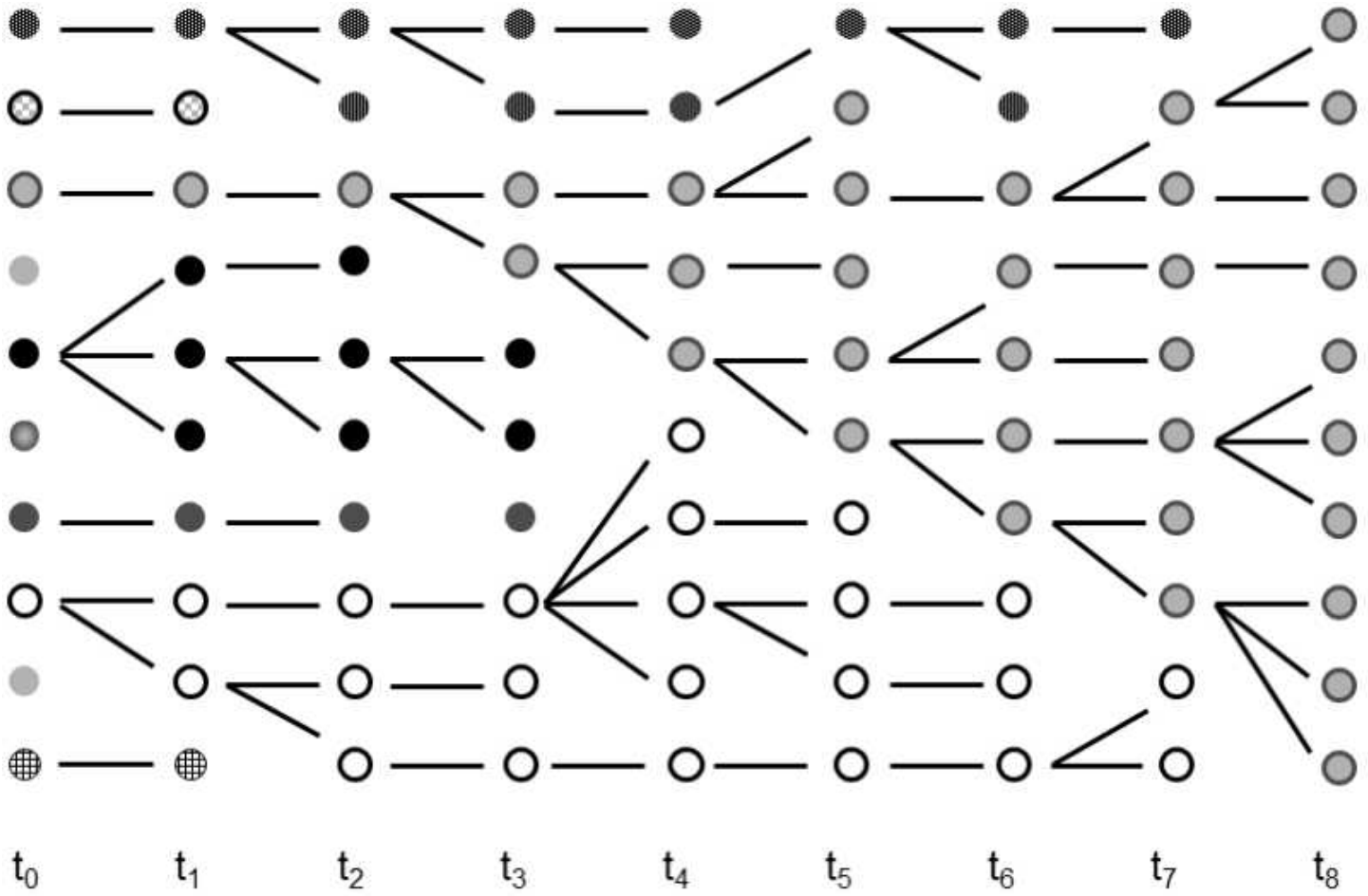
(Fonte: https://evolution.berkeley.edu/evolibrary/home.php)

Variação na frequência relativa dos alelos nas populações ao longo do tempo ao acaso

ocorre em todas as populações de tamanho não infinito

Em função do tamanho populacional, quando deriva vai ser mais forte?

1) SCRIPTS CASEIROS



(Fonte: https://www.frozenevolution.com/v2-populations-finite-size-genetic-drift-leads-fixation-some-alleles)

1) SCRIPTS CASEIROS

- Script modificado de original

  GitHub: https://github.com/TBooker/GlobalAdaptation (autor: Tom Booker, UBC)

- Pacotes utilizados:

  - numpy: gerar "random seed", soma e vetores

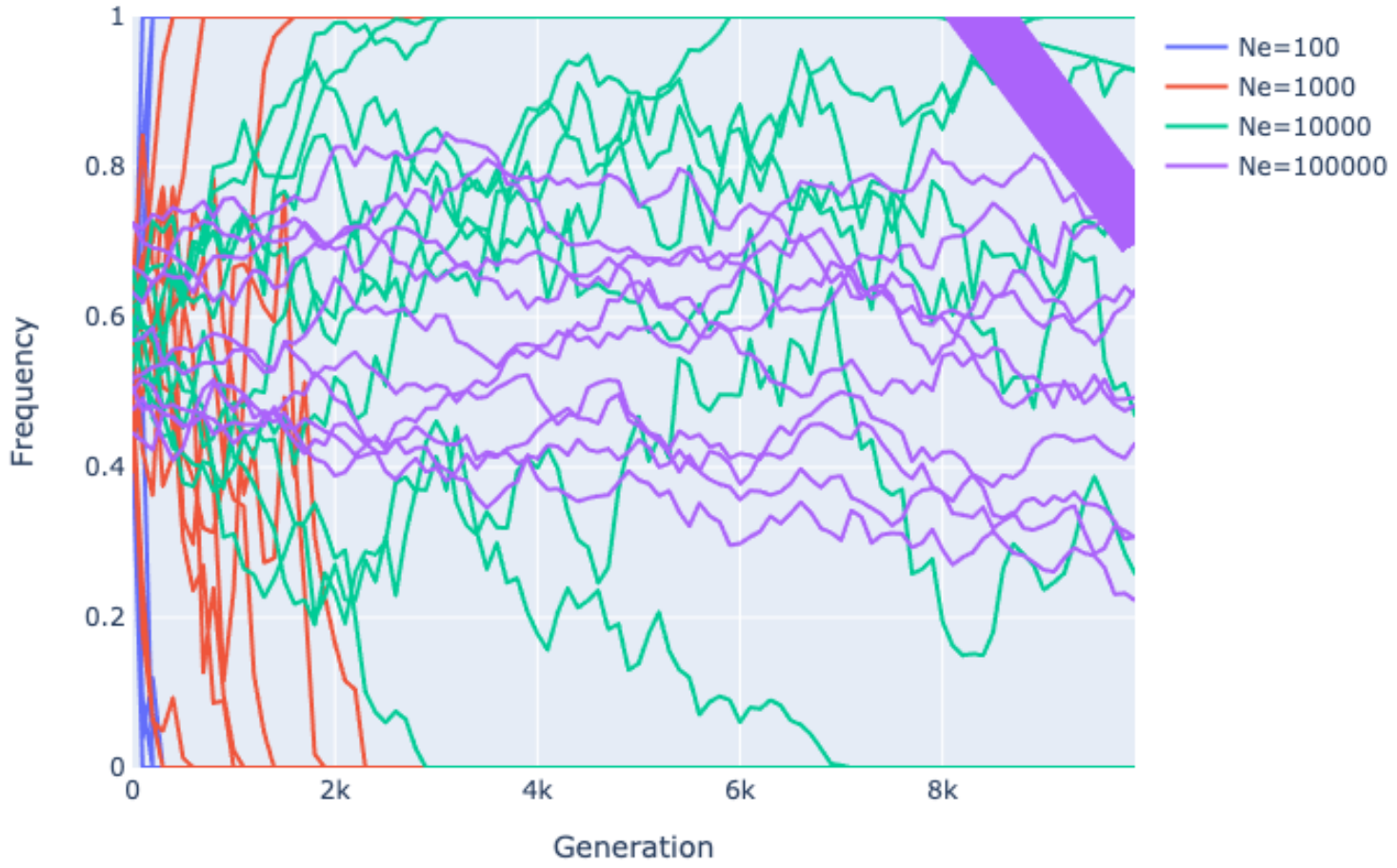
  - argparse: para passar comandos ao chamar o script

- Linha de comando:

```
python drifter.py -k 1 -N 100 -m 0 -p 0.5 -d 10000 -o driftedPop_N100.csv --store
```

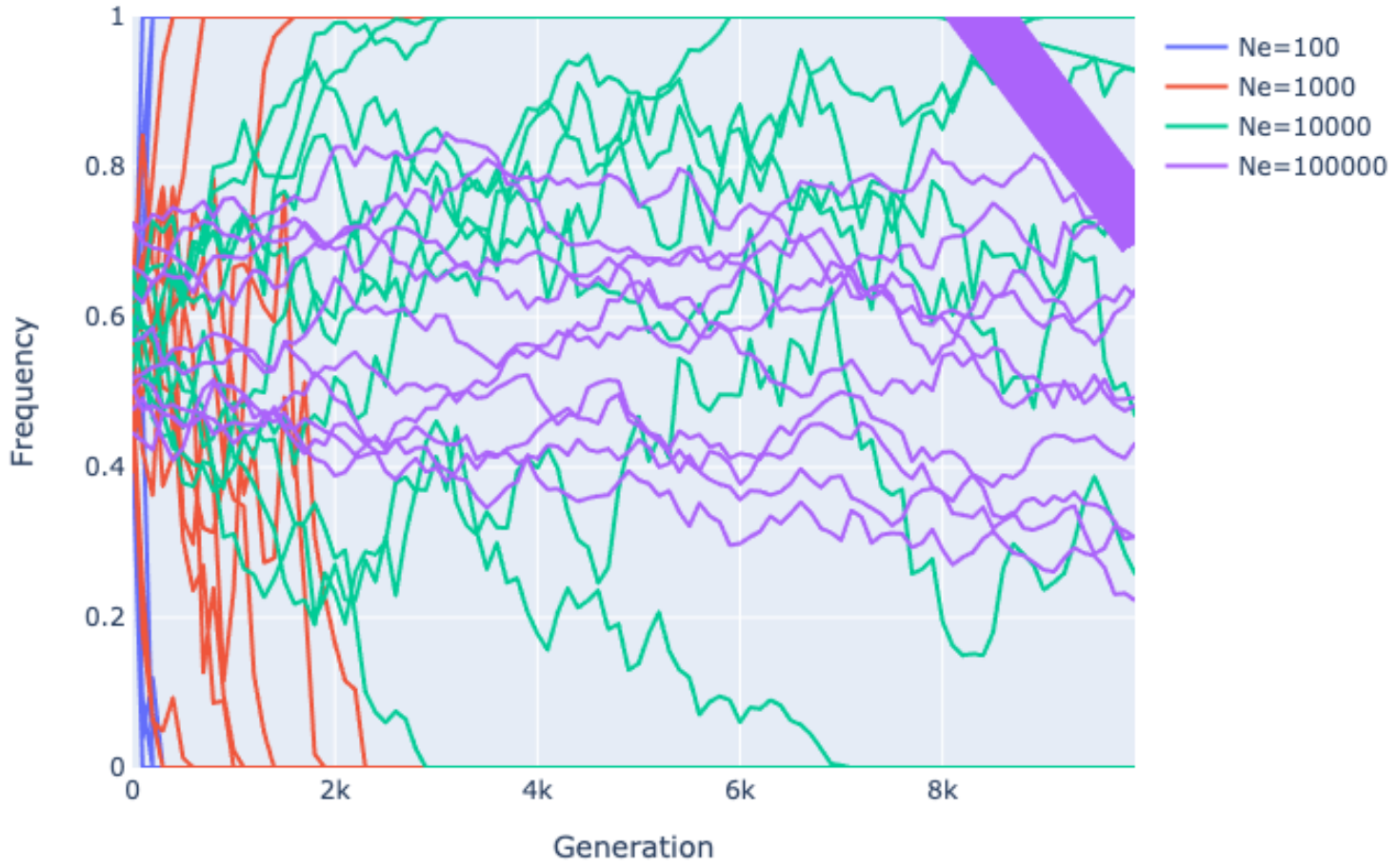
1) SCRIPTS CASEIROS

- Uma população
- Diferentes tamanhos populacionais
- 10k gerações
- Frequência inicial 0.5



1) SCRIPTS CASEIROS

Tamanhos populacionais pequenos sofrem deriva mais forte, perdendo diversidade genética mais rapidamente



USO DE PYTHON EM SIMULAÇÕES BIOLÓGICAS:

1) SCRIPTS CASEIROS

2) COM OUTROS SOFTWARES

3) SUMARIZAR GRANDES QUANTIDADES DE RESULTADOS

2) PYTHON COM OUTROS SOFTWARES E PACOTES JÁ DESENVOLVIDOS

Simulações podem ficar complexas

de uma maneira bastante rápida

- Aumentar o número de populações
- Fluxo gênico (migração)
- Taxa de mutação
- Seleção afetando o fitness...

Diversos recursos disponíveis para simular diferents tipos de dados

2) PYTHON COM OUTROS SOFTWARES E PACOTES JÁ DESENVOLVIDOS

Exemplos de simuladores em Python:

- Biopython: https://biopython.org/wiki/PopGen
- Dendropy: https://dendropy.org/
- DEAP: https://github.com/DEAP/deap
- ***msprime:*** https://msprime.readthedocs.io/en/stable/tutorial.html

Outros utilizados juntamente com Python

- SLIM: https://messerlab.org/slim/
- FastSimcoal: http://cmpg.unibe.ch/software/fastsimcoal2/

(não é uma lista extensa!)

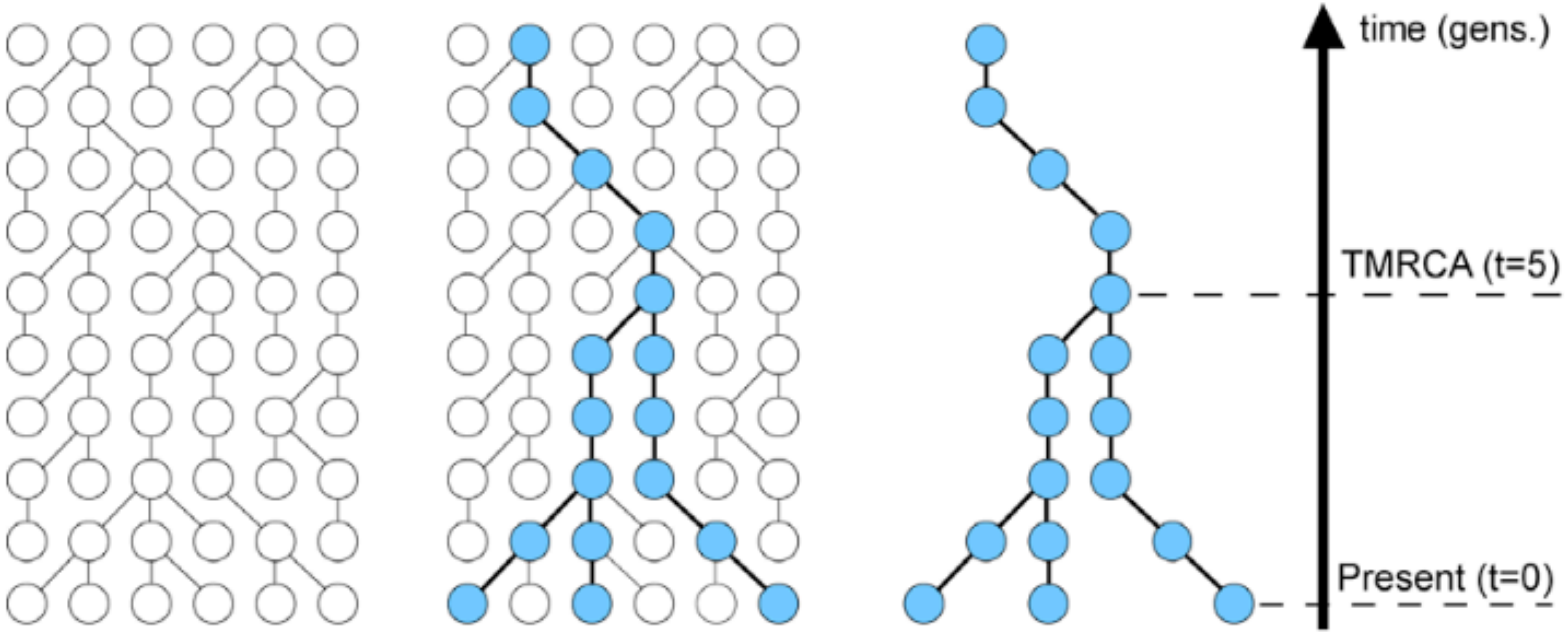
2) PYTHON COM OUTROS SOFTWARES E PACOTES JÁ DESENVOLVIDOS

msprime

Quantas gerações são necessárias para alcançar coalescência em função do tamanho populacional?

Coalescência

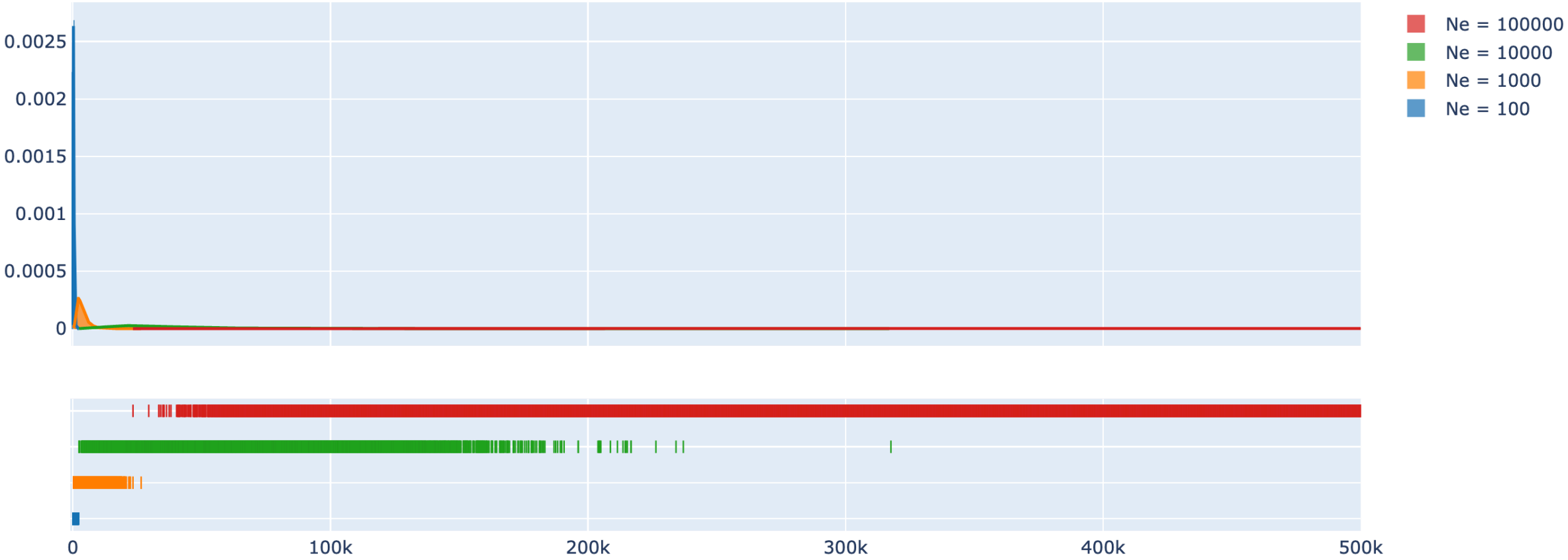
Amostras da população se originaram de um ancestral comum



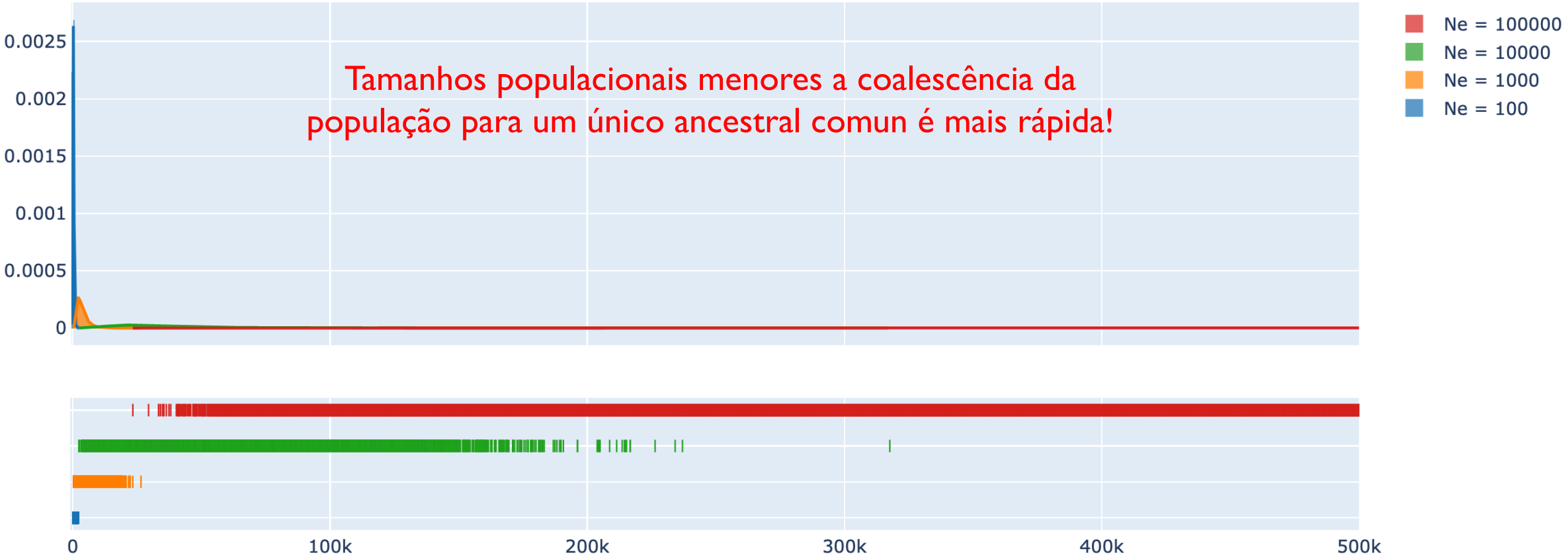
$$\approx \quad 4N_e$$

```
import msprime
import plotly.figure_factory as ff

#number of replicates to perform per Ne
num_replicates = 10000
#array with population sizes to be simulated
Ne_array=[ 100 , 1000 , 10000, 100000 ]
#generate empty array to append in the for loop
T = []

#for loop to simulate trees for each population size
for n in Ne_array:
    replicates = msprime.simulate(sample_size=10, Ne=n, num_replicates=num_replicates)
    # And then iterate over these replicates to calculate the time since the MRCA
    for i, tree_sequence in enumerate(replicates):
        tree = tree_sequence.first()
        T.append( tree.time(tree.root) ) #this will be a long vector with all divergences

#making a data frame with time and Ne
df = pd.DataFrame({'Ne = 100': T[0:num_replicates],
    'Ne = 1000': T[num_replicates:num_replicates*2],
    'Ne = 10000': T[num_replicates*2:num_replicates*3],
    'Ne = 100000': T[num_replicates*3:num_replicates*4]})
#show means
print(df.mean())

#Plotting :)
fig = ff.create_distplot([df[c] for c in df.columns], df.columns, bin_size=.25)
fig.update_xaxes(range=[-1000, 250000])
fig.show()
#fig.write_image("./Figures/Tmrca_distr.png")
```

2) PYTHON COM OUTROS SOFTWARES E PACOTES JÁ DESENVOLVIDOS



2) PYTHON COM OUTROS SOFTWARES E PACOTES JÁ DESENVOLVIDOS



USO DE PYTHON EM SIMULAÇÕES BIOLÓGICAS:

1) SCRIPTS CASEIROS
2) COM OUTROS SOFTWARES
3) SUMARIZAR GRANDES QUANTIDADES DE RESULTADOS

3) SUMARIZAR GRANDES QUANTIDADES DE RESULTADOS

Geralmente requer muitas réplicas

para abranger a distribuição total do parâmetro amostral

- Manipular grandes quantidades de arquivos/tamanhos
- Sumarizar, unir, organizar
- Análises estatísticas
- Visualização gráfica

3) SUMARIZAR GRANDES QUANTIDADES DE RESULTADOS

```
#Plotar a distribuição
fig = px.histogram(chi_distr, labels={"count": "Frequência","value": "X2: df=1"},color_discrete_sequence=["gray"])
# Line Vertical
# add chi_dado line
fig.add_shape(type="line", x0=chi_dado, y0=0, x1=chi_dado, y1=2000,
    line=dict(color="Red",width=4,dash="solid",),)
# add 95% expectation line
fig.add_shape(type="line", x0=alpha95, y0=0, x1=alpha95, y1=2000,
    line=dict(color="Black",width=4,dash="dash",),)
fig.layout.update(showlegend=False)
fig.show()
```

```
#making a data frame with time and Ne
df = pd.DataFrame({'Ne = 100': T[0:num_replicates],
    'Ne = 1000': T[num_replicates:num_replicates*2],
    'Ne = 10000': T[num_replicates*2:num_replicates*3],
    'Ne = 100000': T[num_replicates*3:num_replicates*4]})
```

```
#show means
print(df.mean())
```

```
#reading data frames with pandas
df1e2 = pd.read_csv('./Scripts/02_SteppingStone/driftedPop_Rep10_N100.csv', names=['Generation','Frequency','Replicate'])
df1e3 = pd.read_csv('./Scripts/02_SteppingStone/driftedPop_Rep10_N1000.csv', names=['Generation','Frequency','Replicate'])
df1e4 = pd.read_csv('./Scripts/02_SteppingStone/driftedPop_Rep10_N10000.csv', names=['Generation','Frequency','Replicate'])
df1e5 = pd.read_csv('./Scripts/02_SteppingStone/driftedPop_Rep10_N100000.csv', names=['Generation','Frequency','Replicate'])

#merging all the data frames into a single one
df = pd.concat([df1e2,df1e3,df1e4,df1e5], keys=["Ne=100","Ne=1000","Ne=10000","Ne=100000"]).reset_index()
```

3) SUMARIZAR GRANDES QUANTIDADES DE RESULTADOS

Exemplo:

Muitos parâmetros com diferentes valores e muitas réplicas cada

(8 parâmetros: 2-5 valores: 1000 réplicas)

Combinação de parâmetros = ~60,000 * 1,000 réplicas = 60 milhões de *outputs* a serem avaliados

Combinação entre comandos:

1) *bash* (*awk*) para procurar por um padrões nos arquivos e gerar médias
2) *Python* com *pandas* para unir todas essas médias e gerar arquivos que nos possibilite explorar o espaço de parâmetros
3) Muita diversão analisando a interação entre todas essas variáveis!!

3) SUMARIZAR GRANDES QUANTIDADES DE RESULTADOS

E o que fazer agora com tantos dados?

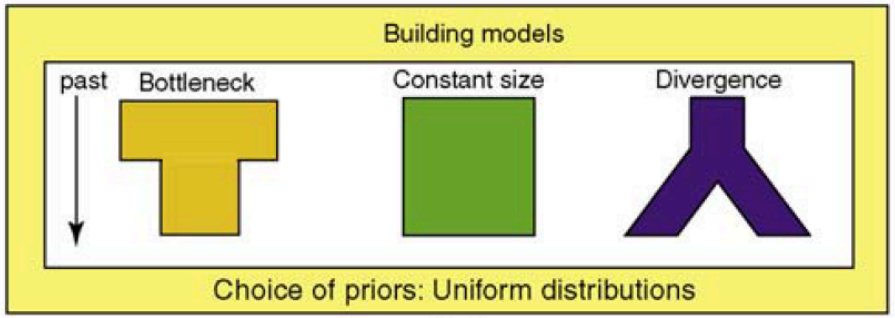
Infinitas possibilidades :p

- Gerar expectativas teóricas
- Previsões (forecasting)
- Mas também podemos comparar com nossos dados empíricos…

3) SUMARIZAR GRANDES QUANTIDADES DE RESULTADOS

ABC

Approximate Bayesian Computation

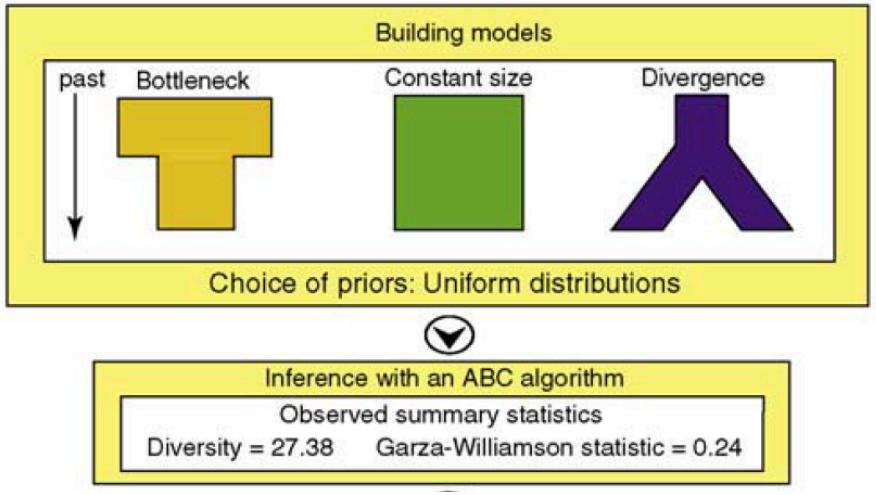


(Csilléry et al., 2010)

3) SUMARIZAR GRANDES QUANTIDADES DE RESULTADOS

ABC

Approximate Bayesian Computation

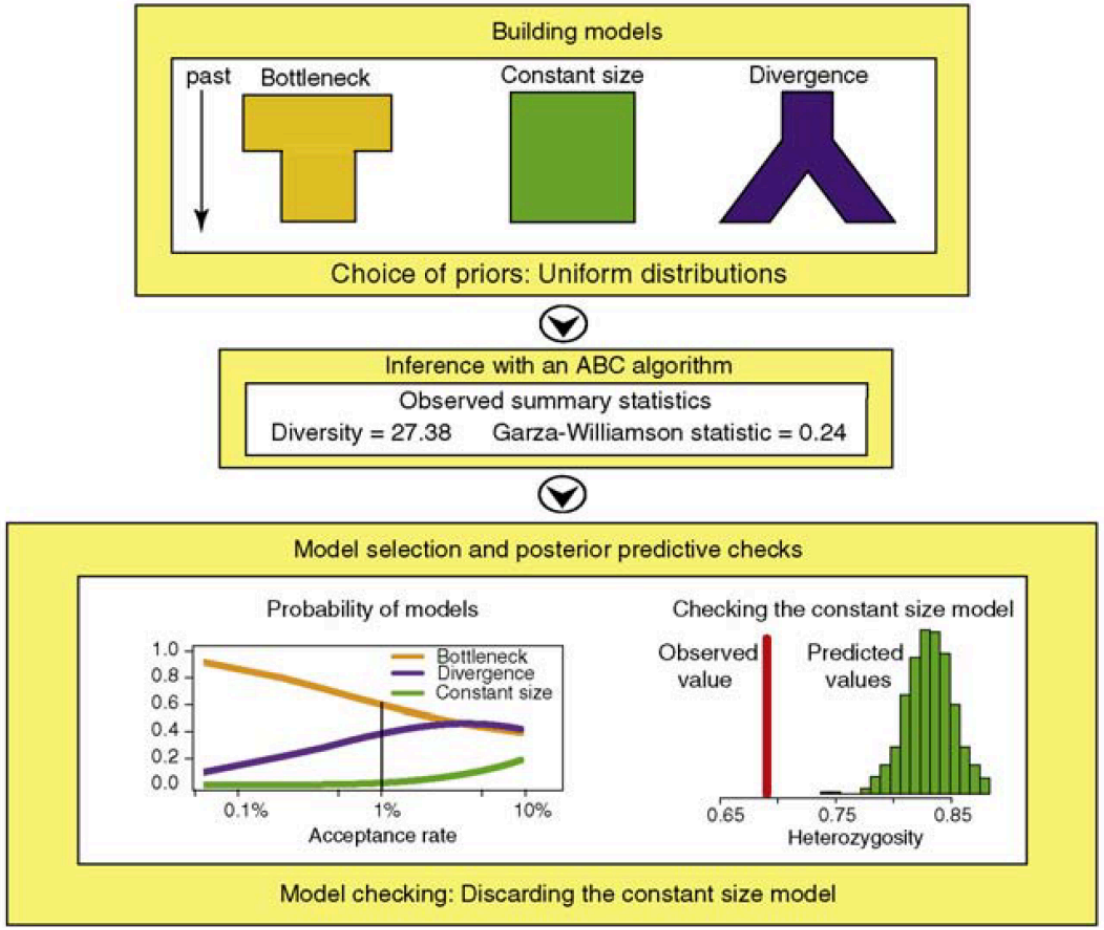


(Csilléry et al., 2010)

3) SUMARIZAR GRANDES QUANTIDADES DE RESULTADOS

ABC

Approximate Bayesian Computation

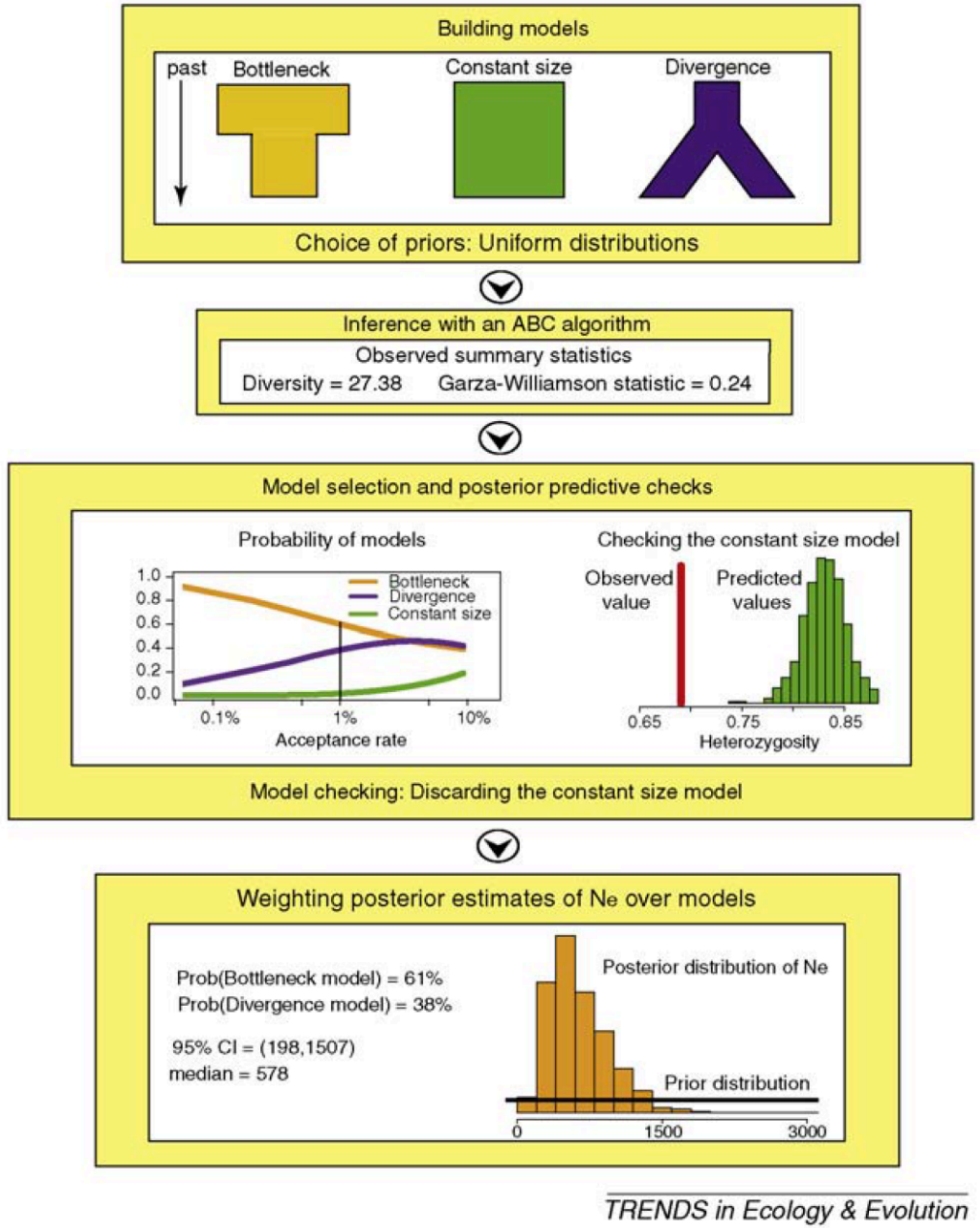


(Csilléry et al., 2010)

3) SUMARIZAR GRANDES QUANTIDADES DE RESULTADOS

ABC

Approximate Bayesian Computation



(Csilléry et al., 2010)

TRENDS in Ecology & Evolution

"TAKE HOME MESSAGES"

- Simulações são amplamente utilizadas em estudos evolutivos
- Possibilita testar hipóteses de maneira relativamente barata (necessita recursos computacionais)
- Gerar hipóteses a priori para contrastar com nossos dados empíricos
- Ótimo durante o tempo de quarentena!
- Conhecimento de bioinformática (ex. *Python*) é fundamental mesmo se tu quer ser "só" biólogo :)

PERGUNTAS, COMENTÁRIOS, CONTRIBUIÇÕES!

deatthomaz@gmail.com
Twitter: @IchthyaDea
GitHub:
ichthya/WorkshopPythonBiol2020_SimulationsScripts

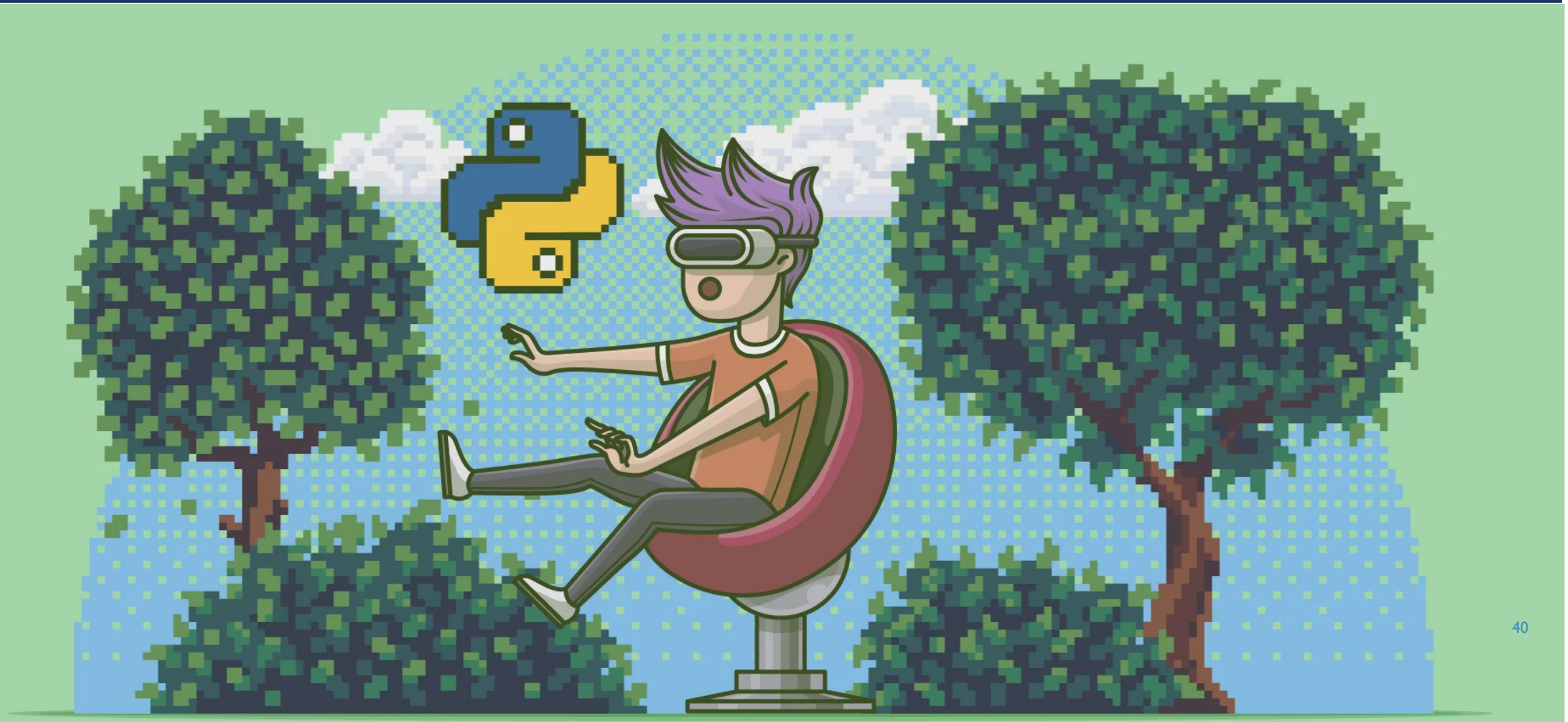


Foto: Real Python