The end is near!

Mudd Summer Research Info Session

Is summer research the right move for me?
How do I know which lab to join?
How do I apply?

Come to the info session to learn about summer research at HMC!

Thursday Nov 18, 4:30-5:30 pm
The info session will be recorded for those who can't attend.

Sign up through Handshake, or use the Zoom info below.
Meeting ID: 994 5831 3034
Passcode: 728687

What's coming next…

- Tuesday: Projects showcase!
- After break
  - 11/30, 12/2 and 12/7 lecture in BECKMAN B126 (big Beckman)
  - Class material: The limits of computation!
  - 12/9 we're back "home" in Shan 2460 for a final lecture
  - Work on your project (milestone + final project)
  - Labs are just for working and getting help on projects (will be at normal time and place with the three of us)

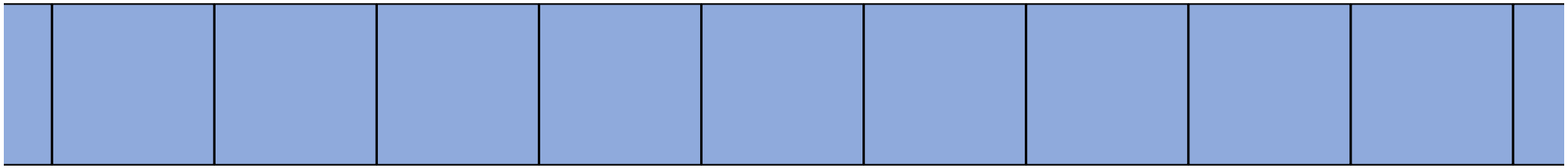
Why are dictionaries fast?

```
>>> D={}
>>> D['gene4242'] = ['gene21','gene5000','gene1987']
```

```
hash('gene4242')
```

```
4192264438798265428
```

Go to memory location based on hash



CS 5 Green

Plan for today

- Describe inheritance and its benefits
- An aside on Prof Bush's research, with examples of OOPs

Inheritance (inherit  s!)

```
class Person:
    def __init__(self, first_name, last_name):
        self.first_name = first_name
        self.last_name = last_name

    def __repr__(self):
        return self.first_name + " " + self.last_name

    def is_asleep(self, time):
        return 0 <= time <= 8   # using 24hr time format
```

```
>>> eliot = Person("Eliot", "Bush")
>>> eliot
Eliot Bush
>>> eliot.is_asleep(2)
True
```

```
class Person:
    def __init__(self, first_name, last_name):
        self.first_name = first_name
        self.last_name = last_name

    def __repr__(self):
        return self.first_name + " " + self.last_name

    def is_asleep(self, time):
        return 0 <= time <= 8

class Student(Person):
    def __init__(self, first_name, last_name, age):
        Person.__init__(self, first_name, last_name) # call base class constructor
        self.age = age

    def __repr__(self):
        return Person.__repr__(self) + ", " + str(self.age) + " years old"

    def is_asleep(self, time):
        return 3 <= time <= 11
```

Inheritance!
Read as: `Student` "is a" `Person`
`Person` is the **base class**.
`Student` is the **derived class**.
`Student` **inherits** all attributes and methods from `Person`.

Function Overriding!

Sleeping until 11 AM!?

```
>>> s = Student("Sue", "Persmart", 18)
>>> s
Sue Persmart, 18 years old
>>> s.is_asleep(2)
False
```

```
class Person:
    def __init__(self, first_name, last_name):
        self.first_name = first_name
        self.last_name = last_name

    def __repr__(self):
        return self.first_name + " " + self.last_name

    def is_asleep(self, time):
        return 0 <= time <= 8

class Student(Person):
    def __init__(self, first_name, last_name, age):
        Person.__init__(self, first_name, last_name)
        self.age = age

    def __repr__(self):
        return Person.__repr__(self) + ", " + \
               str(self.age) + " years old"

    def is_asleep(self, time):
        return 3 <= time <= 11

class Mudder(Student):
    def __init__(self, first_name, last_name, age, dorm):
        Student.__init__(self, first_name, last_name, age)
        self.dorm = dorm

    def is_asleep(self, time):
        return False
```

```
>>> wally = Mudder("wally", "wart",
                   42, "west")
>>> wally
?
>>> wally.is_asleep(10)
?
```

Objects are `self`-ish

```
class Person:
    ...
```

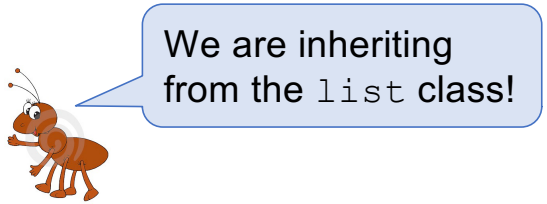
```
def rename(self,
           new_first_name,
           new_last_name):
    self.first_name = new_first_name
    self.last_name = new_last_name
```

```
def rename(self,
           new_first_name,
           new_last_name):
    self = Person(new_first_name,
                  new_last_name)
```

Polite Lists

```
>>> p = polite()
Thank you for using polite lists!
>>> p.append(42)
>>> p.append(50)
>>> p[0] += 1
>>> p
This polite list contains [43, 50]
>>> p[3] = 100
Pardon me, but your index is out of bounds
>>> p[3]
Pardon me, but your index is out of bounds
```

Polite Lists




```
class polite(list):
    def __init__(self):
        print("Thank you for using polite lists!")

    def __repr__(self):
        return "This polite list contains " + list.__repr__(self)

    def __setitem__(self, index, value):
        if index >= len(self):
            print("Pardon me, but your index is out of bounds")
        else:
            list.__setitem__(self, index, value)

    def __getitem__(self, index):
        if index >= len(self):
            print("Pardon me, but your index is out of bounds")
        else:
            return list.__getitem__(self, index)
```


```
class polite(list):
    def __init__(self):
        print("Thank you for using polite lists!")

    def __repr__(self):
        return "This polite list contains " + list.__repr__(self)

    def __setitem__(self, index, value):
        if index >= len(self):
            print("Pardon me, but your index is out of bounds")
        else:
            list.__setitem__(self, index, value)

    def __getitem__(self, index):
        if index >= len(self):
            print("Pardon me, but your index is out of bounds")
        else:
            return list.__getitem__(self, index)
```

```
>>> L = polite()
Thank you for using polite lists!
>>> L.append(42)
>>> L.append(47)
>>> L

>>> M = L + 5

>>> M

>>> N = M + [50]

>>> N
```

What does this do?

```
def __add__(self, other):
    output = polite()
    for x in self:
        output.append(x)
    if isinstance(other, int):
        output.append(other)
    elif isinstance(other, list):
        output.extend(other)
    else:
        print("Pardon me, but your other is confusing")
    return output
```

```
class polite(list):
    def __init__(self):
        print("Thank you for using polite lists!")

    def __repr__(self):
        return "This polite list contains " + list.__repr__(self)

    def __setitem__(self, index, value):
        if index >= len(self):
            print("Pardon me, but your index is out of bounds")
        else:
            list.__setitem__(self, index, value)

    def __getitem__(self, index):
        if index >= len(self):
            print("Pardon me, but your index is out of bounds")
        else:
            return list.__getitem__(self, index)
```


```
>>> L = polite()
Thank you for using polite lists!
>>> L.append(42)
>>> L.append(47)
>>> L
This polite list contains [42, 47]
>>> M = L + 5
Thank you for using polite lists!
>>> M
This polite list contains [42, 47, 5]
>>> N = M + [50]
Thank you for using polite lists!
>>> N
This polite list contains [42, 47, 5, 50]
```

What does this do?

```
def __add__(self, other):
    output = polite()
    for x in self:
        output.append(x)
    if isinstance(other, int):
        output.append(other)
    elif isinstance(other, list):
        output.extend(other)
    else:
        print("Pardon me, but your other is confusing")
    return output
```

Worksheet


```
class polite(list):
    def __init__(self):
        print("Thank you for using polite lists!")

    def __repr__(self):
        return "This polite list contains " + list.__repr__(self)

    def __setitem__(self, index, value):
        if index >= len(self):
            print("Pardon me, but your index is out of bounds")
        else:
            list.__setitem__(self, index, value)

    def __getitem__(self, index):
        if index >= len(self):
            print("Pardon me, but your index is out of bounds")
        else:
            return list.__getitem__(self, index)
```

Complete the function

```
def __mul__(self, other):
```

```
>>> L = polite()
Thank you for using polite lists!
>>> L.append(42)
>>> L.append(47)
>>> L.append(23)
>>> M = polite()
Thank you for using polite lists!
>>> M.append(5)
>>> M.append(6)
>>> N = L * M  # same as N = L.__mul__(M)
Thank you for using polite lists!
>>> N
This polite list contains [(42, 5), (42,
6), (47, 5), (47, 6), (23, 5), (23, 6)]
```

Worksheet


```
class polite(list):
    def __init__(self):
        print("Thank you for using polite lists!")

    def __repr__(self):
        return "This polite list contains " + list.__repr__(self)

    def __setitem__(self, index, value):
        if index >= len(self):
            print("Pardon me, but your index is out of bounds")
        else:
            list.__setitem__(self, index, value)

    def __getitem__(self, index):
        if index >= len(self):
            print("Pardon me, but your index is out of bounds")
        else:
            return list.__getitem__(self, index)
```

Complete the function

```
def __mul__(self, other):
    output = polite()
    for x in self:
        for y in other:
            output.append((x,y))
    return output
```

```
>>> L = polite()
Thank you for using polite lists!
>>> L.append(42)
>>> L.append(47)
>>> L.append(23)
>>> M = polite()
Thank you for using polite lists!
>>> M.append(5)
>>> M.append(6)
>>> N = L * M  # same as N = L.__mul__(M)
Thank you for using polite lists!
>>> N
This polite list contains [(42, 5), (42,
6), (47, 5), (47, 6), (23, 5), (23, 6)]
```

```
class polite(list):
    def __init__(self):
        print("Thank you for using polite lists!")

    def __repr__(self):
        return "This list contains " + list.__repr__(self)

    def __setitem__(self, index, value):
        if index >= len(self):
            print("Pardon me, but your index is out of bounds")
        else:
            list.__setitem__(self, index, value)

    def __getitem__(self, index):
        if index >= len(self):
            print("Pardon me, but your index is out of bounds")
        else:
            return list.__getitem__(self, index)
```

What needs to be done to do this?

```
>>> L = polite([1, 2, 3, 4])
Thank you for using polite
lists!
>>> L
This polite list contains [1,
2, 3, 4]
>>> 3 in L
True
```

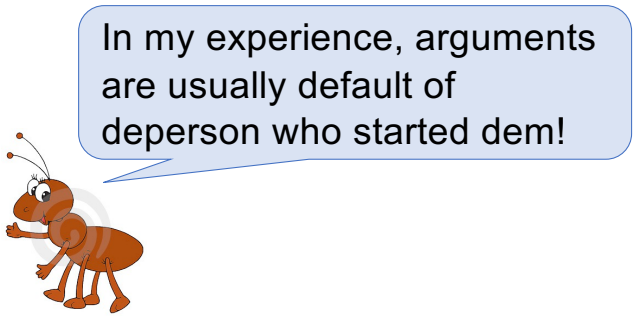
```
class polite(list):
    def __init__(self):
        print("Thank you for using polite lists!")

    def __repr__(self):
        return "This list contains " + list.__repr__(self)

    def __setitem__(self, index, value):
        if index >= len(self):
            print("Pardon me, but your index is out of bounds")
        else:
            list.__setitem__(self, index, value)

    def __getitem__(self, index):
        if index >= len(self):
            print("Pardon me, but your index is out of bounds")
        else:
            return list.__getitem__(self, index)
```


```
>>> L = polite([1, 2, 3, 4])
Thank you for using polite
lists!
>>> L
This polite list contains [1,
2, 3, 4]
>>> 3 in L
True
```

What needs to be done to do this?

```
def __init__(self, lst=[]):
    print("Thank you for using polite lists!")
    list.__init__(self, lst)

# nothing needs to be done
# polite is a list
# list knows in (__contains__)
```

Default Argum*ants*




```
class Person:
    def __init__(self, ...):
    def __repr__(self):
    def is_asleep(self):

class Student(Person):
    def __init__(self,
                           first_name,
                                  last_name,
                                  school = "Claremont Colleges",
                                  major  = "undeclared"):
        self.first_name = first_name
        self.last_name = last_name
        self.college = school
        self.major = major
```

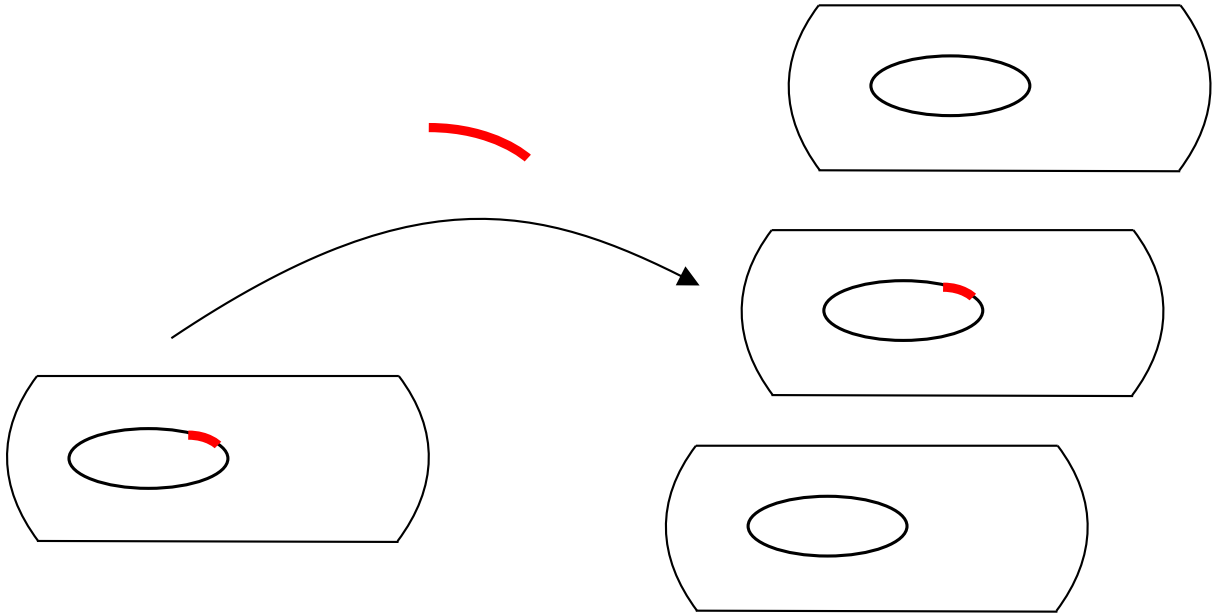
```
>>> wally = Student("Wally", "Wart", "Harvey Mudd")
>>> cecil = Student("Cecil", "Sagehen", "Pomona", "Biology and Mathematics")
>>> la_semeuse = Student("La", "Semeuse", school="Scripps")
>>> sammy = Student("Sammy", "the Owl", major="Engineering")
>>> elmo = Student("Elmo")
```

CS 5 Green

Plan for today

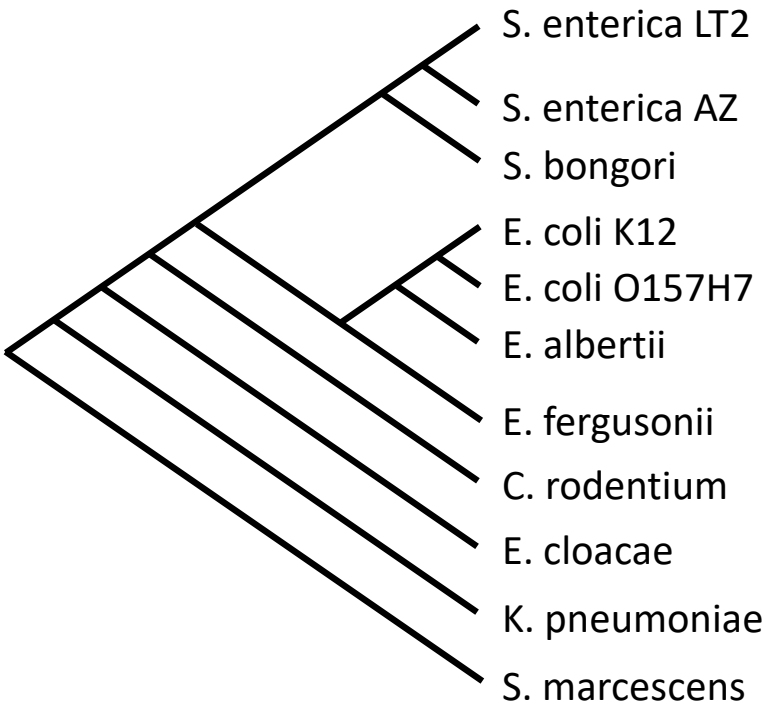
- Describe inheritance and its benefits
- An aside on Prof Bush's research, with examples of OOPs

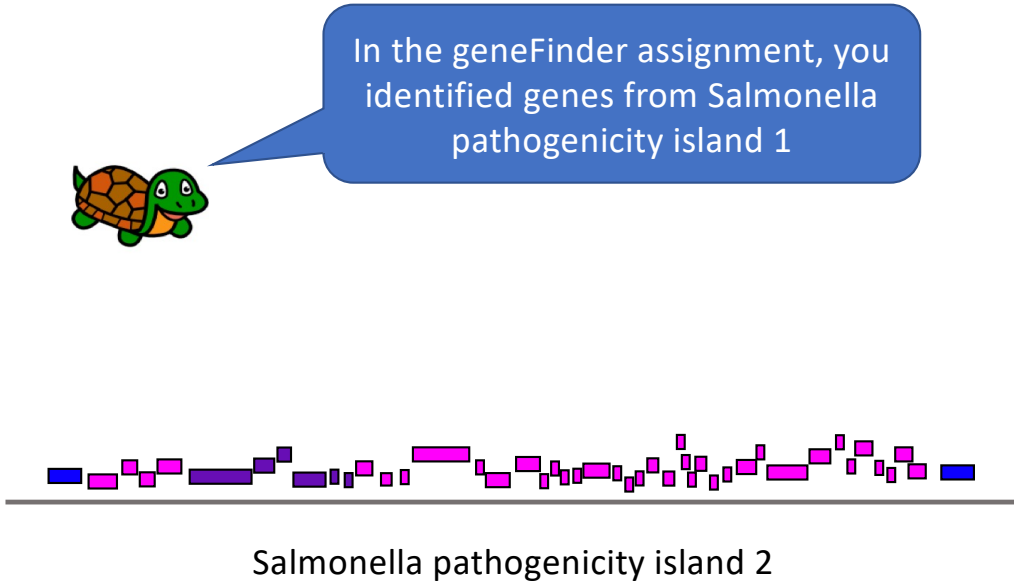
Horizontal transfer plays a large role in microbial genome evolution



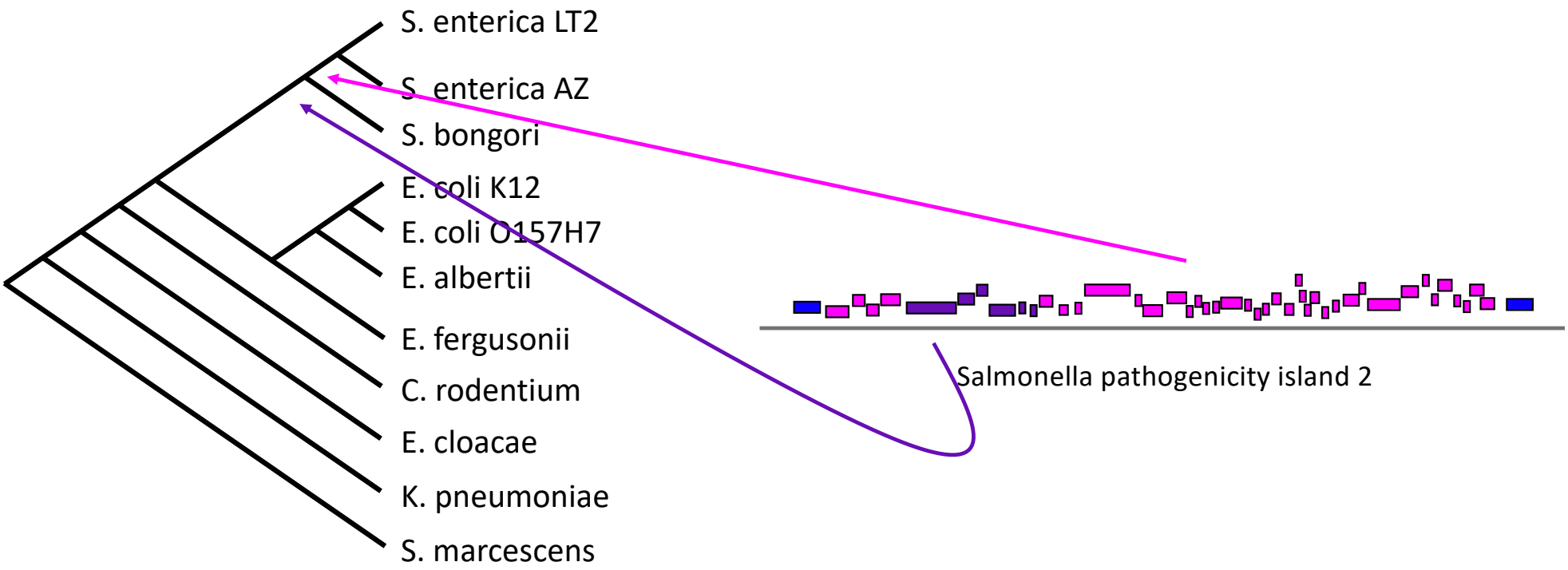
Genomic island

The problem: placing genomic islands in their phylogenetic context





The problem: placing genomic islands in their phylogenetic context

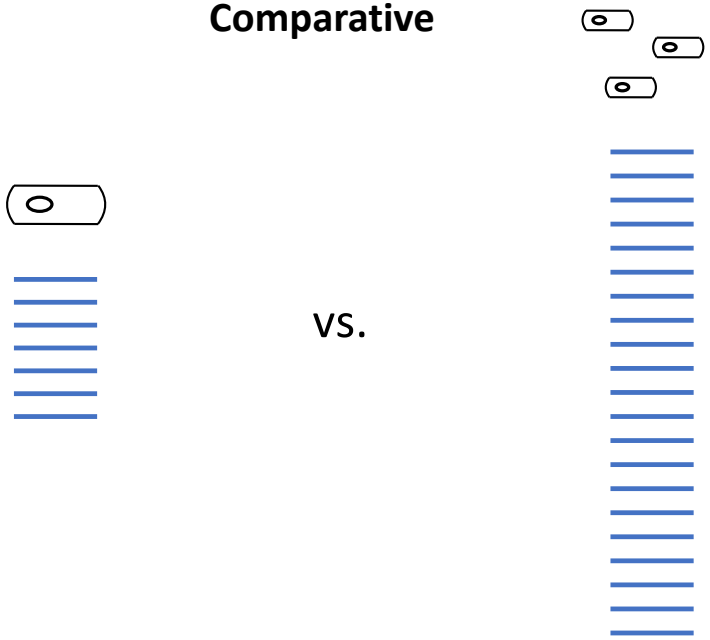


Previously existing automated methods don't identify islands in their phylogenetic context

Sequence composition based

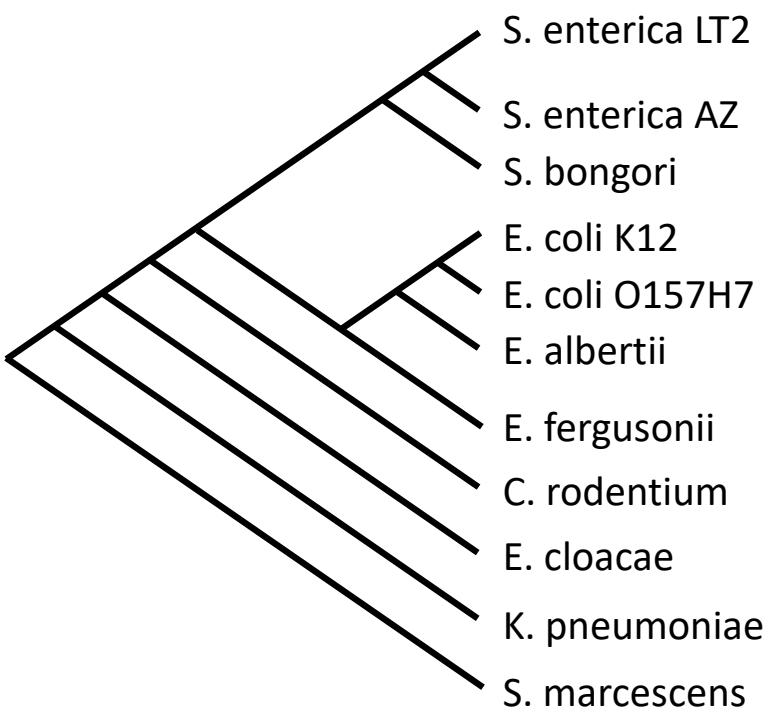
```
GTGGTAACGTTGGTTAGTGGTTGAAAGTAAACCCGGTGGCA
GCAGCACATGAACAAGTTTCGTCAGGTTAATAAATCAAATA
TTTATTGACTTAGGTCACCAGATACTTTAACCAATAGAGAC
ACACAGCACAGACAGATAAAAATTACAGAGTACACAACATC
CATGAAACGCATCAGCACCACCATTATTACCACCACCACCA
TTACCACAGGTAACGGTGCGGGCTGACGCGTACAGGAAATA
CAGAAAAAAGCCCGCACCTGAACAGTGCGGGCTTTTTTTTC
GACCAAAGTTAACGAGGTAACAACCATGCGAGTGTTGAAGT
TCGGCGGTACGTCAGTGGCAAATGCAGAACGGTTTCTGCGG
GTTGCCGATATTCTGGAAAGCAATGCCAGGCAGGGGCAGGT
AGCGACCGTACTTTCTGCCCCCGCAAAAATTACCAACCACC
TGGTGGCGATGATTGAAAAGACTATCGGC...
```

Comparative



vs.

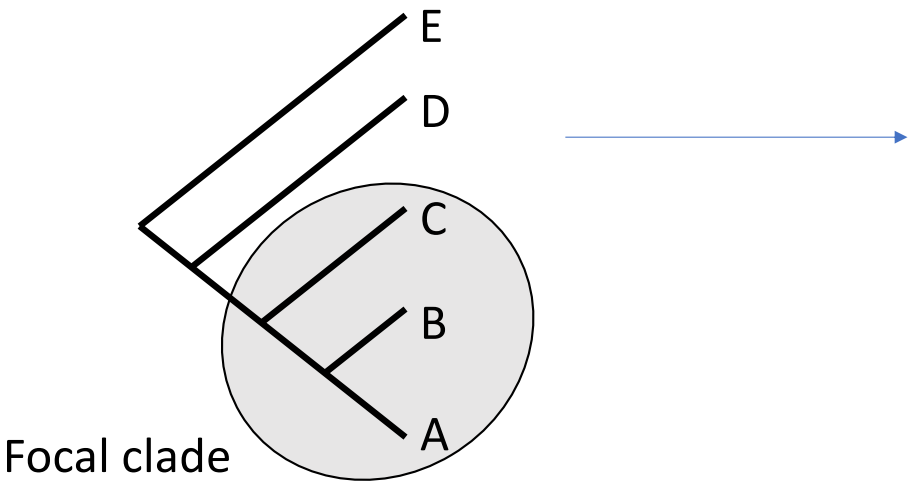
xenoGI: an automated approach to reconstructing the history of GI insertions into a clade



- Determine which genes were present in the MRCA, and which arrived via horizontal transfer
- Group genes with common origin into islands
- Map island insertion events onto the phylogenetic tree

xenoGI: inputs and outputs

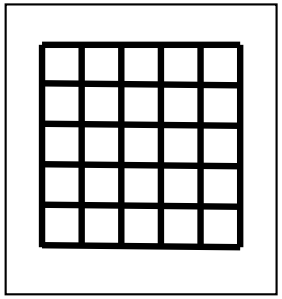
- A set of genomes in genbank format
- A phylogenetic tree for the species



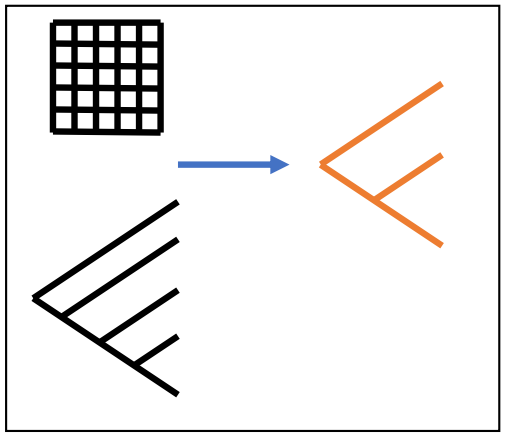
- Text output specifying the origin of each gene
- Bed files to visualize islands in a genome browser

xenoGI: the general approach

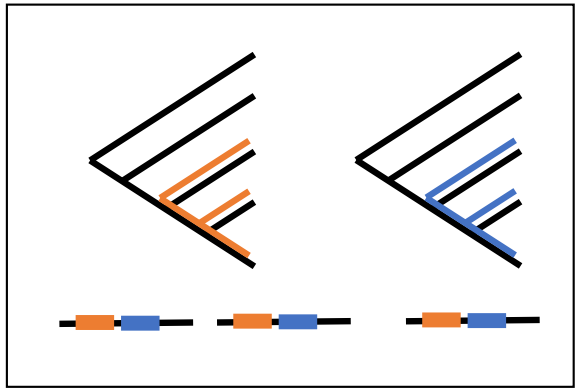
Calculate similarity scores



Form gene families

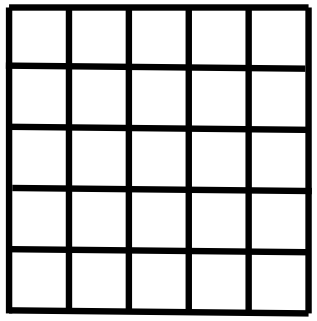


Group families into islands



Calculate similarity scores between genes

- All vs. all protein blast
- Global-alignment-based protein similarity score (raw score)
- Synteny scores

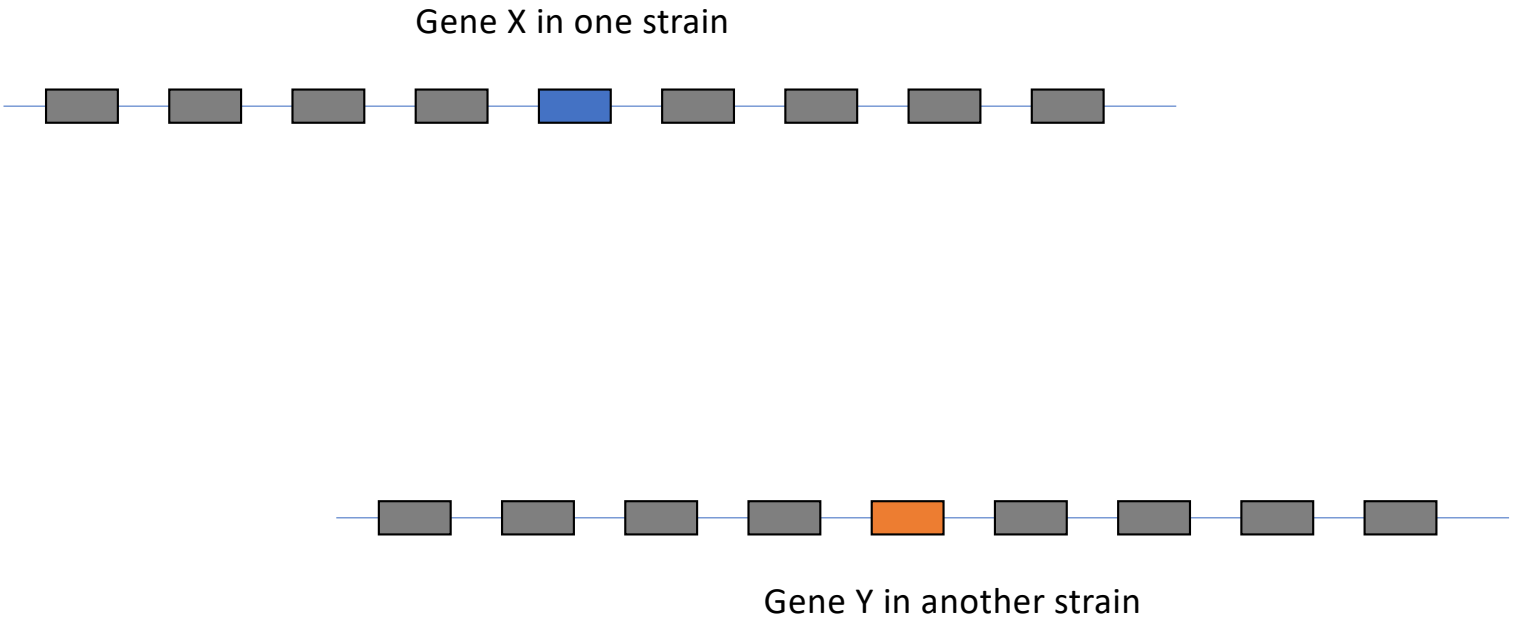


Raw score

```
E_coli_K12-AA      1 MRVLKFGGTSVANAERFLRVADILESNARQGQVATVLSAPAKITNHLVAM     50
                     |||||||||||||||||||||||||||||||||||||||||||||||||||
E_albertii-BA      1 MRVLKFGGTSVANAERFLRVADILESNARQGQVATVLSAPAKITNHLVAM     50


E_coli_K12-AA     51 IEKTISGQDALPNISDAERIFAELLTGLAAAQPGFPLAQLKTFVDQEFAQ    100
                     |||||||||||||||||||||||||||||.||||||||||||||||.||||
E_albertii-BA     51 IEKTISGQDALPNISDAERIFAELLTGLADAQPGFPLAQLKTFVDLEFAQ    100


E_coli_K12-AA    101 IKHVLHGISLLGQCPDSINAALICRGEKMSIAIMAGVLEARGHNVTVIDP    150
                     |||||||||||||||||||||||||||||||||||||||||||||||||||
E_albertii-BA    101 IKHVLHGISLLGQCPDSINAALICRGEKMSIAIMAGVLEARGHNVTVIDP    150
```

- Based on global alignment
- Uses parasail python package

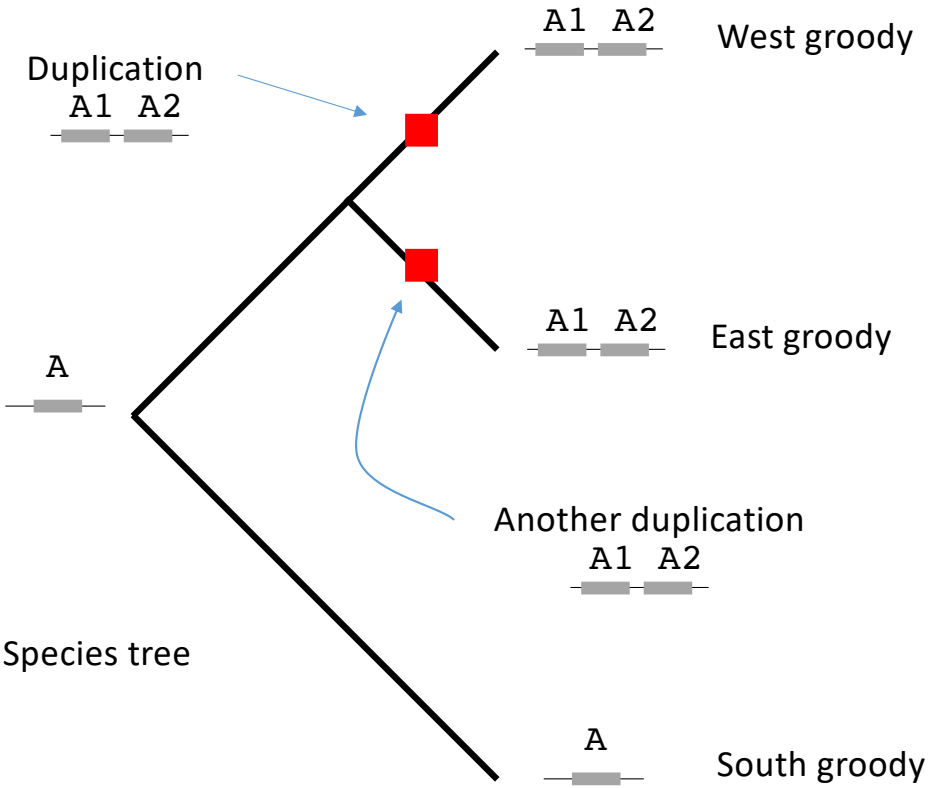
Local synteny score

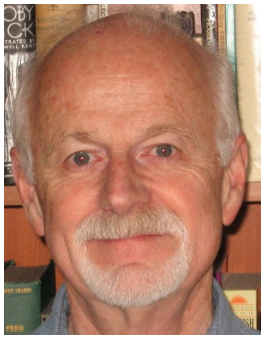


There's a Score class…

```
class Score:

    def __init__(self):
        '''Create an object for storing scores.'''

        self.endNodesToEdgeD = {}
        self.numEdges = 0 # to start out
        self.strainPairScoreLocationD = {}
        self.scoreD = {}

    def initializeDataAttributes(self,blastFnL,paramD,strainNamesT):
        '''This method takes a new, empty object and fills the data attributes
by reading through blast files to identify pairs of genes with
edges. Also creates the array for storing raw scores, initialized to
0. Arrays for other score types will be created later. The genes
included here depend on what is found in the blast files in blastFnL.
```

Evolution of the “A” gene family

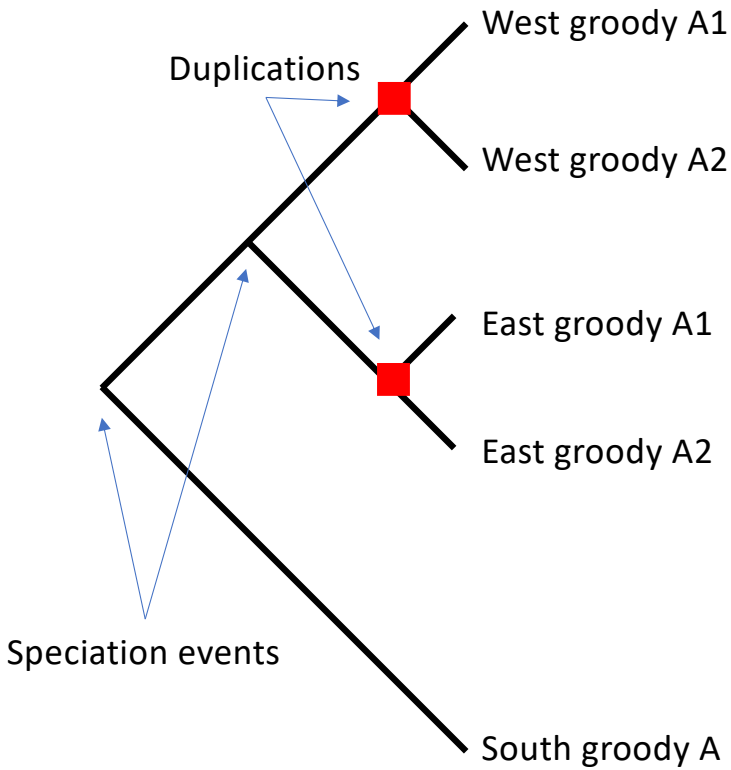




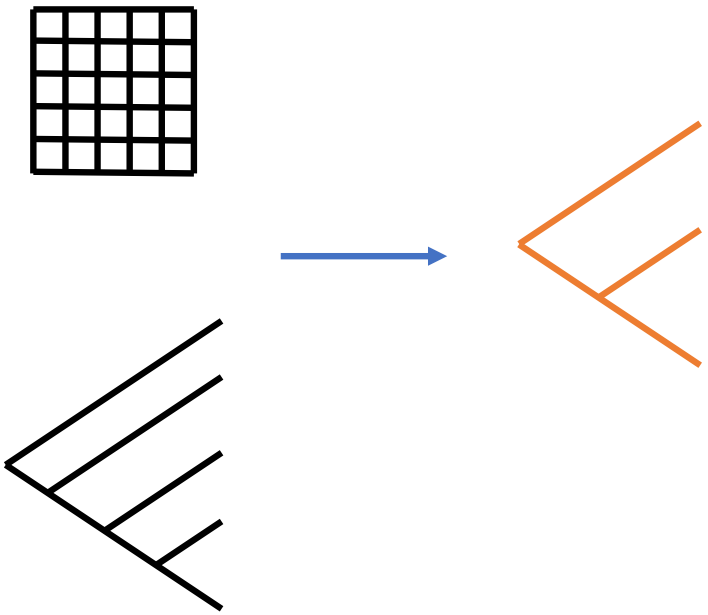
Bill Purves

A gene tree

- Tips are genes
- Internal nodes may be speciation events **or** duplication events



Form gene families taking account of the species tree and synteny information



Use custom DTLOR reconciliation algorithm to make gene families

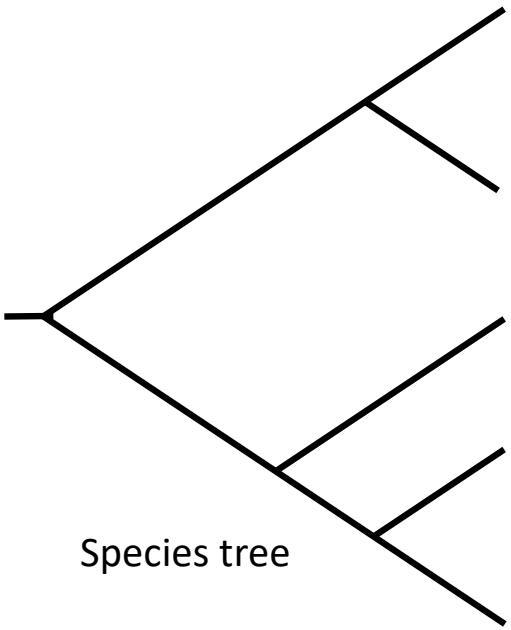
D Duplication

T Transfer

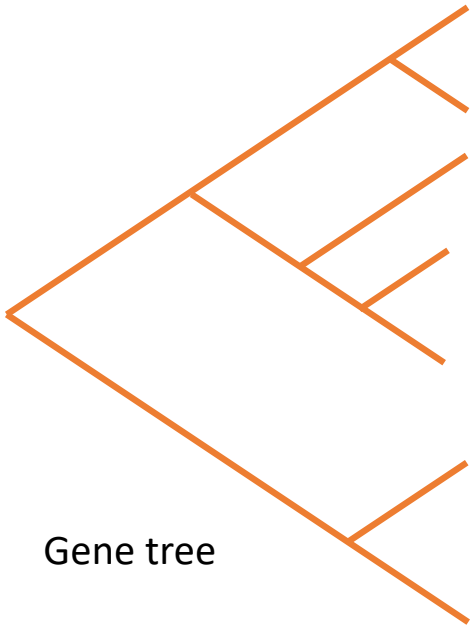
L Loss

O Origin

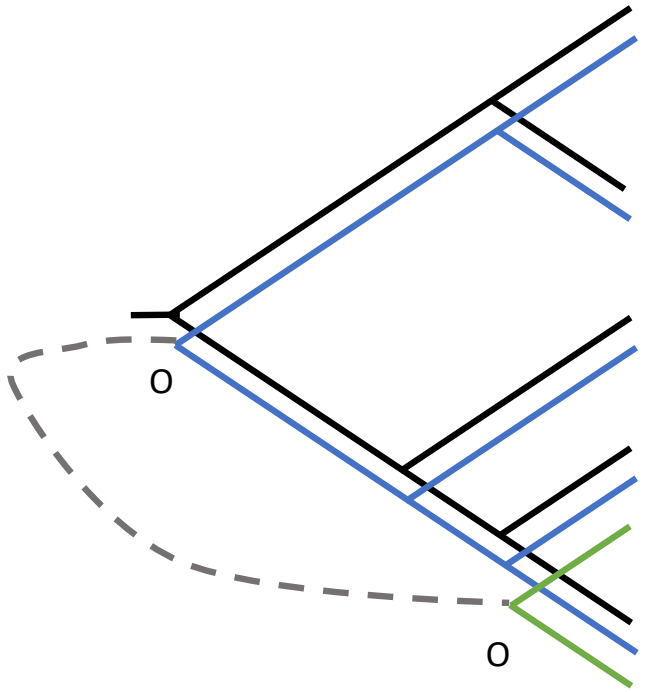
R Rearrangement



Species tree

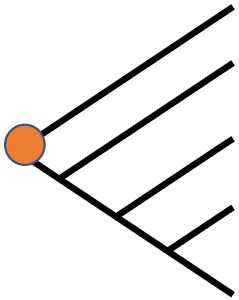


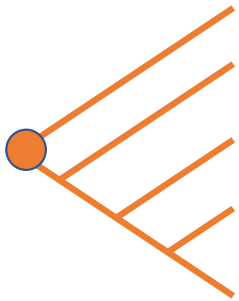
Gene tree



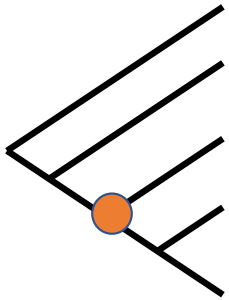
An “origin” family descends from a single ancestor gene within the species tree

Core gene family





HGT gene family

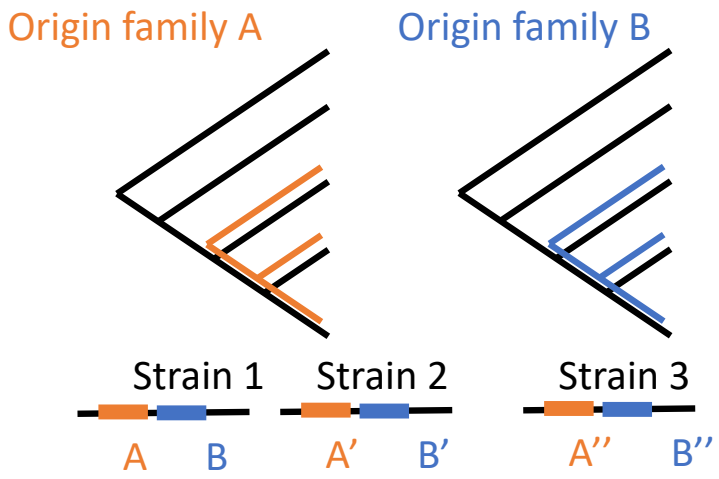


```
class Family:
    def __init__(self,famNum,mrca,geneTreeO=None,dtlorMprD=None,sourceFam=None):
        '''Base class to be inherited by initialFamily and originFamily.'''

        self.famNum = famNum
        self.mrca = mrca
        self.locusFamiliesL = []   # will contain locusFamily objects for this family
        self.geneTreeO = geneTreeO # gene tree object
        self.dtlorMprD = dtlorMprD # mpr dict
        self.sourceFam = sourceFam # specifies the corresponding fource fam (blast or ifam)
        # rest not shown...
```

```
class initialFamily(Family):

    def __init__(self,famNum,mrca,geneTreeO=None,dtlorCost=None,dtlorGraphD=None,dtlorMprD=None,sour
ceFam=None,productFamT=None):
        '''Initialize an object of class initialFamily.'''
        super().__init__(famNum,mrca,geneTreeO,dtlorMprD,sourceFam)
        #  dtlorCost is cost from dtlor alg. A positive integer. In
        #  the case where a family was re-run with permissive origin
        #  costs, we make this value negative.
        self.dtlorCost = dtlorCost
        self.dtlorGraphD = dtlorGraphD
        self.productFamT = productFamT # ids of origin families made from this ifam
        # rest not shown...
```

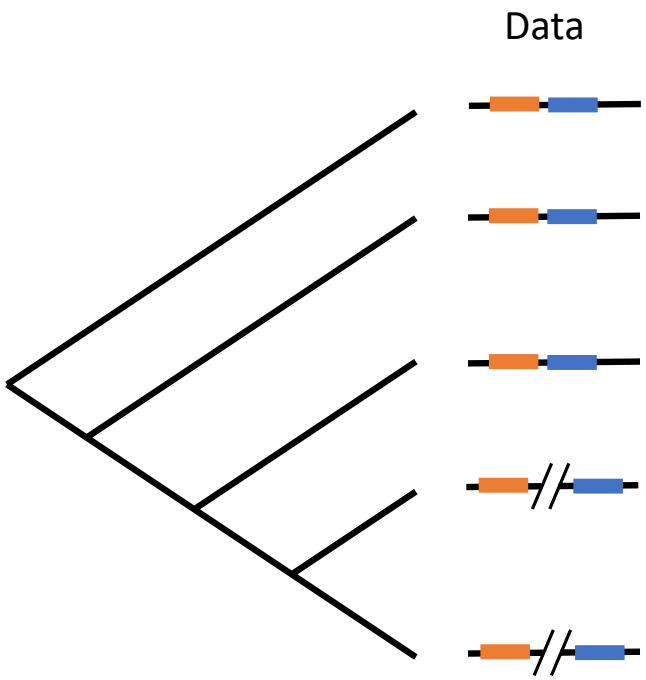
```
class originFamily(Family):

    def __init__(self,famNum,mrca,geneTreeO=None,dtlorMprD=None,sourceFam=None):
        '''Initialize an object of class originFamily.'''
        super().__init__(famNum,mrca,geneTreeO,dtlorMprD,sourceFam)
        # rest not shown...
```

Merge origin families with a common origin into islands



A case where we’d like to merge families into an island.

Parsimony based cost with a simple model of genome evolution



Assuming  in ancestor: cost = 1

Assuming  in ancestor: cost = 3

- Create Island data structure. Initially, each island contains one family
- Calculate rearrangement score between all pairs of islands

Rearrangement score = Cost assuming  in ancestor − Cost assuming  in ancestor

- Greedily merge islands

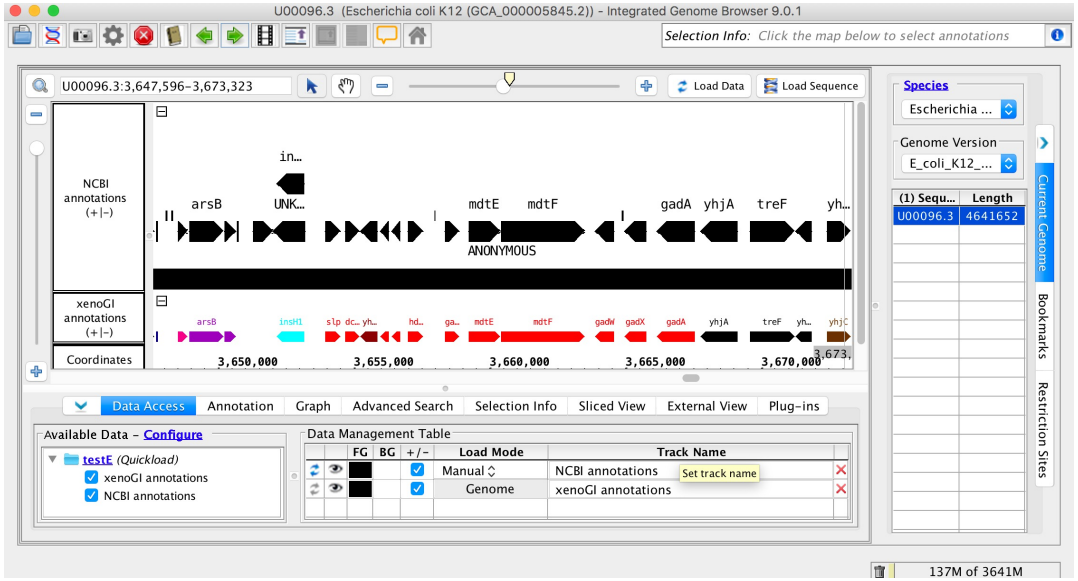
```
class LocusIsland:

    def __init__(self, idnum, mrca, locusFamilyL):
        '''Create a LocusIsland object.'''
        self.id = idnum
        self.mrca = mrca
        self.locusFamilyL=locusFamilyL # list of locus family numbers in the island, in order

    def __repr__(self):
        return "<id:"+str(self.id)+", mrca:"+str(self.mrca)+", locusFamilyL:"+str(self.locusFamilyL)+">"

    def __len__(self):
        return len(self.locusFamilyL)
```

rest not shown…

Output

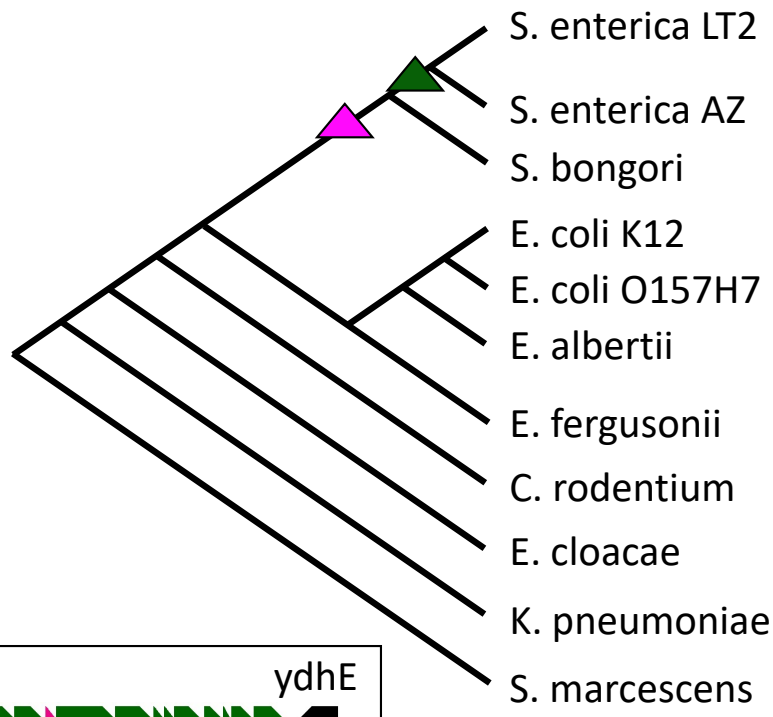
- Text based
  - Organized by species tree node
  - Organized by gene
- Bed file for visualization in a genome browser

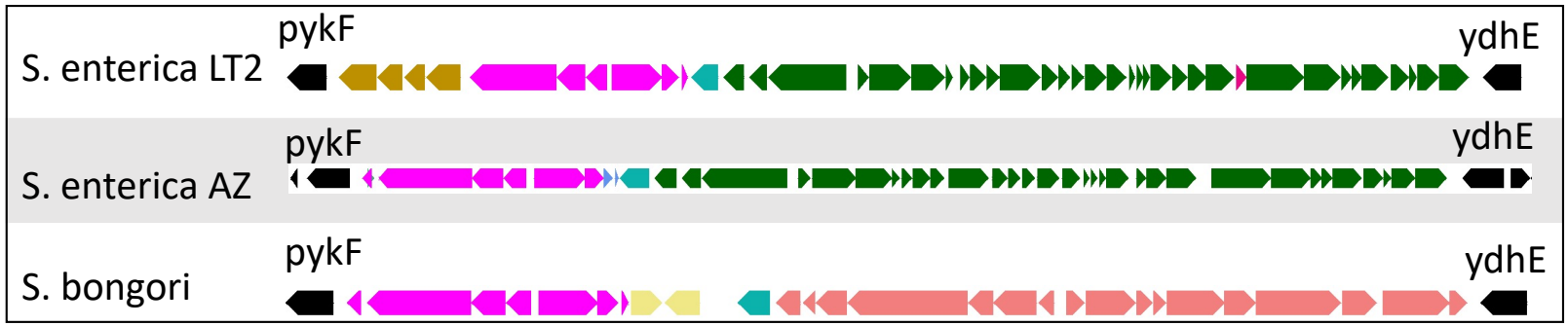
```
track name=E_coli_K12 type=bedDetail visibility=full itemRgb="On" useScore=0
U00096.3        189     255     thrL    0       +       189     189     245,130,48 ...
U00096.3        336     2799    thrA    0       +       336     336     0,0,0      ...
```




```
##########################  Islands at node i0

  ---
  Island 3456
    Family | E_coli_K12                  | E_albertii
    3456   | E_coli_K12-AAC76118.1(higB) | E_albertii-BAT45100.1
    3586   | E_coli_K12-AAC76117.1(higA) | E_albertii-BAT45099.1(ygjM)
    3559   | E_coli_K12-AAC76116.1(fadH) | E_albertii-BAT45098.1(fadH)
    3520   | E_coli_K12-AAC76115.1(ygjK) | E_albertii-BAT45097.1(ygjK)
    3532   | E_coli_K12-AAC76114.1(ygjJ) | E_albertii-BAT45096.1(ygjJ)
    3497   | E_coli_K12-AAT48165.1(ygjI) | E_albertii-BAT45095.1(ygjI)
    3484   | E_coli_K12-AAC76112.1(ebgC) | E_albertii-BAT45094.1(ebgC)
    3562   | E_coli_K12-AAT48164.1(ebgA) | E_albertii-BAT45093.1(ebgA)
    3564   | E_coli_K12-AAC76110.1(ebgR) | E_albertii-BAT45092.1(ebgR)
    3533   | E_coli_K12-AAC76109.1(ygjH) | E_albertii-BAT45091.1(ygjH)
    3472   | E_coli_K12-AAC76108.3(patA) | E_albertii-BAT45090.1(ygjG)
```

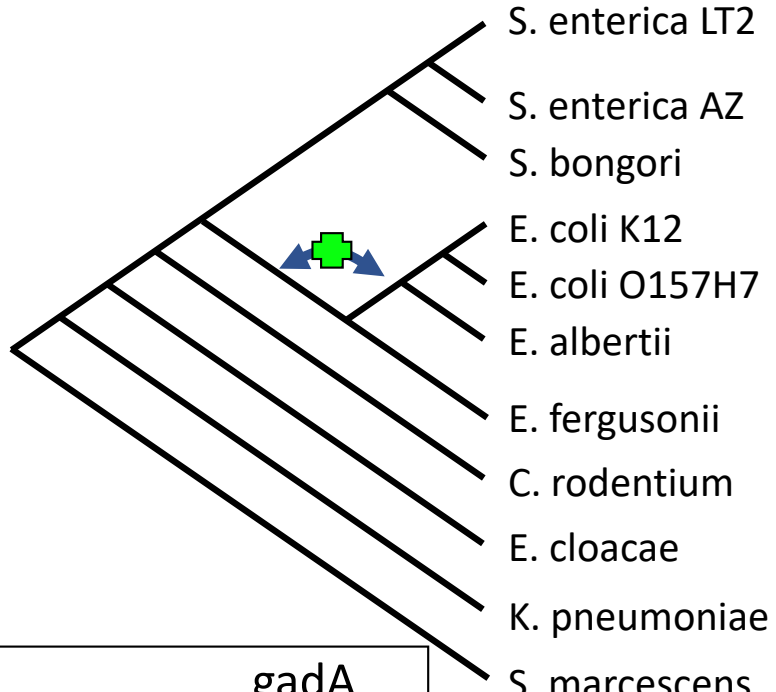
Examples from enteric bacteria: Salmonella pathogenicity island 2

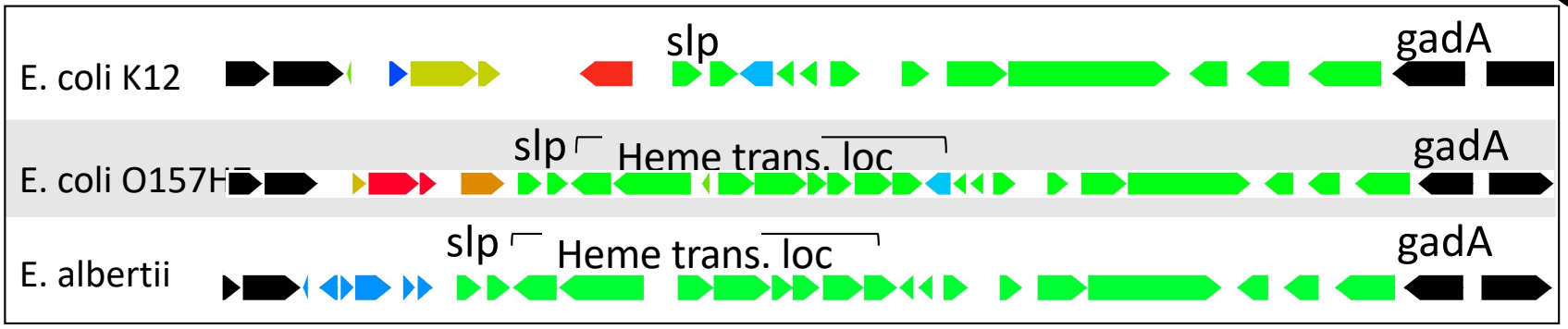




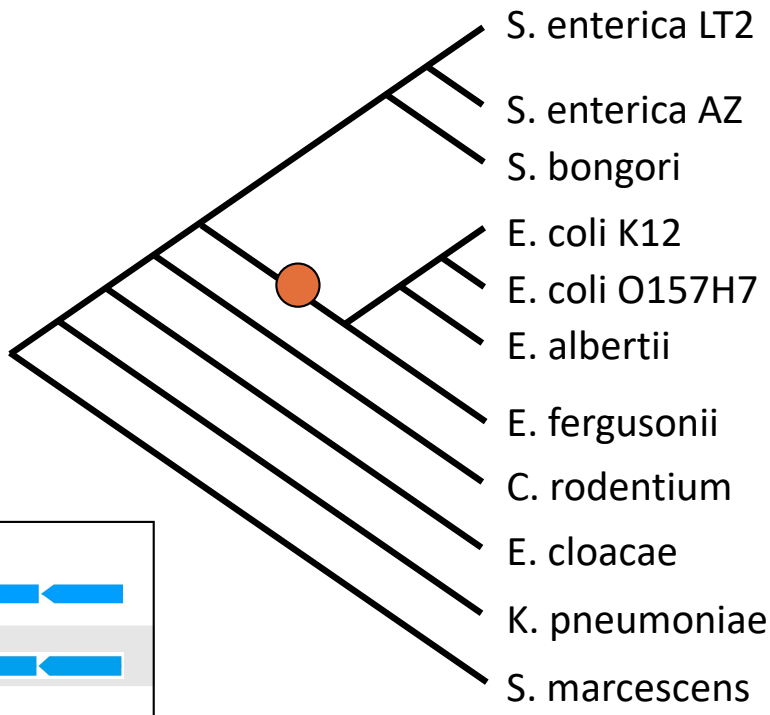
Examples from enteric bacteria: acid fitness in the Escherichia clade

- AFI correctly identified
- Co-localized with heme transport locus
- Timing

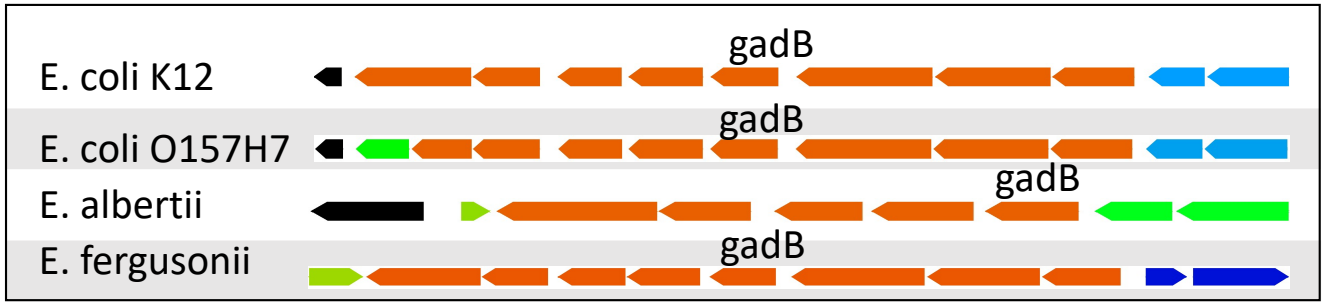




Example from enteric bacteria: the genomic island around gadB



Island around gadB



Resources

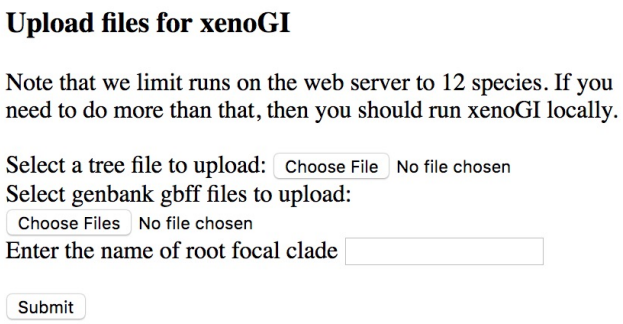
Available as a pip installable python package.

Code :

https://github.com/ecbush/xenoGI

A simple web service:

http://www.cs.hmc.edu/xgiWeb/



Acknowledgments

Anne Clark
Carissa DeRanek
Alex Eng
Jacob Fischer
Juliette Forman
Tona Gonzalez
Kevin Heath
Chan Hong
Michelle Johnson
Bo Lee
Katie Li (Pomona ‘18)
Ran Libeskind-Hadas
Ivy Liu
Rose Liu
Fabrizia Mugnatto
Ross Mawhorter
Santi Santichaivekin
Rachael Soh
Zunyan Wang
Matt Wilbur
Joe Wirth
Helen Wu