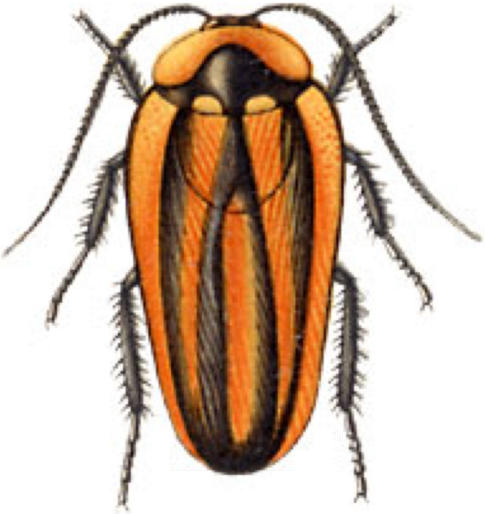
One way that programming helps us understand biology

Rob Lanfear

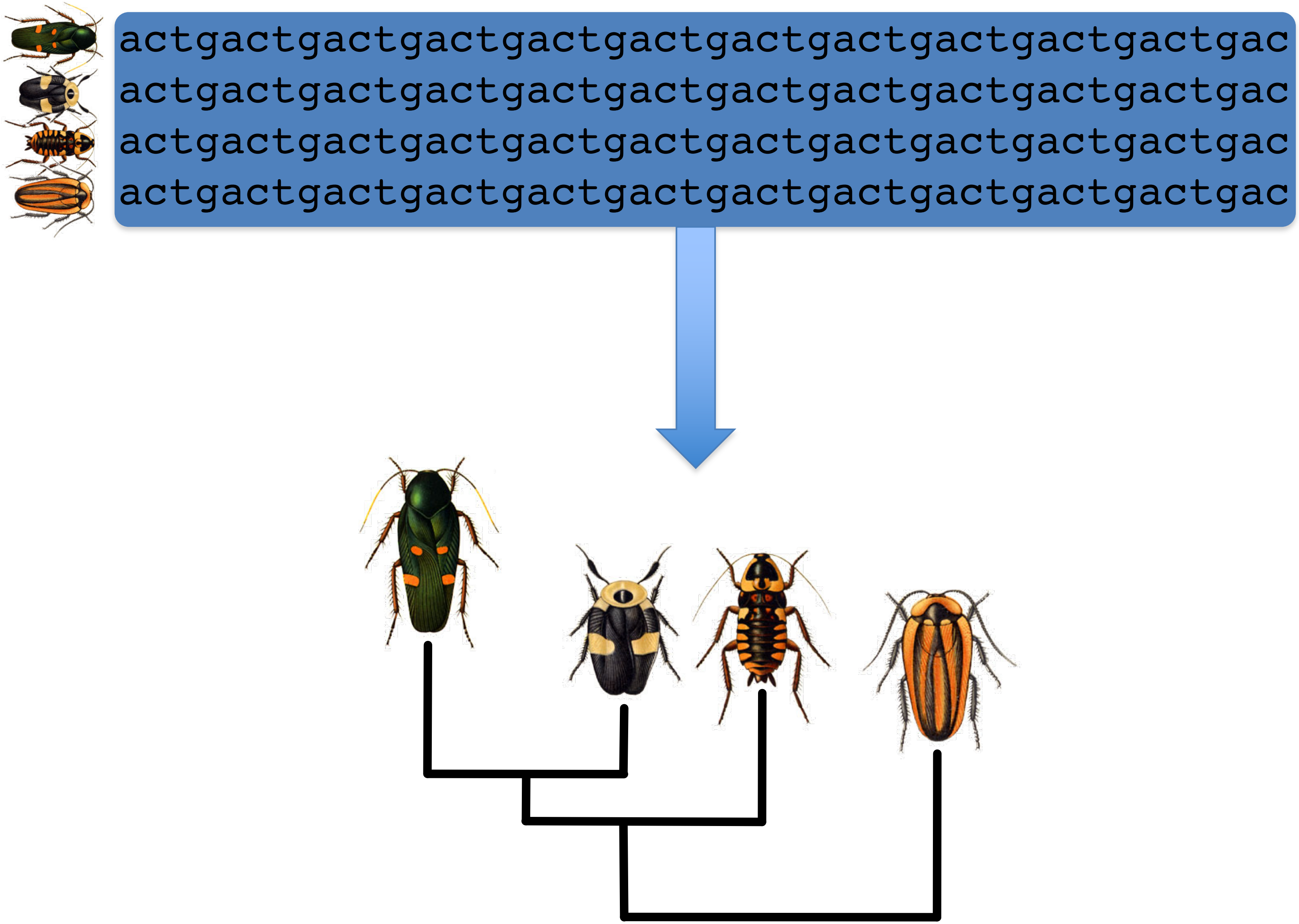
Ecology & Evolution

Australian National University



What is phylogenetics?

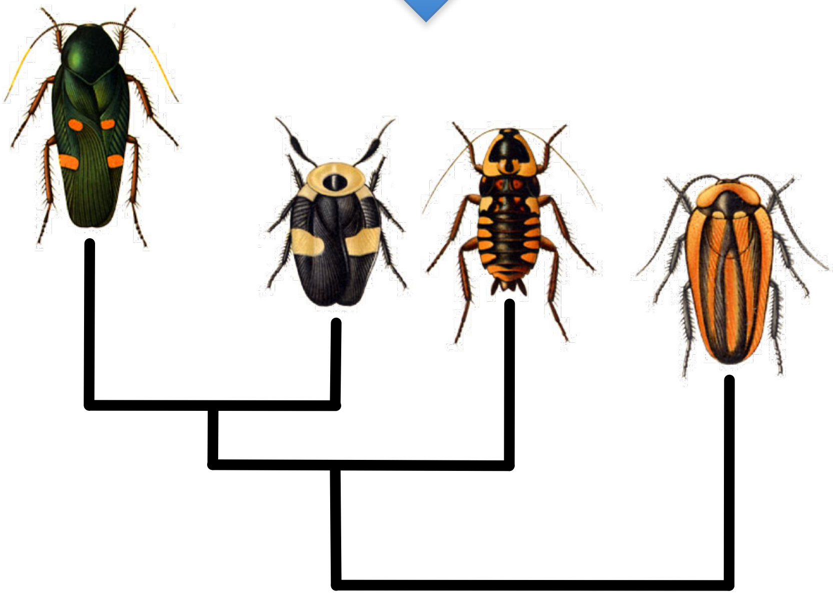
actgactgactgactgactgactgactgactgactgactgactgac
actgactgactgactgactgactgactgactgactgactgactgac
actgactgactgactgactgactgactgactgactgactgactgac
actgactgactgactgactgactgactgactgactgactgactgac

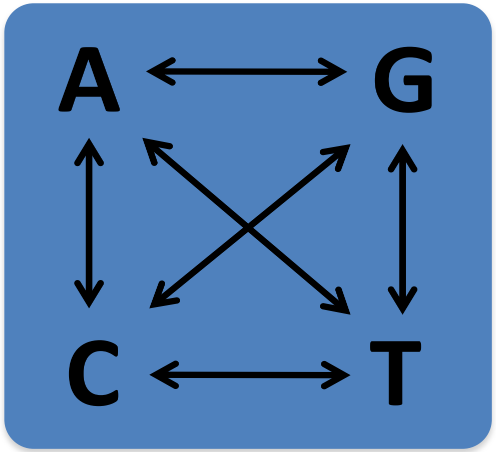


```
actgactgactgactgactgactgactgactgactgactgactgac
actgactgactgactgactgactgactgactgactgactgactgac
actgactgactgactgactgactgactgactgactgactgactgac
actgactgactgactgactgactgactgactgactgactgactgac
```

```
actgactgactgactgactgactgactgactgactgactgactgac
actgactgactgactgactgactgactgactgactgactgactgac
actgactgactgactgactgactgactgactgactgactgactgac
actgactgactgactgactgactgactgactgactgactgactgac
```

Molecular evolution

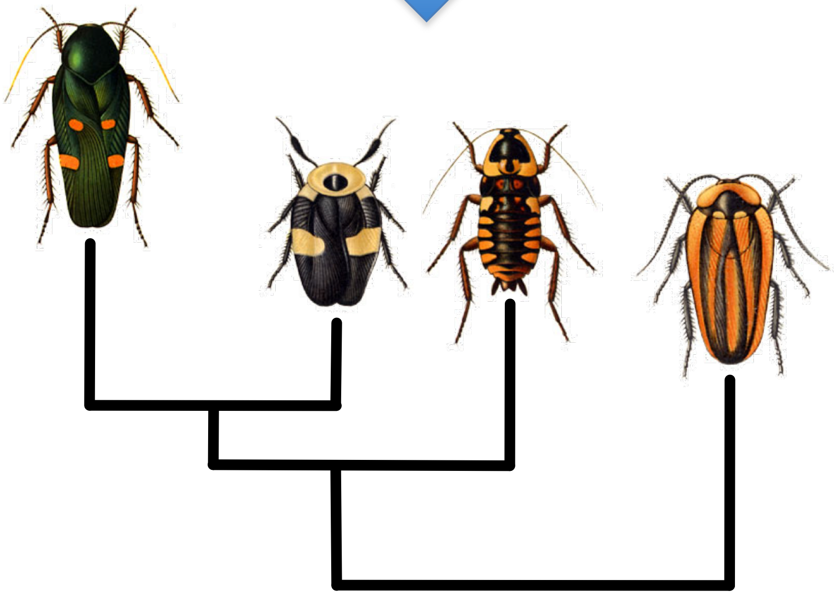


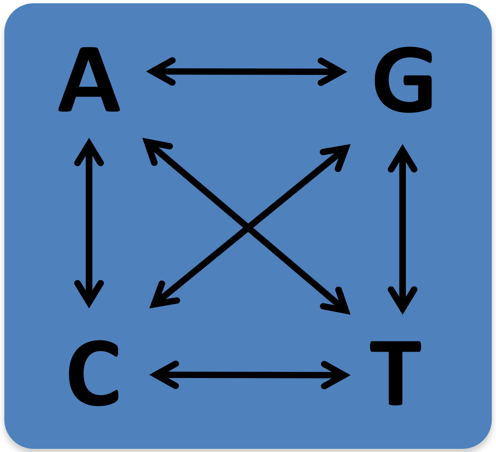


What is partitioning?

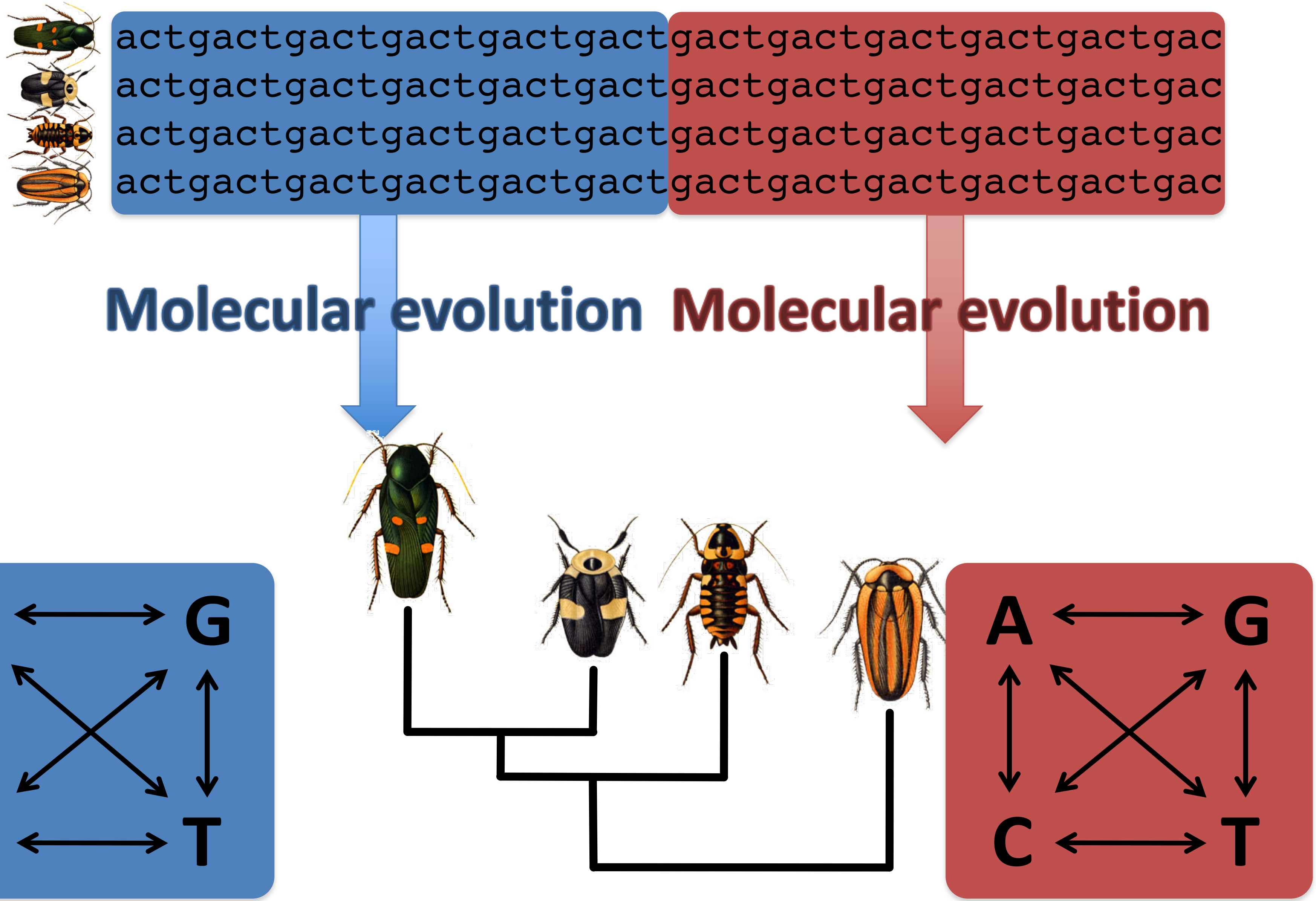
```
actgactgactgactgactgactgactgactgactgactgactgac
actgactgactgactgactgactgactgactgactgactgactgac
actgactgactgactgactgactgactgactgactgactgactgac
actgactgactgactgactgactgactgactgactgactgactgac
```

Molecular evolution





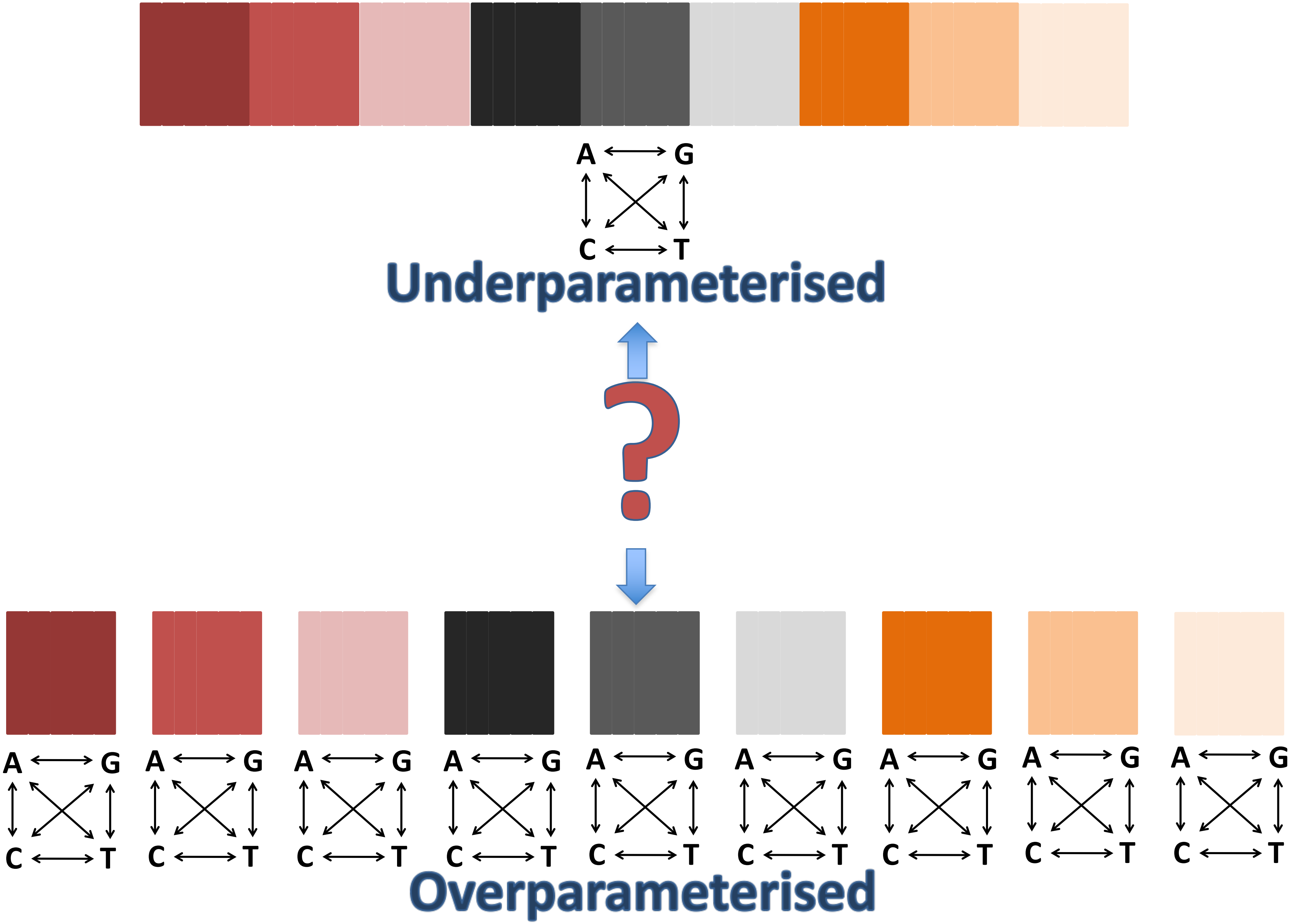
actgactgactgactgactgactgactgactgactgactgactgac
actgactgactgactgactgactgactgactgactgactgactgac
actgactgactgactgactgactgactgactgactgactgactgac
actgactgactgactgactgactgactgactgactgactgactgac



```
actgactgactgactgactgactgactgactgactgactgactgac
actgactgactgactgactgactgactgactgactgactgactgac
actgactgactgactgactgactgactgactgactgactgactgac
actgactgactgactgactgactgactgactgactgactgactgac
```

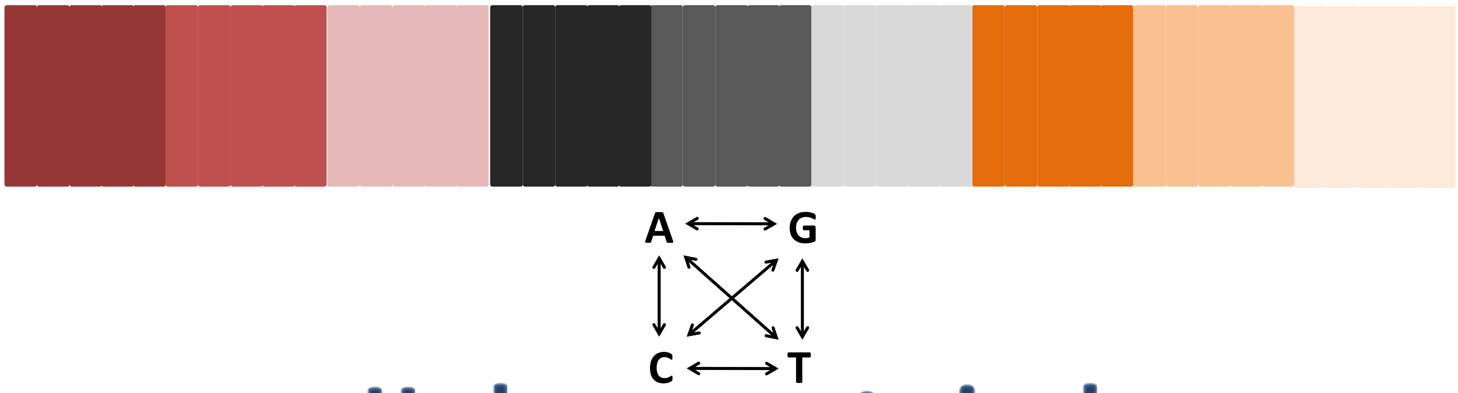
Partitioning can improve inference

So what’s the problem?



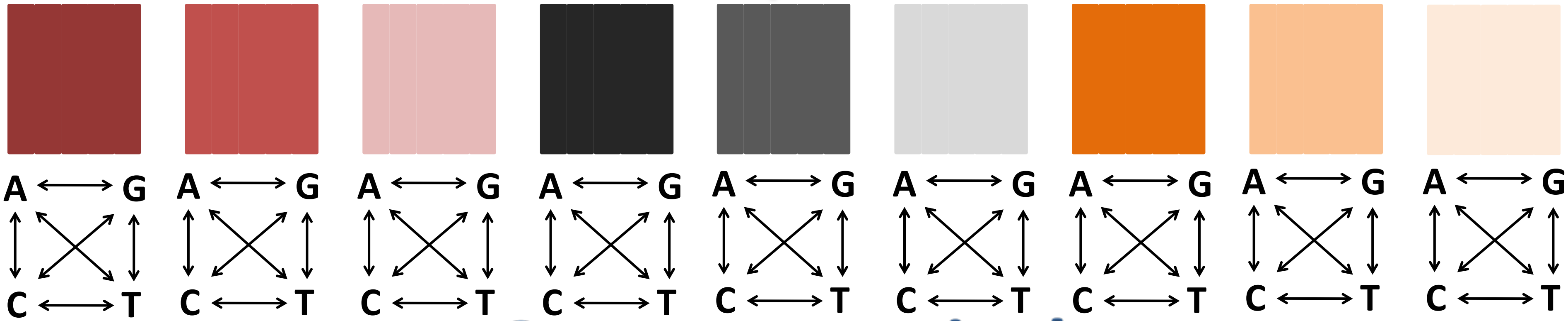
Underparameterised

Overparameterised



Underparameterised

21,147



Overparameterised

Let’s solve the problem

> “We should forget about small efficiencies, say about 97% of the time: **premature optimization is the root of all evil**”

— Donald Knuth

A Simple Algorithm

Compare all possible partitioning schemes.

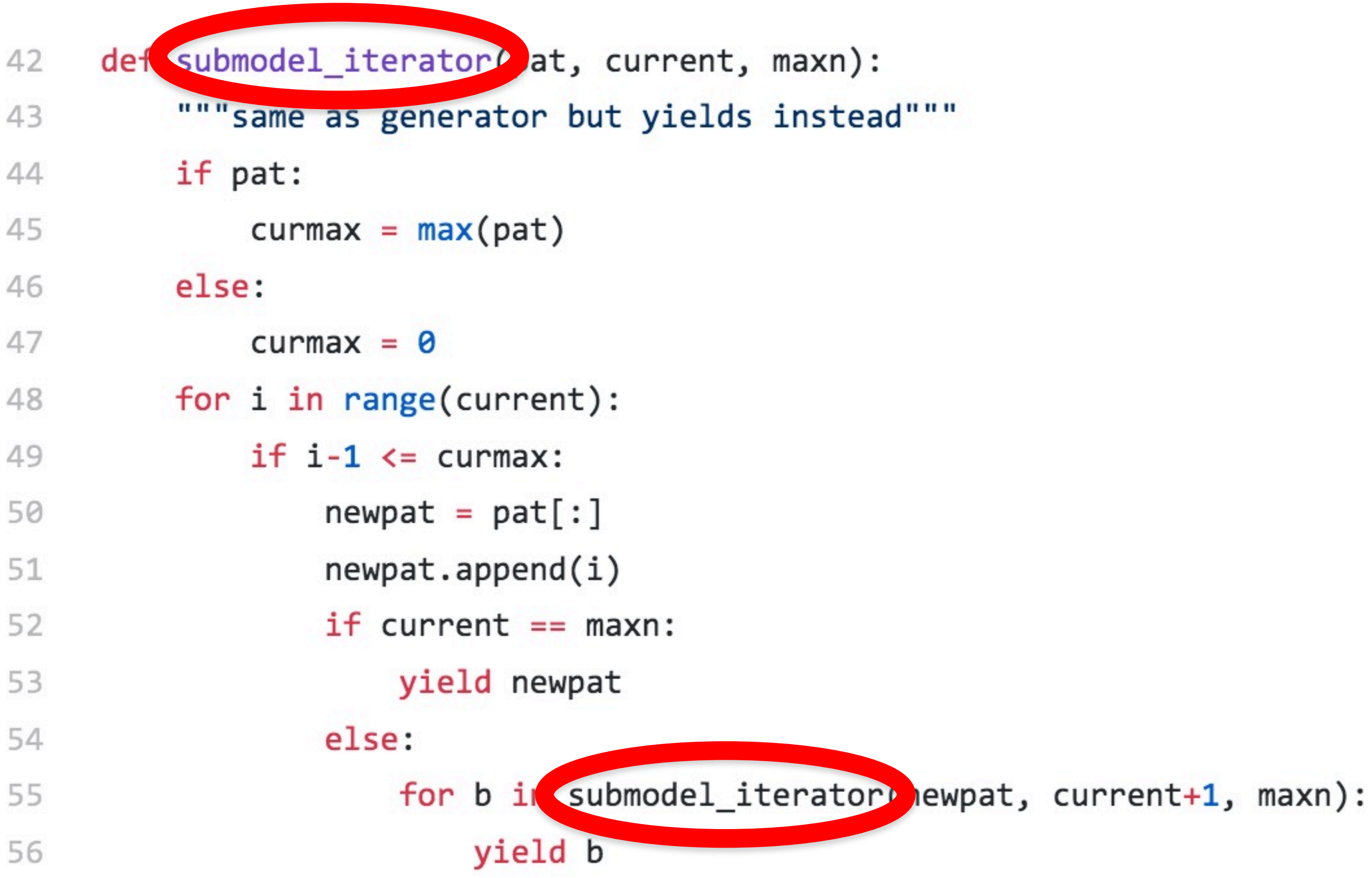
1. Calculate the AIC score of every possible partitioning scheme.

RECURSION

AIC (Akaike's Information Criterion) measures how far a model is from the truth.

2. Use the scheme with the smallest AIC score

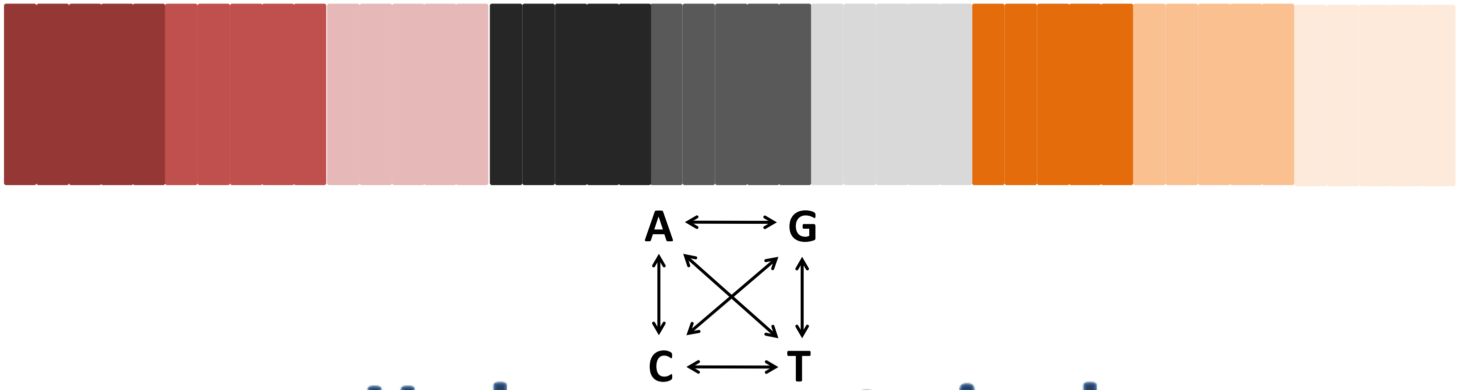
RECURSION



```
def submodel_iterator(pat, current, maxn):
    """same as generator but yields instead"""
    if pat:
        curmax = max(pat)
    else:
        curmax = 0
    for i in range(current):
        if i-1 <= curmax:
            newpat = pat[:]
            newpat.append(i)
            if current == maxn:
                yield newpat
            else:
                for b in submodel_iterator(newpat, current+1, maxn):
                    yield b
```

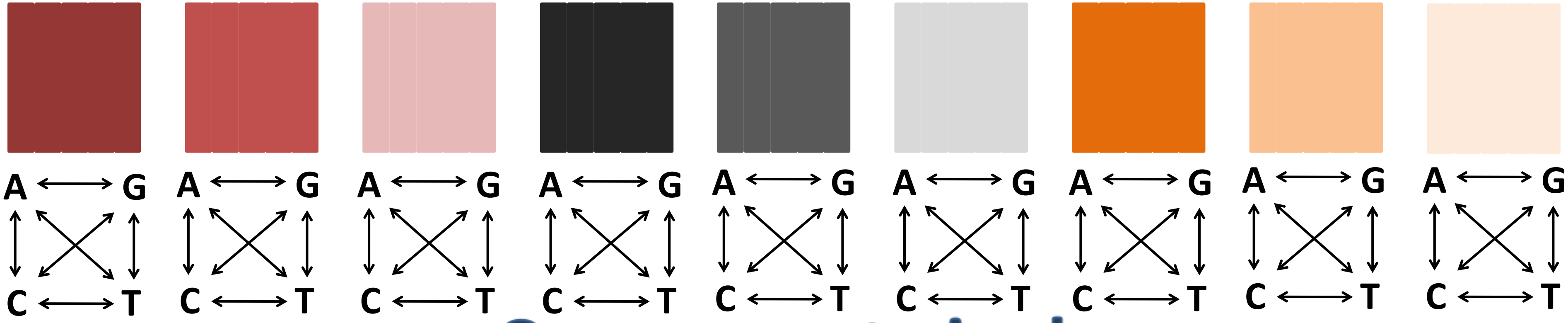
https://github.com/brettc/partitionfinder/blob/master/partfinder/submodels.py

Now we find a new problem…



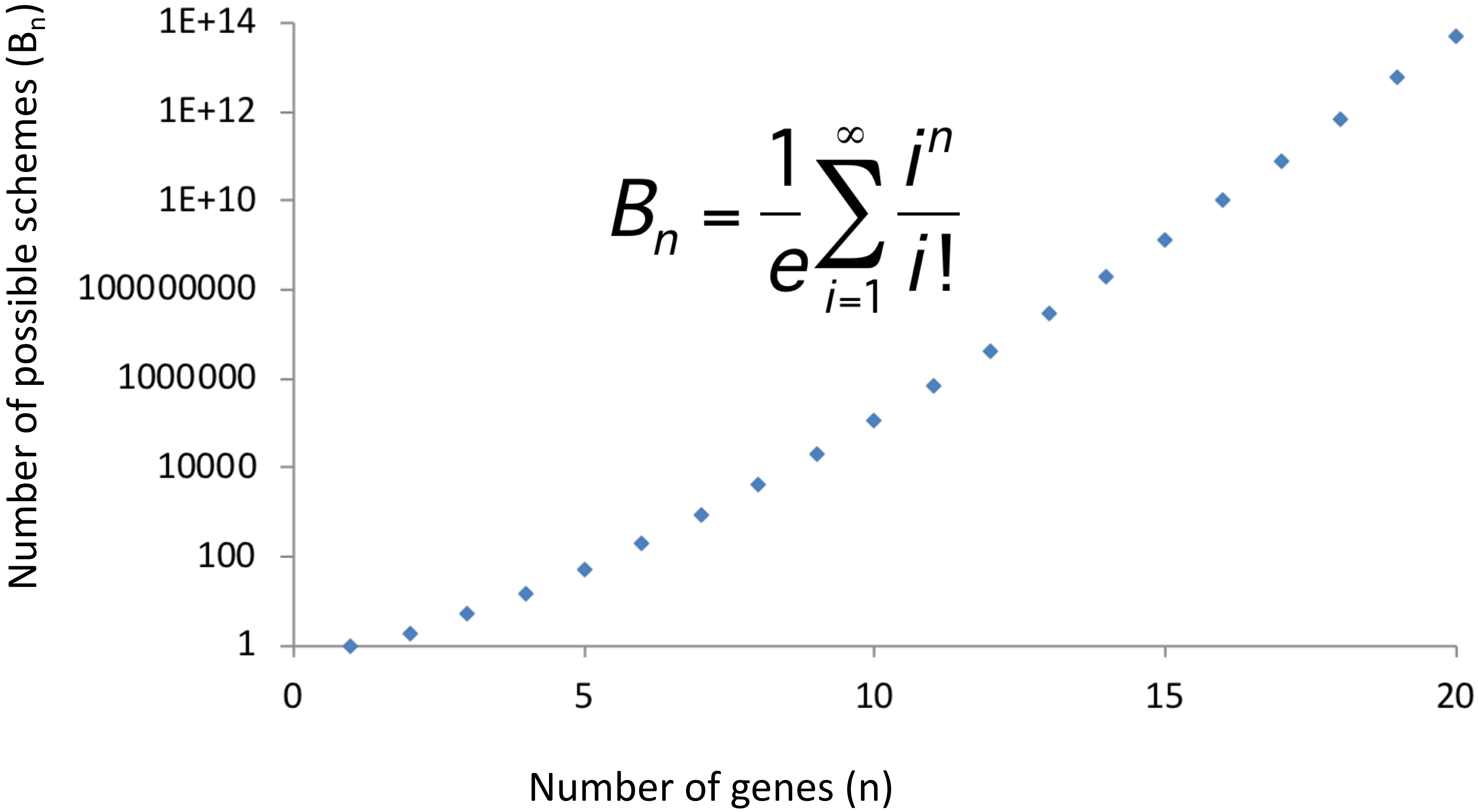
Underparameterised

21,147



Overparameterised

How many schemes are there?

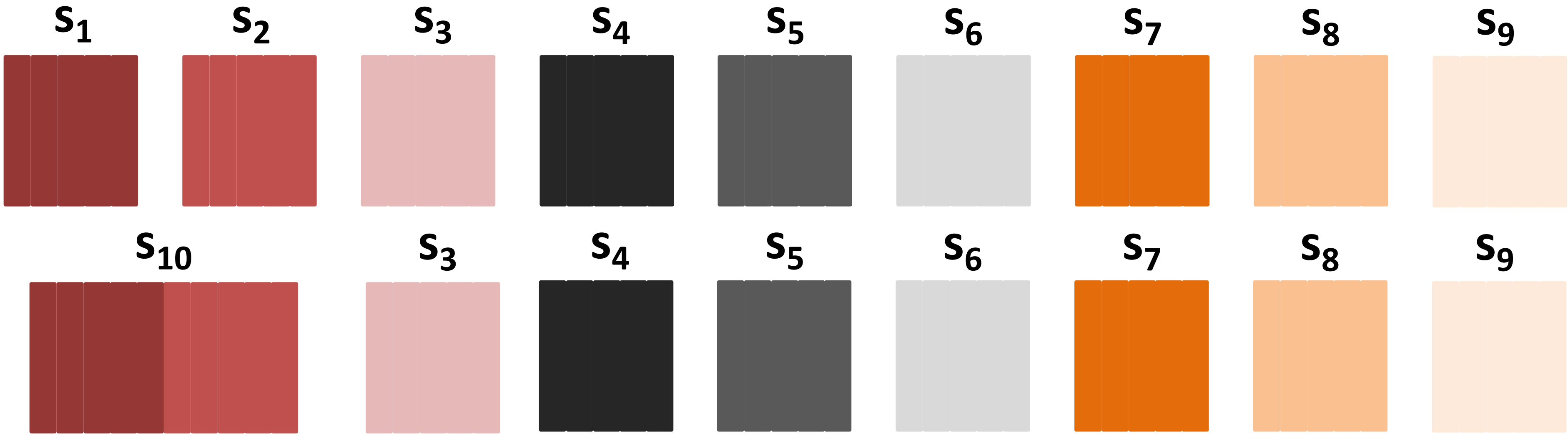


$$B_n = \frac{1}{e}\sum_{i=1}^{\infty}\frac{i^n}{i!}$$

Our first approach won’t work for large datasets.

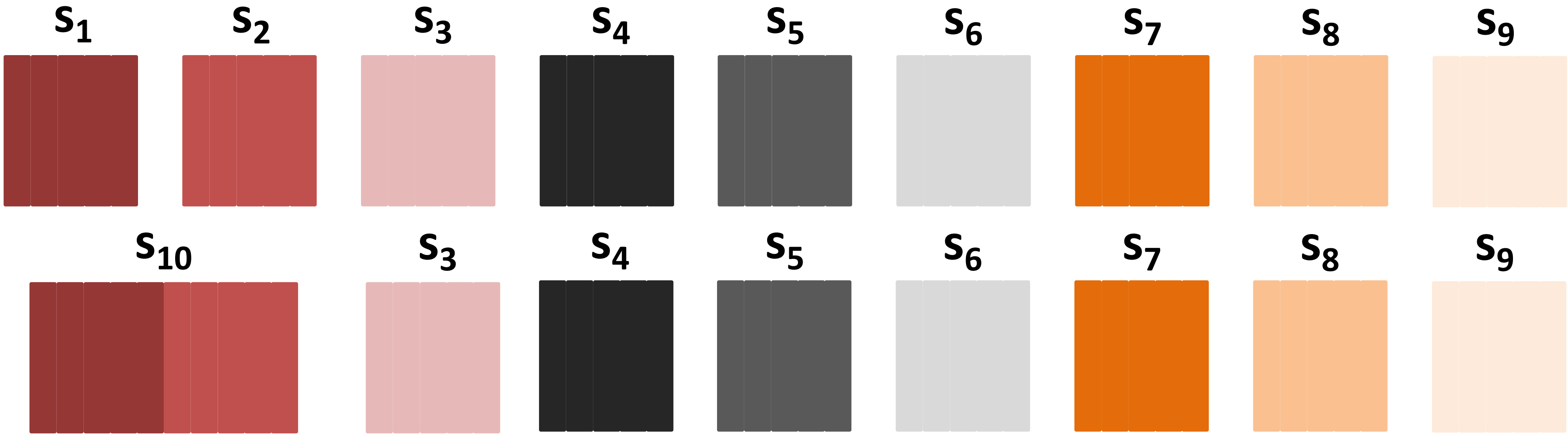
What can we do?

OPTMISE!



Analysing EVERY scheme EVERY time is inefficient
Because we analyse the same subsets over and over…

To calculate the AIC of all possible partitioning schemes,
we only need to calculate the AIC of all possible subsets of genes
Then we can just add these together to get the AIC of the schemes…



How do we count the number of unique subsets?

Combinatorics!

$$T_{subsets} = \sum_{i=1}^{n} C_i^n = 2^n - 1$$

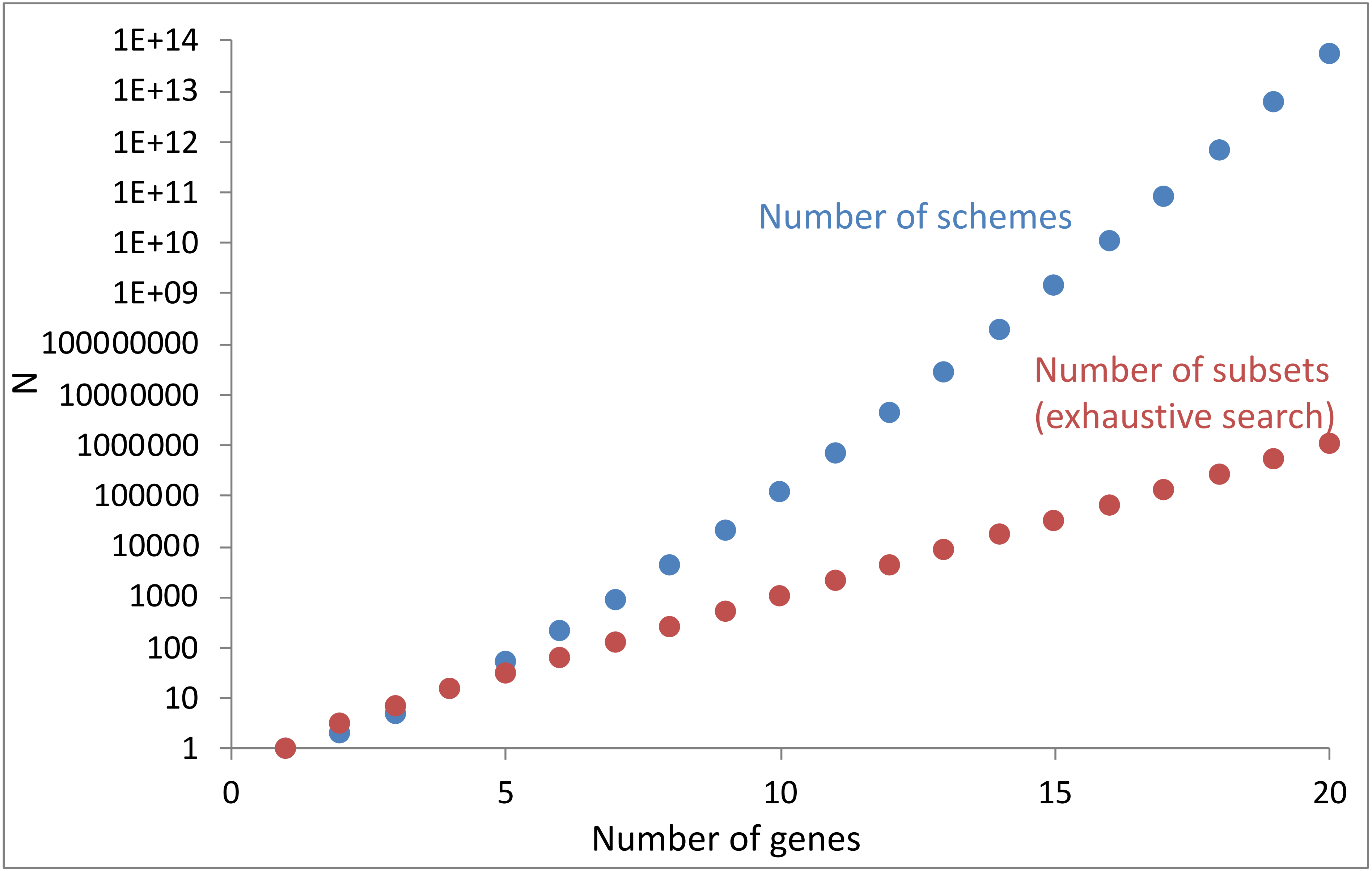
For example, if we start with 20 genes

There are **51,724,158,235,372** possible partitioning schemes

but these are made up of just 1,048,575 possible subsets

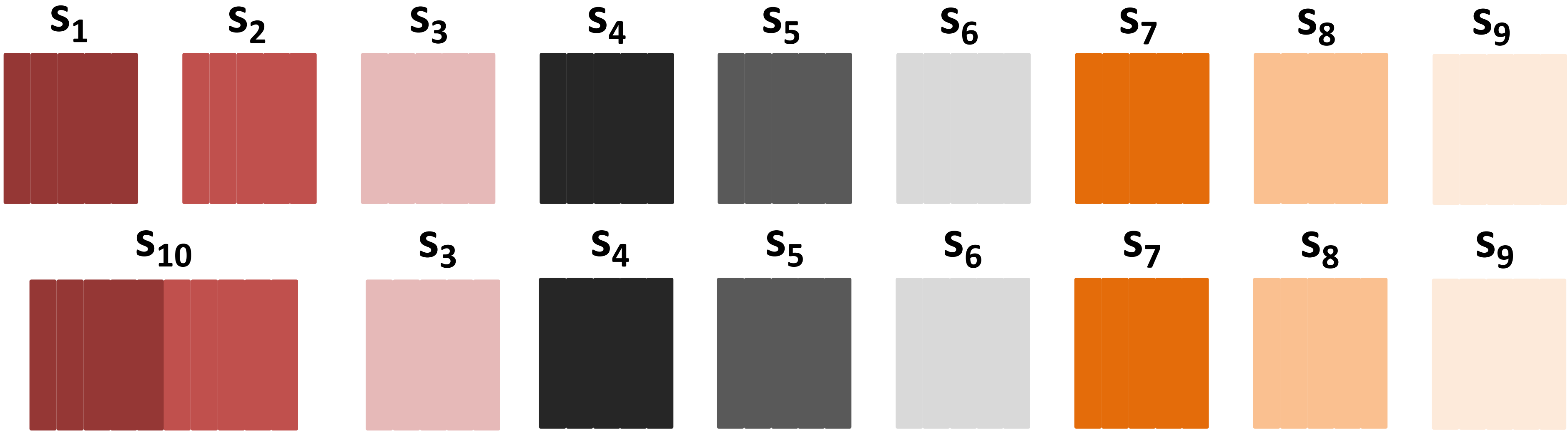
That's a time saving of 99.99998%!!!!!

How many subsets and schemes?



Converting schemes to subsets

- Using our recursive function from earlier, we can get all the schemes as a list of lists:



```
all_schemes = [[1, 2, 3, 4, 5, 6, 7, 8, 9],
               [10, 3, 4, 5, 6, 7, 8, 9]]
```

Converting schemes to subsets

- We want to convert our list of lists into a **set** of unique subsets, i.e.

```
all_schemes = [[1, 2, 3, 4, 5, 6, 7, 8, 9],
               [10, 3, 4, 5, 6, 7, 8, 9]]
```

↓

```
all_subsets = set([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
```

- Take 2 minutes and talk to your neighbour. Try to think of ways you would do this.
- **How do you think I did it?**

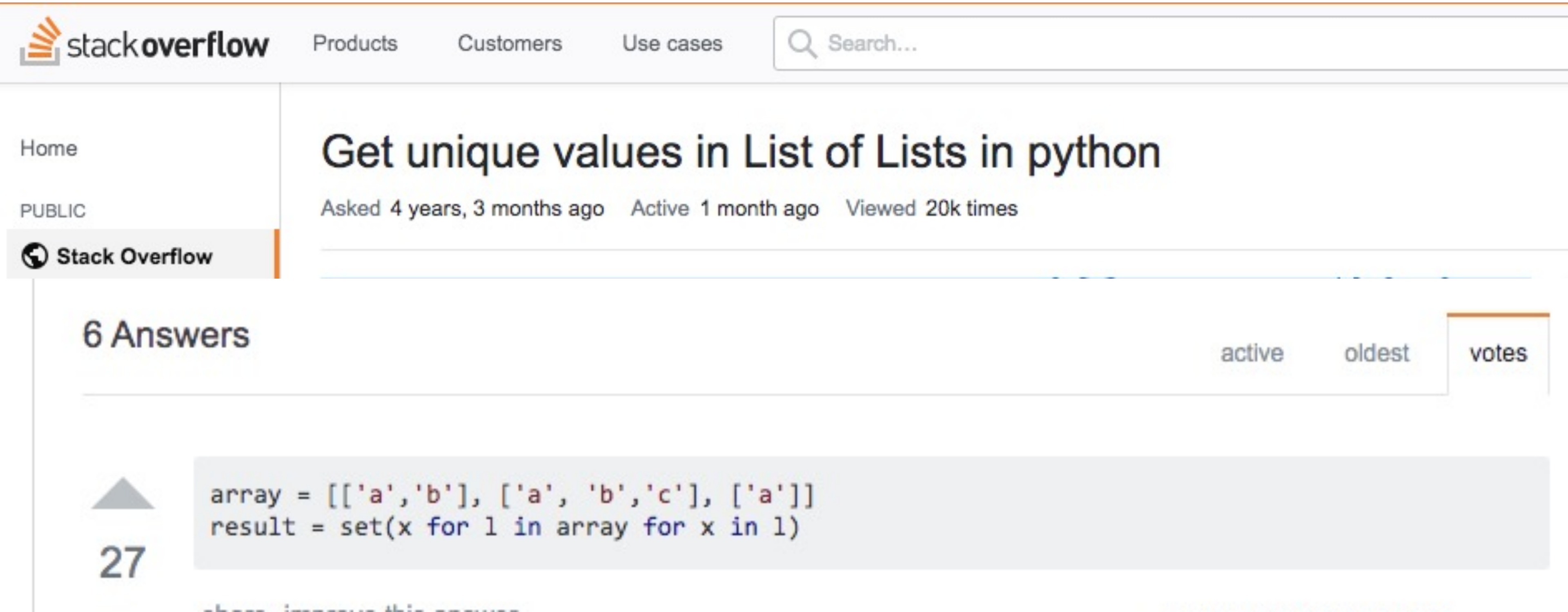
Converting schemes to subsets

- Lots of solutions…
- A simple one:

```
all_schemes = [[1, 2, 3, 4, 5, 6, 7, 8, 9],
               [10, 3, 4, 5, 6, 7, 8, 9]]

all_subsets = set([]) #empty set

for i in all_schemes: # loop through lists
   for j in i: # loop through each list

       all_subsets.add(j) # add value if not already there



>all_subsets
set([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
```

Converting schemes to subsets

- Lots of solutions…
- A better one:
  - Google "stack overflow python list of lists to set of unique values"



Get unique values in List of Lists in python

Asked 4 years, 3 months ago Active 1 month ago Viewed 20k times

6 Answers

27

```
array = [['a','b'], ['a', 'b','c'], ['a']]
result = set(x for l in array for x in l)
```

Real programmers use Google and Stack Overflow. A lot.

You don’t need to solve every problem from scratch.

Converting schemes to subsets

- Convert our list of lists into a **set** of unique subsets, i.e.

```
all_schemes = [[1, 2, 3, 4, 5, 6, 7, 8, 9],
               [10, 3, 4, 5, 6, 7, 8, 9]]
```

⇩

```
all_subsets = set([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
```

- Now we calculate the AIC of each subset and store it.
- **What data structure would you use?**

Summary

- We started with a naïve solution to analyse all partitioning schemes independently
- We only optimized this when we knew it wouldn’t work
- Optimisation takes many forms – but the key is to find the **biggest** inefficiencies and improve them.
- With a simple trick, we could speed up the code by millions of fold for large datasets!

Problem solved, right?

Wrong!

Today’s datasets can have 1000’s of gene.

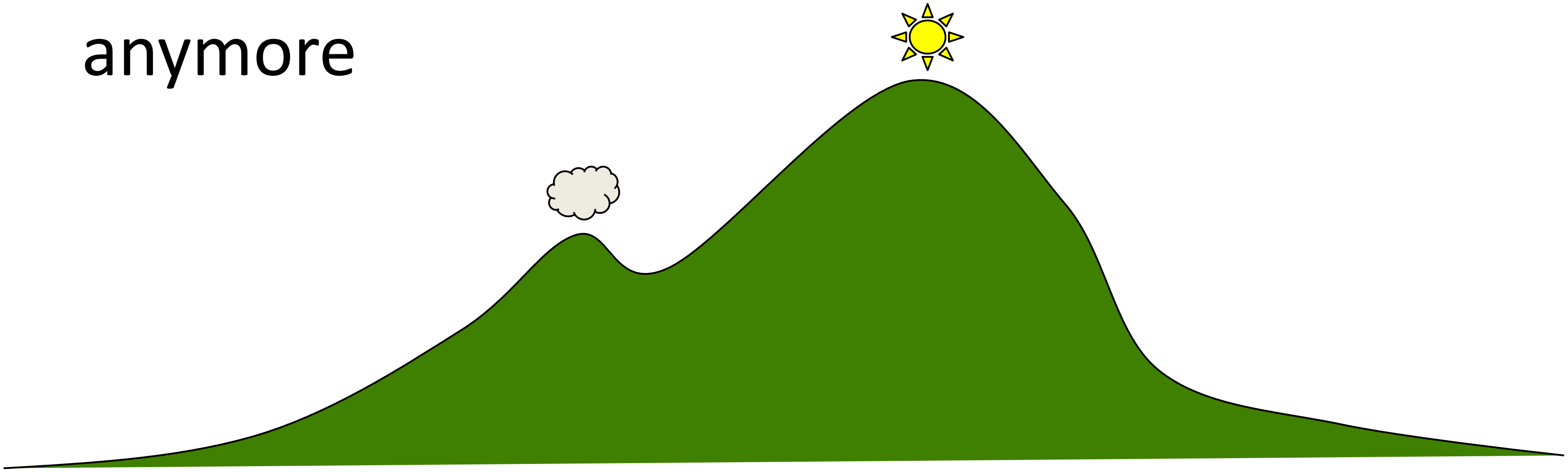
Even with our new algorithm that’s at least 1.07x10^{301} subsets to analyse…

A Solution

Heuristic search

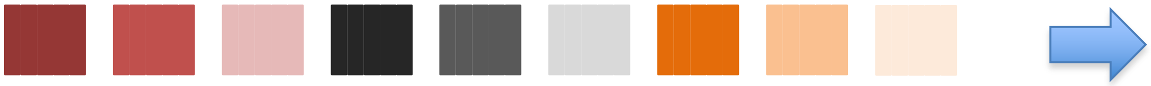
Heuristic search

1. Pick a starting partitioning scheme
2. Get the AIC
3. Try a few similar partitioning schemes
4. Get the best AIC score
5. Go to step 3
6. Stop when you can't improve the AIC anymore

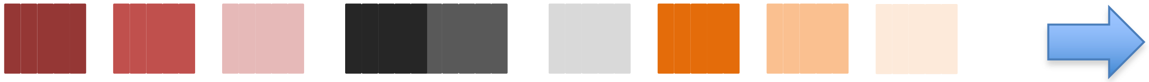


Greedy Algorithm

Subsets Examined

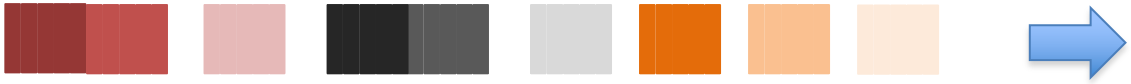


AIC_9 9



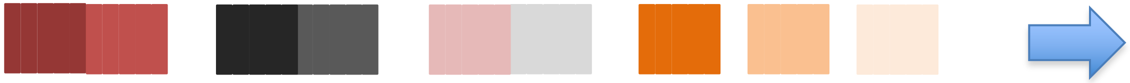
AIC_8 $C_2^9 = 36$

If: $AIC_8 < AIC_9$



AIC_7 7

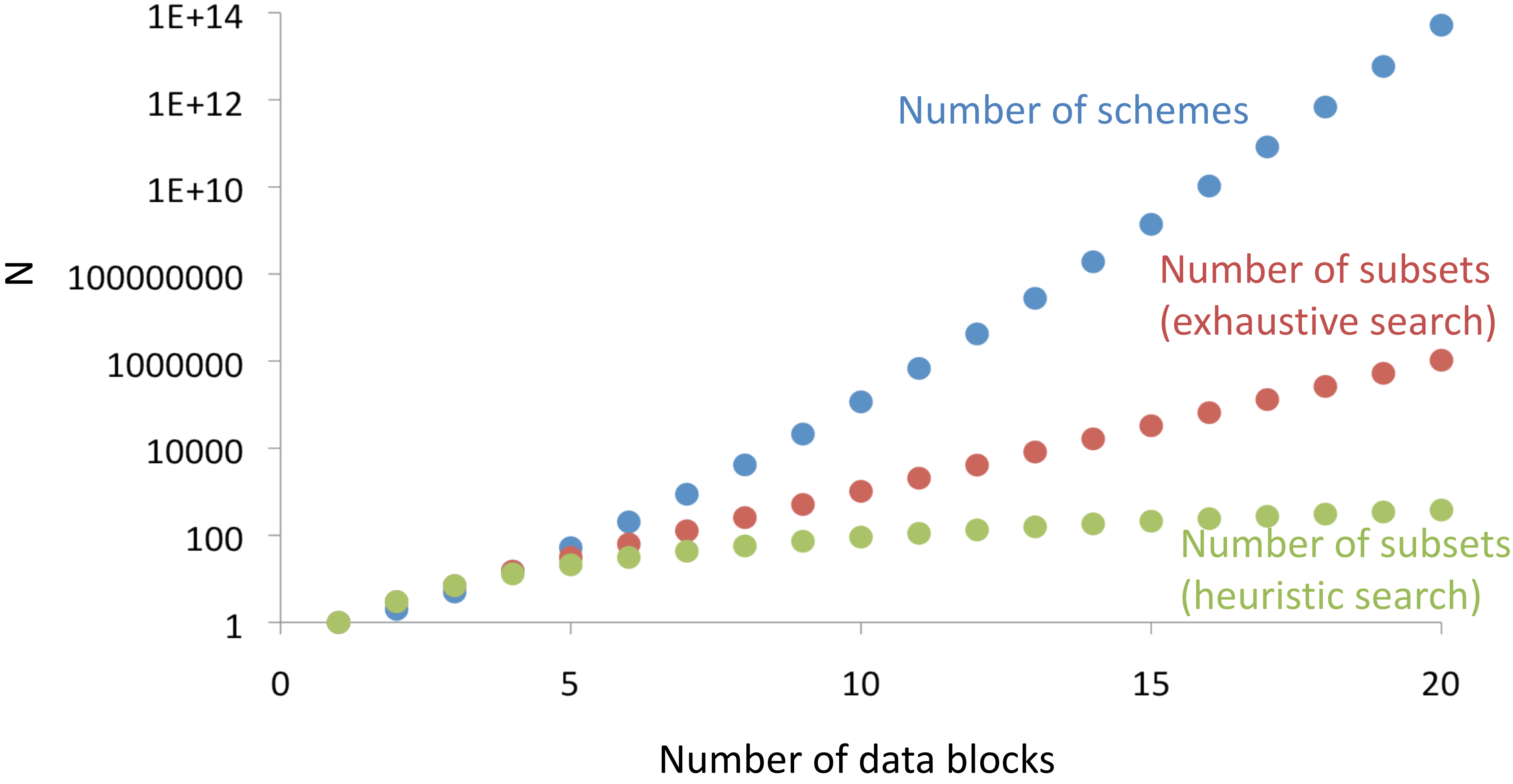
If: $AIC_7 < AIC_8$



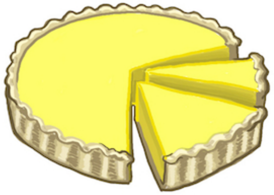
AIC_6 6

...

Efficient Heuristic Search



PartitionFinder: Combined Selection of Partitioning Schemes and Substitution Models for Phylogenetic Analyses



Robert Lanfear,*[,1] Brett Calcott,[1,2] Simon Y. W. Ho,[3] and Stephane Guindon[4]

- >100,000 downloads of the software
- >3000 citations of the paper
- Many follow up papers and algorithms, including upcoming work with Dr. Bui where we have algorithms 1000's of times faster than those I introduced today.

Take homes

- Start with simple, naïve solutions
  - Build something that works
- Avoid premature optimisation
- Use Google and Stack Overflow
- Go and look up:
  - Git and version control
  - E.g. https://github.com/brettc/partitionfinder
- Find a problem you're interested in.
- Start coding!