# Phylogenetic reconstruction: criteria

Bastien Boussau
*Bastien.boussau@univ-lyon1.fr*
*@bastounette*



# Phylogenetic inference

*How to find the best tree given my data?*

- **Need for a criterion/score**

- **Need for an algorithm to find/construct the tree**

# Phylogenetic inference

*How to find the best tree given my data?*

- **Need for a criterion/score**
  - Maximum Parsimony
  - Minimum Evolution or least squares (distance methods)
  - Maximum Likelihood ~ P(D|M)
  - Posterior Probability P(M|D)
- **Need for an algorithm to find/construct the tree**

# Phylogenetic inference

*How to find the best tree given my data?*

- **Need for a criterion/score**
  - Maximum Parsimony
  - Minimum Evolution or least squares (distance methods)
  - Maximum Likelihood ~ P(D|M)
  - Posterior Probability P(M|D)
- **Need for an algorithm to find/construct the tree**

# Phylogenetic inference

*How to find the best tree given my data?*

- **Need for a criterion/score**
  - Maximum Parsimony
  - Minimum Evolution or least squares (distance methods)
  - Maximum Likelihood ~ P(D|M)
  - Posterior Probability P(M|D)
- **Need for an algorithm to find/construct the tree**
  - e.g.: try several topologies, (choose some branch lengths,) score the topologies, choose the one that has the best score

# Plan: Criteria for evaluating phylogenies

- Criteria for evaluating phylogenetic trees:
  - Distance methods
  - Parsimony
  - Maximum Likelihood

  } ***Alexis!***

  - Posterior probability (Bayesian approach)
- Conventions:
  - We’re dealing with aligned sequence data
  - gaps are not taken into account

# Distance methods

- Distance-based approaches:
  - least squares methods,
  - Minimum evolution method,
  - Neighbor Joining.

# Minimum Evolution or least squares: distance methods

- Use a distance matrix:

*Sp1* ATGCGCT...

*Sp2* AGTCGCA...

*Sp3* AGGTGCA...

Sp4 ATGCCCT...

# Minimum Evolution or least squares: distance methods

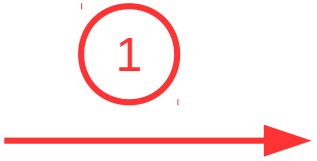
- Use a distance matrix:

*Sp1* ATGCGCT...

*Sp2* AGTCGCA...

*Sp3* AGGTGCA...

*Sp4* ATGCCCT...




| | Sp1 | Sp2 | Sp3 | Sp4 |
|---|---|---|---|---|
| Sp1 | 0 | 0.1 | 0.2 | 0.15 |
| Sp2 | 0.1 | 0 | 0.3 | 0.01 |
| Sp3 | 0.2 | 0.3 | 0 | 0.6 |
| Sp4 | 0.15 | 0.01 | 0.6 | 0 |

# Minimum Evolution or least squares: distance methods

- Uses a distance matrix:

*Sp1* ATGCGCT...

*Sp2* AGTCGCA...

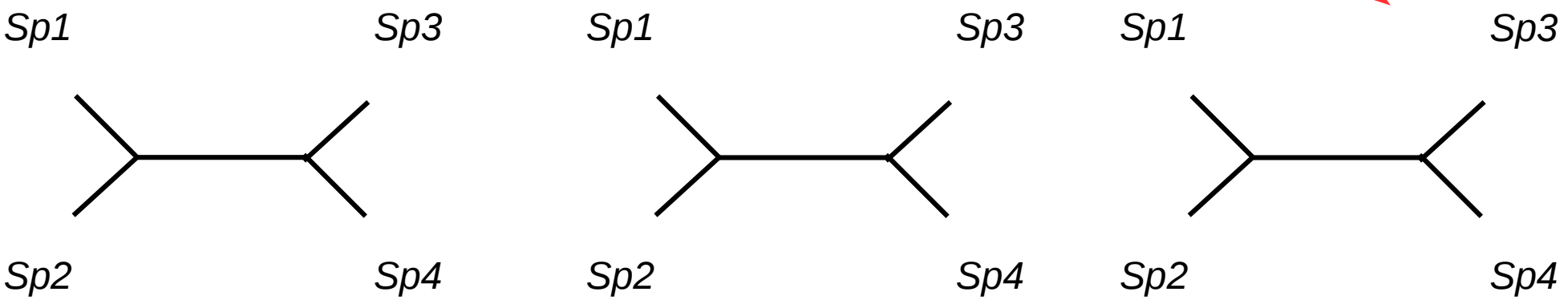
*Sp3* AGGTGCA...

*Sp4* ATGCCCT...

1

| | Sp1 | Sp2 | Sp3 | Sp4 |
|---|---|---|---|---|
| Sp1 | 0 | 0.1 | 0.2 | 0.15 |
| Sp2 | 0.1 | 0 | 0.3 | 0.01 |
| Sp3 | 0.2 | 0.3 | 0 | 0.6 |
| Sp4 | 0.15 | 0.01 | 0.6 | 0 |

2



# Minimum Evolution or least squares: distance methods

- Uses a distance matrix:

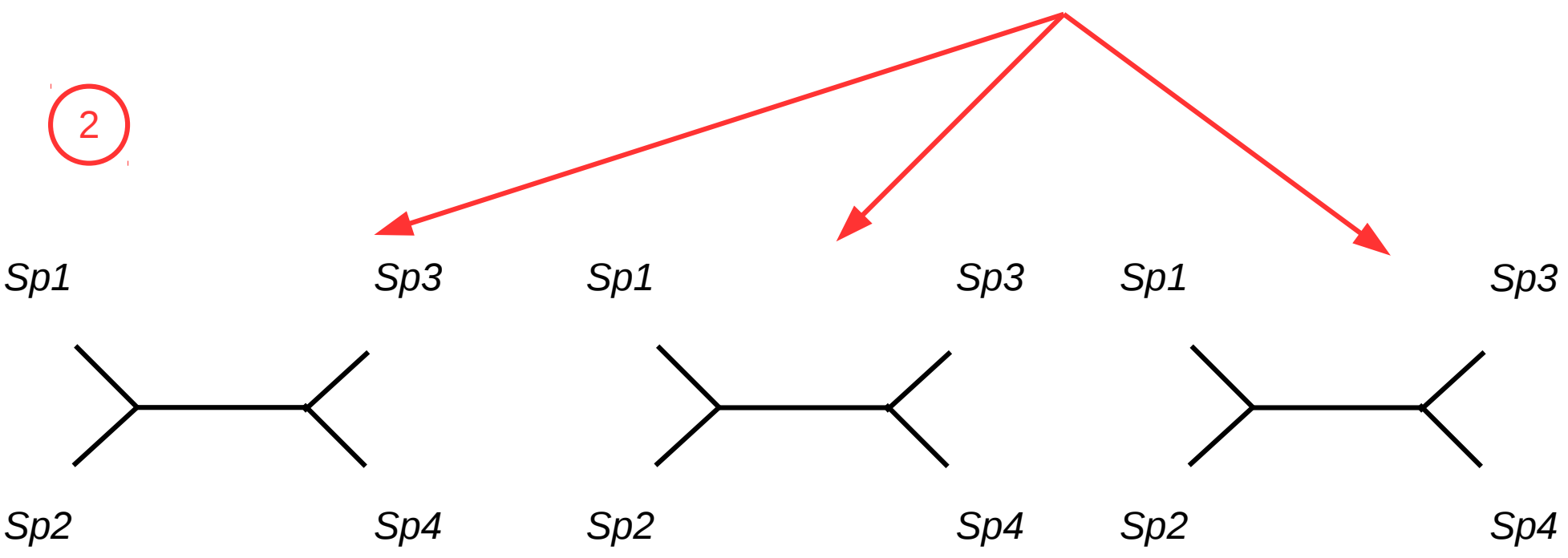
*Sp1* ATGCGCT...
*Sp2* AGTCGCA...
*Sp3* AGGTGCA...
*Sp4* ATGCCCT...

1

***How to compute the distance matrix?***

| | Sp1 | Sp2 | Sp3 | Sp4 |
|---|---|---|---|---|
| Sp1 | 0 | 0.1 | 0.2 | 0.15 |
| Sp2 | 0.1 | 0 | 0.3 | 0.01 |
| Sp3 | 0.2 | 0.3 | 0 | 0.6 |
| Sp4 | 0.15 | 0.01 | 0.6 | 0 |

2



# Minimum Evolution or least squares: distance methods

- Uses a distance matrix:

```
Sp1 ATGCGCT...
Sp2 AGTCGCA...
Sp3 AGGTGCA...
Sp4 ATGCCCT...
```

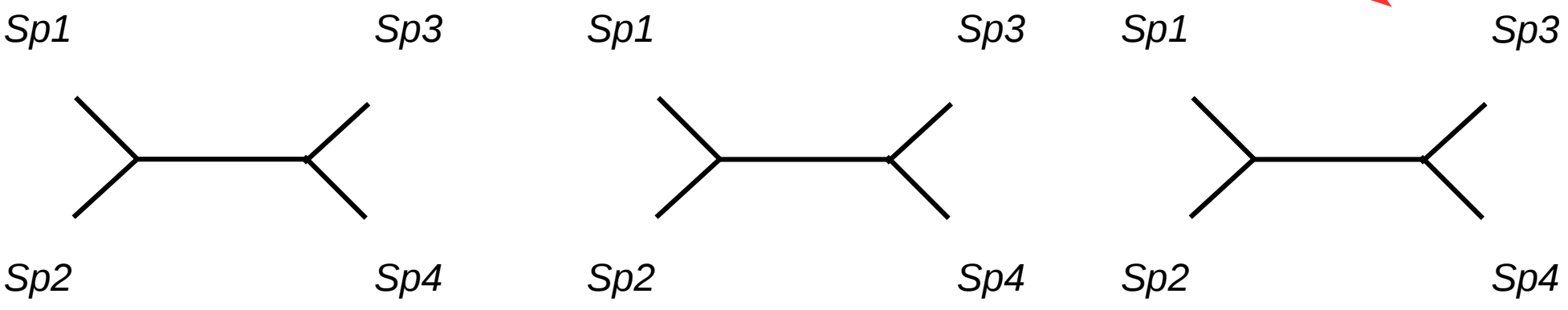
1

***How to compute the distance matrix?***

| | Sp1 | Sp2 | Sp3 | Sp4 |
|---|---|---|---|---|
| Sp1 | 0 | 0.1 | 0.2 | 0.15 |
| Sp2 | 0.1 | 0 | 0.3 | 0.01 |
| Sp3 | 0.2 | 0.3 | 0 | 0.6 |
| Sp4 | 0.15 | 0.01 | 0.6 | 0 |

2

***Which tree fits the distance matrix best?***



# 1: How to compute distances between sequences?

- Simply count differences (observed divergence)

*Sp1* ATGCGCT
*Sp2* AGTCGCA ⟶ d(Sp1-Sp2) =

# 1: How to compute distances between sequences?

- Simply count differences (observed divergence)

```
Sp1 ATGCGCT
Sp2 AGTCGCA
      --   -
```

d(Sp1-Sp2) =

# 1: How to compute distances between sequences?

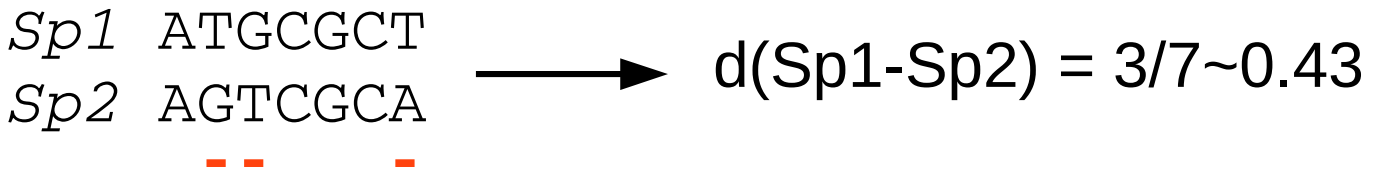
- Simply count differences (observed divergence)

```
Sp1 ATGCGCT
Sp2 AGTCGCA
      --   -
```

d(Sp1-Sp2) = 3/7~0.43

# 1: How to compute distances between sequences?

- Simply count differences (observed divergence)




- Use a model of sequence evolution

  → *cf. talk on models, and talk by Maria!*

  - Advantages:
    - *Hidden substitutions are taken into account*
    - *Parameters of the model of substitution can be estimated*

# Minimum Evolution or least squares: distance methods

- Uses a distance matrix:

```
Sp1 ATGCGCT...
Sp2 AGTCGCA...
Sp3 AGGTGCA...
Sp4 ATGCCCT...
```

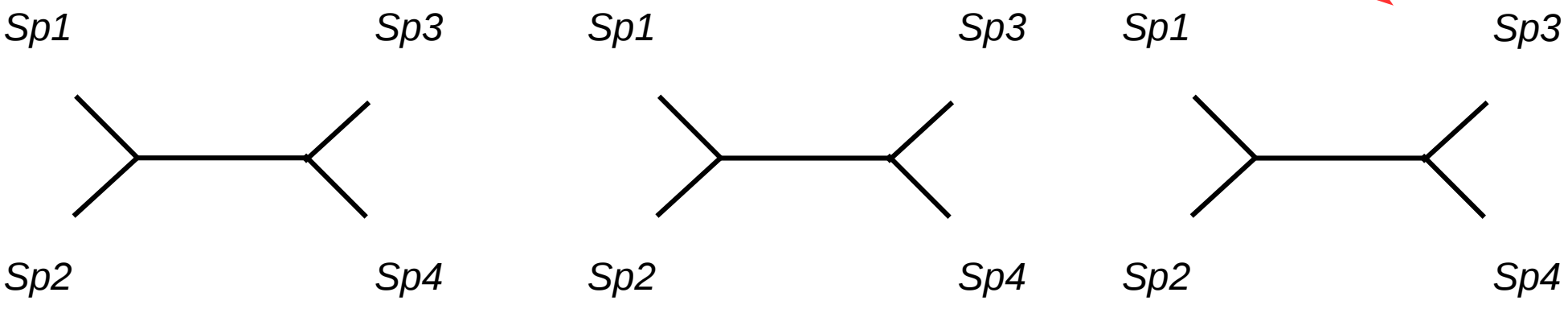
1

***How to compute the distance matrix?*** ✓

| | Sp1 | Sp2 | Sp3 | Sp4 |
|---|---|---|---|---|
| Sp1 | 0 | 0.1 | 0.2 | 0.15 |
| Sp2 | 0.1 | 0 | 0.3 | 0.01 |
| Sp3 | 0.2 | 0.3 | 0 | 0.6 |
| Sp4 | 0.15 | 0.01 | 0.6 | 0 |

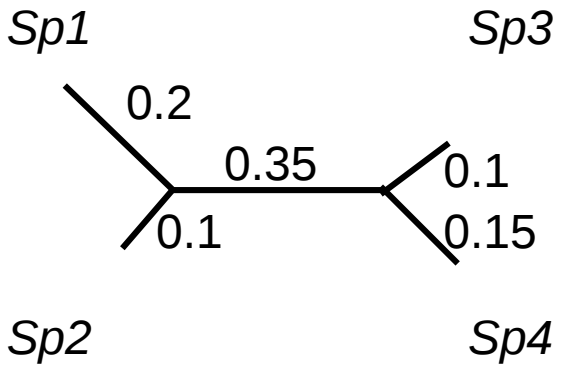
2

***Which tree fits the distance matrix best?***



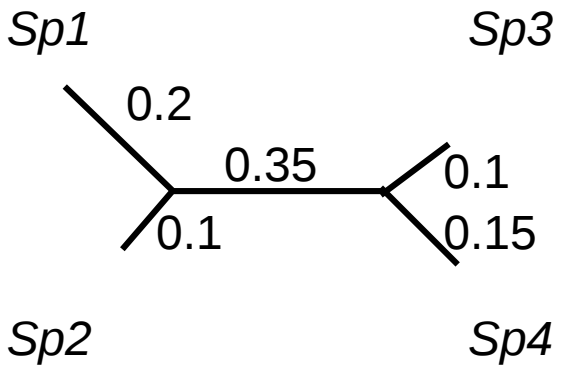
# 2: How to compute the fit between a distance matrix and a tree?

A tree implies distances between tips: compare those **patristic** distances to sequence-based distances



# 2: How to compute the fit between a distance matrix and a tree?

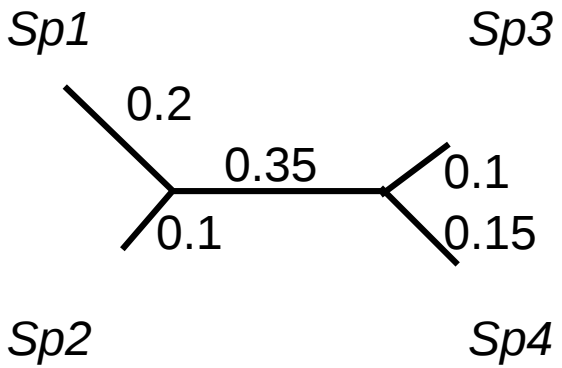
A tree implies distances between tips: compare those **patristic** distances to sequence-based distances




| | Sp1 | Sp2 | Sp3 | Sp4 |
|---|---|---|---|---|
| Sp1 | 0 | 0.3 | 0.65 | 0.15 |
| Sp2 | 0.3 | 0 | 0.55 | 0.6 |
| Sp3 | 0.65 | 0.55 | 0 | 0.25 |
| Sp4 | 0.15 | 0.6 | 0.25 | 0 |

# 2: How to compute the fit between a distance matrix and a tree?

A tree implies distances between tips: compare those **patristic** distances to sequence-based distances

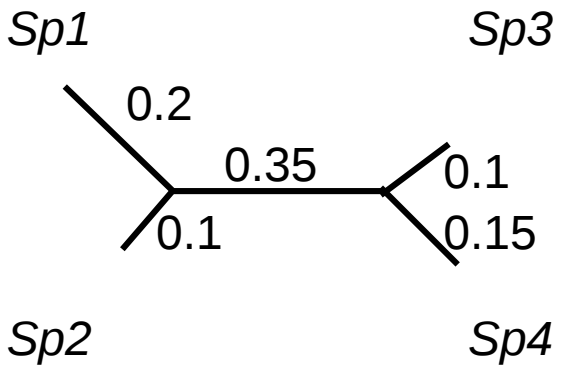



| | Sp1 | Sp2 | Sp3 | Sp4 |
|---|---|---|---|---|
| Sp1 | 0 | 0.3 | 0.65 | 0.15 |
| Sp2 | 0.3 | 0 | 0.55 | 0.6 |
| Sp3 | 0.65 | 0.55 | 0 | 0.25 |
| Sp4 | 0.15 | 0.6 | 0.25 | 0 |

| | Sp1 | Sp2 | Sp3 | Sp4 |
|---|---|---|---|---|
| Sp1 | 0 | 0.1 | 0.2 | 0.15 |
| Sp2 | 0.1 | 0 | 0.3 | 0.01 |
| Sp3 | 0.2 | 0.3 | 0 | 0.6 |
| Sp4 | 0.15 | 0.01 | 0.6 | 0 |

# 2: How to compute the fit between a distance matrix and a tree?

A tree implies distances between tips: compare those **patristic** distances to sequence-based distances




| | Sp1 | Sp2 | Sp3 | Sp4 |
|---|---|---|---|---|
| Sp1 | 0 | 0.3 | 0.65 | 0.15 |
| Sp2 | 0.3 | 0 | 0.55 | 0.6 |
| Sp3 | 0.65 | 0.55 | 0 | 0.25 |
| Sp4 | 0.15 | 0.6 | 0.25 | 0 |

| | Sp1 | Sp2 | Sp3 | Sp4 |
|---|---|---|---|---|
| Sp1 | 0 | 0.1 | 0.2 | 0.15 |
| Sp2 | 0.1 | 0 | 0.3 | 0.01 |
| Sp3 | 0.2 | 0.3 | 0 | 0.6 |
| Sp4 | 0.15 | 0.01 | 0.6 | 0 |

$score_{ULS} = (0-0)^2+(0.3-0.1)^2+(0.65-0.2)^2+\ldots$

With ULS: Unweighted Least Squares
(other criteria have been proposed)

# Computing the optimal distances on a given topology

Using the ULS criterion, we can compute the fit between a sequence-based distance matrix and any tree (topology + branch lengths), thanks to the patristic matrix trick.

# Computing the optimal distances on a given topology

Using the ULS criterion, we can compute the fit between a sequence-based distance matrix and any tree (topology + branch lengths), thanks to the patristic matrix trick.

*But how can we estimate branch lengths on the topology?*

# Computing the optimal distances on a given topology

Using the ULS criterion, we can compute the fit between a sequence-based distance matrix and any tree (topology + branch lengths), thanks to the patristic matrix trick.

*But how can we pick branch lengths on the topology?*

ULS provides a mathematical way to find the optimal branch lengths on a given topology! This involves some simple matrix algebra (solving a set of linear equations).

# Searching for the best tree using Unweighted Least Squares

- We now know how to compute the ULS score of a tree topology. It involves:
  - Matrix algebra to find the best branch lengths
  - Computing the $score_{ULS}$ for that tree
- Given a set of tree topologies, we can compute the “best” tree topology according to the ULS criterion: it is the one with the lowest $score_{ULS}$
- How to obtain a set of tree topologies to score is tackled later in the course *(see Maria’s talk)*

# Searching for the best tree using Unweighted Least Squares

- We now know how to compute the ULS score of a tree topology. It involves:
  - Matrix algebra to find the best branch lengths
  - Computing the $score_{ULS}$ for that tree
- Given a set of tree topologies, we can compute the “best” tree topology according to the ULS criterion: it is the one with the lowest $score_{ULS}$
- How to obtain a set of tree topologies to score is tackled later in the course *(see Maria’s talk)*

# Minimum evolution criterion

- Motivation similar to parsimony
- **Hypothesis:** the true tree should be the shortest tree
- → Idea:
  - Given a matrix of pairwise distances and a set of tree topologies to evaluate
  - Match pairwise distances onto each tree topology
  - Sum the branch lengths on each tree
  - ***Your best estimate is the tree with the smallest sum of branch lengths***

# Minimum evolution criterion

- Motivation similar to parsimony
- **Hypothesis**: the true tree should be the shortest tree
- → Idea:
  - Given a matrix of pairwise distances and a set of tree topologies to evaluate
  - Match pairwise distances onto each tree topology: *Use least-squares fitting!*
  - Sum the branch lengths on each tree
  - ***Your best estimate is the tree with the smallest sum of branch lengths***

# Minimum Evolution or least squares: distance methods

- Uses a distance matrix:

```
Sp1 ATGCGCT...
Sp2 AGTCGCA...
Sp3 AGGTGCA...
Sp4 ATGCCCT...
```

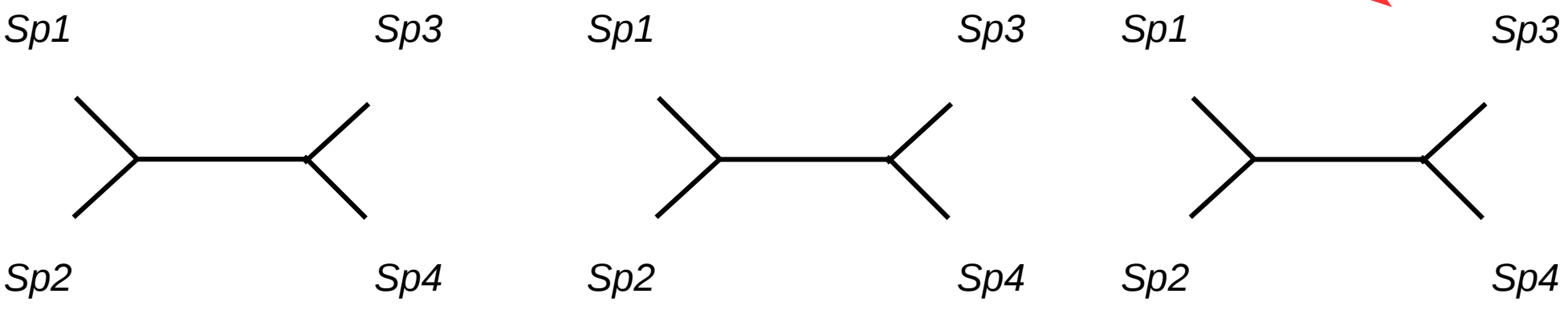
1

***How to compute the distance matrix?***

| | Sp1 | Sp2 | Sp3 | Sp4 |
|---|---|---|---|---|
| Sp1 | 0 | 0.1 | 0.2 | 0.15 |
| Sp2 | 0.1 | 0 | 0.3 | 0.01 |
| Sp3 | 0.2 | 0.3 | 0 | 0.6 |
| Sp4 | 0.15 | 0.01 | 0.6 | 0 |

2

***ME: involves the patristic matrix, matrix algebra and summing branch lengths***



# Minimum evolution criterion

- To obtain a Minimum Evolution tree, at some point we have to use Least Squares estimation to assign branch lengths to a tree topology

  → hybrid approach where two different criteria are mixed up

- However, Minimum evolution works pretty well in practice
- Neighbor-Joining (Saitou and Nei, 1987) is a famous heuristic algorithm for finding the Minimum Evolution tree (not seen in our course, but has been very widely used); see Gascuel and Steel, 2006 for a clear explanation

# Summary on distance methods

- Distance methods are the fastest phylogenetic methods available, notably thanks to Neighbor Joining and others (e.g. BioNJ, Weighbor, FastME…)
- Can be based on models of sequence evolution
- Better than Maximum Parsimony when sequences are divergent, but less accurate than Maximum Likelihood or Bayesian Inference
- The main reason is that distance methods do not use the entire data matrix together, but look at it pair of sequences by pair of sequences

# Plan: Criteria for evaluating phylogenies

- Criteria for evaluating phylogenetic trees:
  - Distance methods
  - Parsimony
  - Maximum Likelihood

  } ***Alexis!***

  - Posterior probability (Bayesian approach)
- Conventions:
  - We’re dealing with aligned sequence data
  - gaps are not taken into account