Parsimony-Based Approaches to Inferring Phylogenetic Trees

BMI/CS 576

www.biostat.wisc.edu/bmi576.html

Mark Craven

craven@biostat.wisc.edu

Phylogenetic tree approaches

- three general types
  - *distance*: find tree that accounts for estimated evolutionary distances
  - *parsimony*: find the tree that requires minimum number of changes to explain the data
  - *maximum likelihood*: find the tree that maximizes the likelihood of the data

Parsimony based approaches

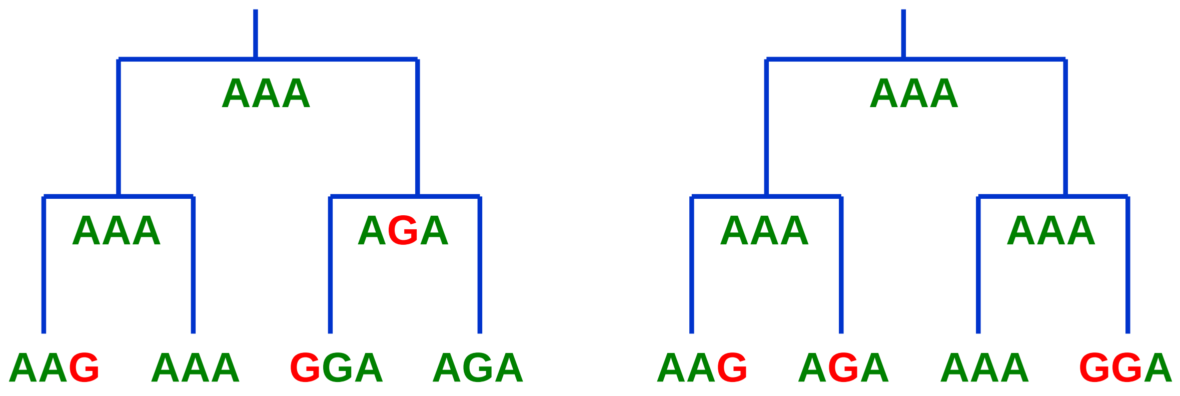
given: character-based data

do: find tree that explains the data with a minimal number of changes

- focus is on finding the right tree topology, not on estimating branch lengths

Parsimony example

- there are various trees that could explain the phylogeny of the sequences AAG, AAA, GGA, AGA including these two:



- parsimony prefers the first tree because it requires fewer substitution events

Parsimony based approaches

- usually these approaches involve two separate components
  1. a procedure to find the minimum number of changes needed to explain the data (for a given tree topology)
  2. a search through the space of trees

Finding minimum number of changes for a given tree

- basic assumptions
  – any state (e.g. nucleotide, amino acid) can convert to any other state
  – the “costs” of these changes are uniform
  – positions are independent; we can compute the min number of changes for each position separately

Finding minimum number of changes for a given tree

- brute force approach
  - for each possible assignment of states to the internal nodes, calculate the number of changes
  - report tne min number of changes found

- runtime is $O(Nk^N)$

  k = number of possible character states (4 for DNA)

  N = number of leaves

Fitch's Algorithm [1971]

1. traverse tree from leaves to root <u>determining set of possible *states*</u> (e.g. nucleotides) for each internal node

2. traverse tree from root to leaves <u>picking ancestral states</u> for internal nodes

Fitch's algorithm: Step 1 possible states for internal nodes

- do a *post-order* (from leaves to root) traversal of tree
- determine possible states R_i of internal node i with children j and k

$$R_i = \begin{cases} R_j \cup R_k, & \text{if } R_j \cap R_k = \varnothing \\ R_j \cap R_k, & \text{otherwise} \end{cases}$$

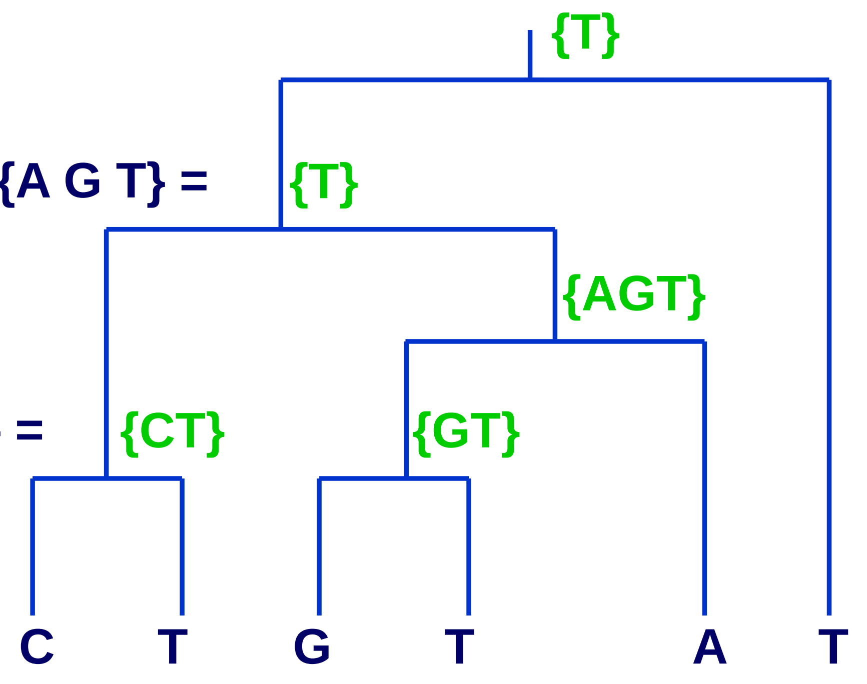
- this step calculates the number of changes required

  # of changes = # union operations

Fitch's algorithm: step 1 example

{C T} ∩ {A G T} =

{C} ∪ {T} =

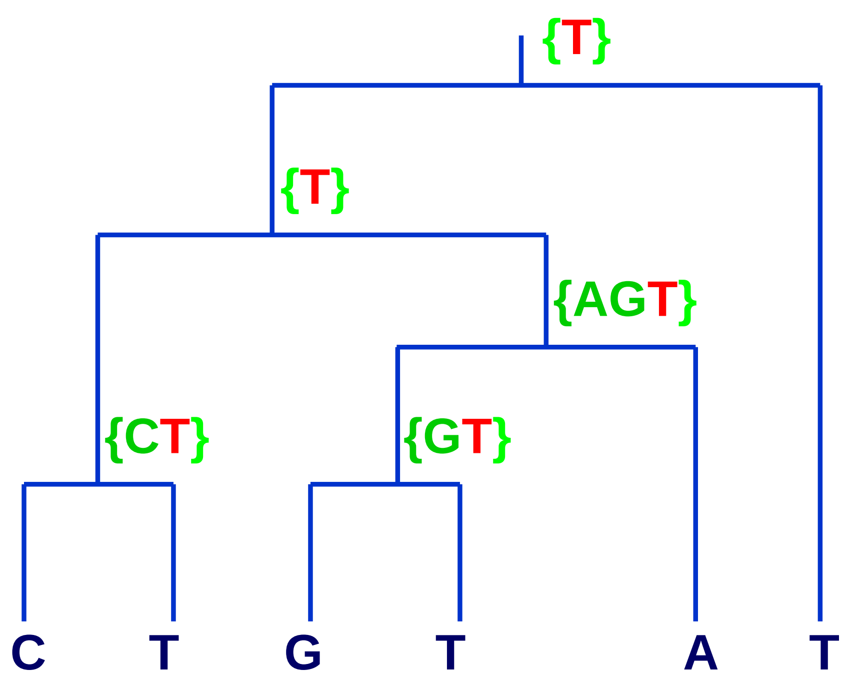


Fitch’s algorithm: step 2 select states for internal nodes

- do a *pre-order* (from root to leaves) traversal of tree
- select state r_j of internal node j with parent i

$$r_j = \begin{cases} r_i, \text{ if } r_i \in \mathrm{R}_j \\ \text{arbitrary state} \in R_j, \text{ otherwise} \end{cases}$$

Fitch’s algorithm: step 2



Weighted parsimony

- [Sankoff & Cedergren, 1983]
- instead of assuming all state changes are equally likely, use different costs $S(a,b)$ for different changes
- 1st step of algorithm is to propagate costs up through tree

$a \rightarrow b$

Weighted parsimony

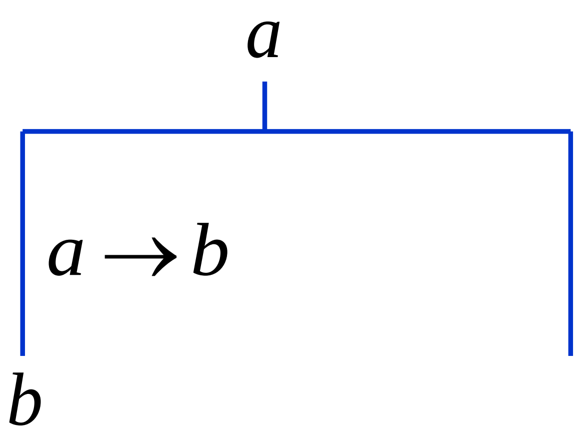
- want to determine cost $R_i(a)$ of assigning character a to node i
- for leaves:

$$R_i(a) = \begin{cases} 0, & \text{if } a \text{ is character at leaf} \\ \infty, & \text{otherwise} \end{cases}$$

Weighted parsimony

- for an internal node i with children j and k:

$$R_i(a) = \min_b (R_j(b) + S(a,b)) + \min_b (R_k(b) + S(a,b))$$



Example: weighted parsimony

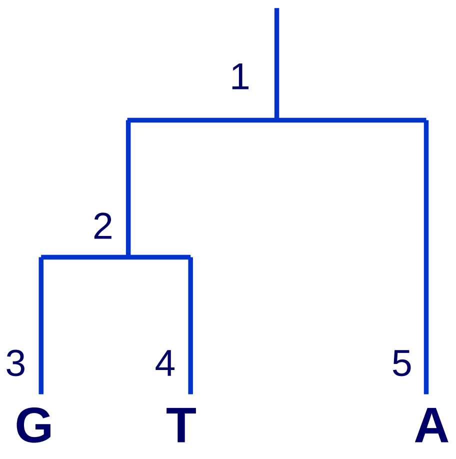
$$R_3[A]=\infty, R_3[C]=\infty, R_3[G]=0, R_3[T]=\infty$$

$$R_4[A]=\infty, R_4[C]=\infty, R_4[G]=\infty, R_4[T]=0$$

$$R_2[A] = R_3[G]+S(A,G)+R_4[T]+S(A,T)$$
$$\vdots$$
$$R_2[T] = R_3[G]+S(T,G)+R_4[T]+S(T,T)$$

$$R_5[A]=0, R_5[C]=\infty, R_5[G]=\infty, R_5[T]=\infty$$

$$R_1[A] = \min\left(R_2[A]+S(A,A), \quad \ldots \quad , \quad R_2[T]+S(A,T)\right) \; + \; R_5[A]+S(A,A)$$
$$\vdots$$
$$R_1[T] = \min\left(R_2[A]+S(T,A), \quad \ldots \quad , \quad R_2[T]+S(T,T)\right) \; + \; R_5[A]+S(T,A)$$

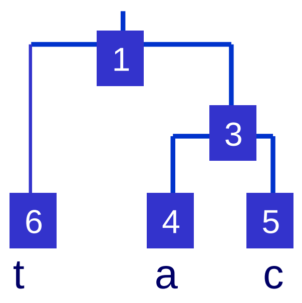


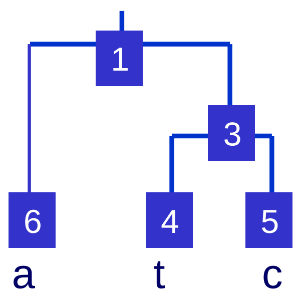
Weighted parsimony: step 2

- do a pre-order (from root to leaves) traversal of tree
  - for root node: select minimal cost character
  - for each internal node: select the character that resulted in the minimum cost explanation of the character selected at the parent

Weighted parsimony example

Consider the two simple phylogenetic trees shown below, and the symmetric cost matrix for assessing nucleotide changes. The tree on the right has a cost of 0.8





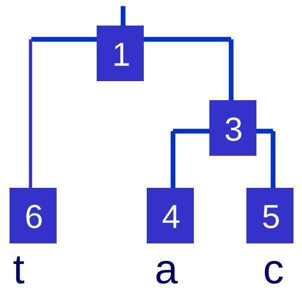
	a	c	g	t
a	0	0.8	0.2	0.9
c	0.8	0	0.7	0.5
g	0.2	0.7	0	0.1
t	0.9	0.5	0.1	0

Show how the weighted version of parsimony would determine the cost of the tree on the left.

What are the minimal cost characters for the internal nodes in the tree on the left?

Which of the two trees would the maximum parsimony approach prefer?

Weighted Parsimony Example



	a	c	g	t
a	0	0.8	0.2	0.9
c	0.8	0	0.7	0.5
g	0.2	0.7	0	0.1
t	0.9	0.5	0.1	0

$$R_3(a) = 0 + 0.8 = 0.8$$
$$R_3(c) = 0.8 + 0 = 0.8$$
$$R_3(g) = 0.2 + 0.7 = 0.9$$
$$R_3(t) = 0.9 + 0.5 = 1.4$$

$$R_1(a) = 0.9 + \min\{0.8,\ \ 0.8 + 0.8,\ \ 0.2 + 0.9,\ \ 0.9 + 1.4\} = 1.7$$
$$R_1(c) = 0.5 + \min\{0.8 + 0.8,\ \ 0.8,\ \ 0.7 + 0.9,\ \ 0.5 + 1.4\} = 1.3$$
$$R_1(g) = 0.1 + \min\{0.2 + 0.8,\ \ 0.7 + 0.8,\ \ 0.9,\ \ 0.1 + 1.4\} = 1.0$$
$$R_1(t) = 0 + \min\{0.9 + 0.8,\ \ 0.5 + 0.8,\ \ 0.1 + 0.9,\ \ 1.4\} = 1.0$$

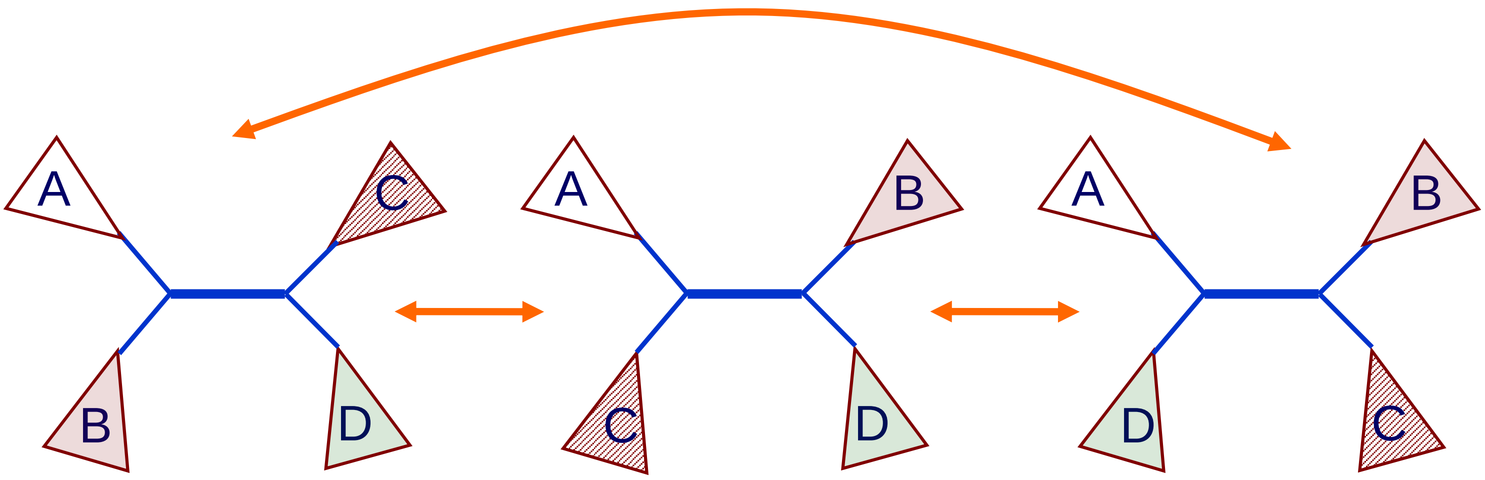
The minimal cost characters for node 1 are either **g** or **t**. The minimal cost character for node 3 is **g**. The maximum parsimony approach would prefer the other tree (exercise left to the reader).

Exploring the space of trees

- we've considered how to find the minimum number of changes for a given tree topology
- need some search procedure for exploring the space of tree topologies

Heuristic method: nearest neighbor interchange

- for any internal edge in a tree, there are 3 ways the four subtrees can be grouped
- nearest neighbor interchanges move from one grouping to another

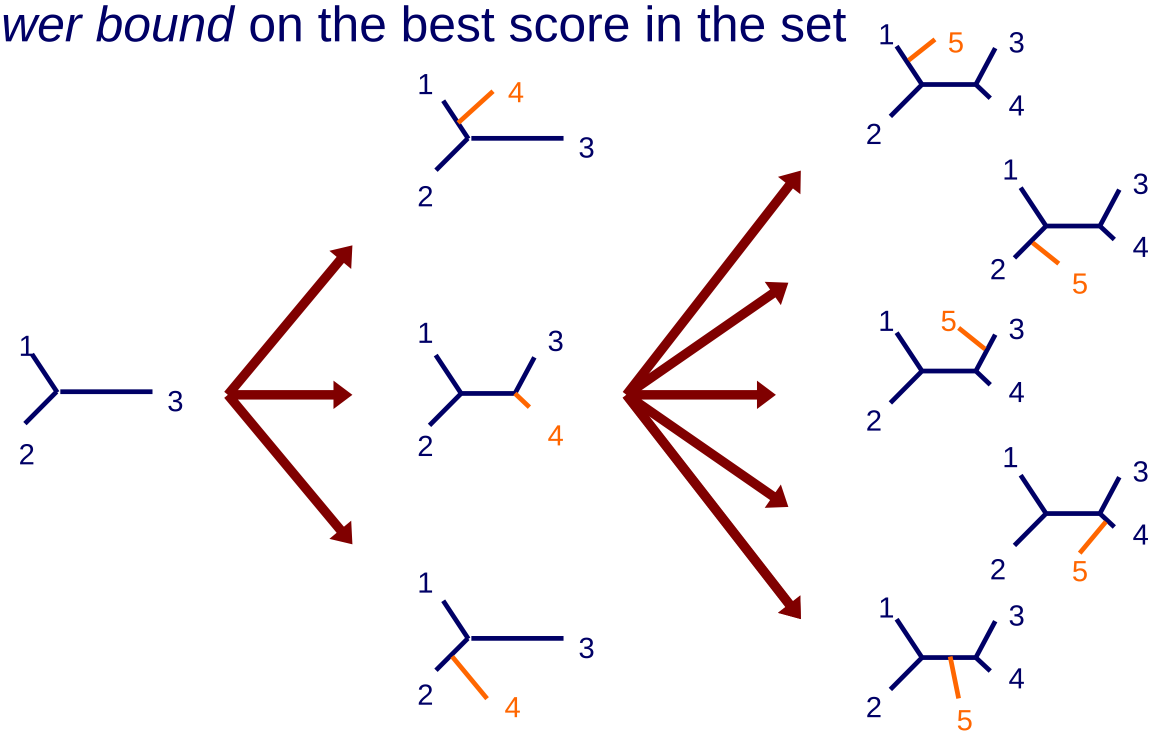


Heuristic method: hill-climbing with nearest neighbor interchange

```
given: set of leaves $L$
create an initial tree $t$ incorporating all leaves in $L$
best-score = parsimony algorithm applied to $t$
repeat
        for each internal edge $e$ in $t$
                for each nearest neighbor interchange
                        $t'$ ← tree with interchange applied to edge $e$ in $t$
                        score = parsimony algorithm applied to $t'$
                        if score < best-score
                                best-score = score
                                best-tree = $t'$
        $t$ = best-tree
until stopping criteria met
```

Exact method: branch and bound

- each partial tree represents a set of complete trees
- the parsimony score on a partial tree provides a *lower bound* on the best score in the set



- search by repeatedly selecting the partial tree with the lowest lower bound

Exact method: branch and bound

```
given: set of leaves $L$
initialize $Q$ with a partial tree with 3 leaves from $L$
repeat
    $t \leftarrow$ tree in $Q$ with lowest lower bound
    if $t$ has incorporated all leaves in $L$
        return $t$
    else
        create new trees by adding next leaf from $L$ to each branch of $t$
        compute lower bound for each tree
        put trees in $Q$ sorted by lower bound
```

Branch and bound (alternate version)

```
given: set of leaves $L$
use heuristic method to grow initial tree $t'$
initialize $Q$ with a partial tree with 3 leaves from $L$
repeat
    $t \leftarrow$ tree in $Q$ with lowest lower bound
    if $t$ has incorporated all leaves in $L$
        return $t$
    else
        create new trees by adding next leaf from $L$ to each branch of $t$
        for each new tree $n$
            if lower-bound($n$) < score($t'$)
            put $n$ in $Q$ sorted by lower bound
```

Rooted or unrooted trees for parsimony?

- we described parsimony calculations in terms of rooted trees
- but we described the search procedures in terms of unrooted trees

- *unweighted parsimony*: minimum cost is independent of where root is located
- *weighted parsimony*: minimum cost is independent of root if substitution cost is a metric (refer back to definition of metric from distance-based methods)

Comments on branch and bound

- it is a *complete* search method
  - guaranteed to find optimal solution
- may be much more efficient than exhaustive search
- in the worst case, it is no better
- efficiency depends
  - the tightness of the lower bound
  - the quality of the initial tree

Comments on tree inference

- search space may be large, but
  - can find the optimal tree efficiently in some cases
  - heuristic methods can be applied
- difficult to evaluate inferred phylogenies: ground truth not usually known
  - can look at agreement across different sources of evidence
  - can look at repeatability across subsamples of the data
  - can look at indirect predictions, e.g. conservation of sites in proteins
- some newer methods use data based on linear order of orthologous genes along chromosome
- phylogenies for bacteria, viruses not so straightforward because of *lateral transfer* of genetic material; “local” phylogenies might be more appropriate

Phylogenetic inference case study: identifying functional regions in proteins

[Pupko et al., Bioinformatics 2002]

Given:

- – multiple sequence alignment for a set of protein sequences
- – a distance-based phylogenetic tree for the sequences

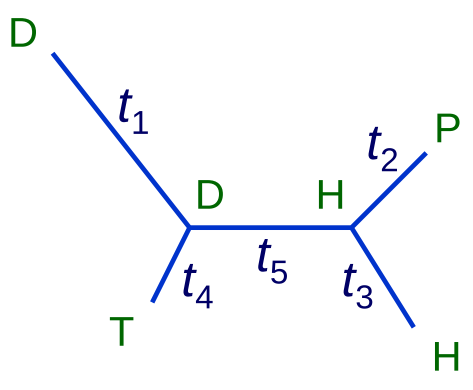
Do:

- – estimate rate of evolution of individual sites in the sequence

- motivation: identify the sites that are most important for the function of the proteins

Identifying functional regions in proteins

- we want to estimate the rate r for each position
- consider a four-sequence example



frequency of amino acid D

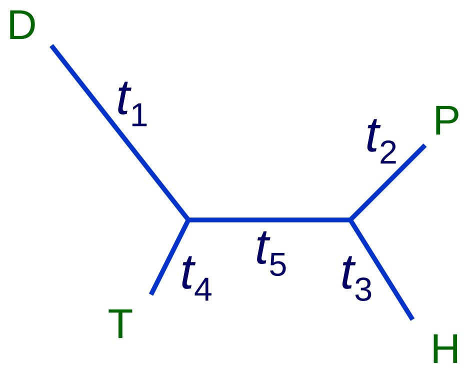
$$P(\{D,H\}, data \mid r) = \pi_D \times P_{D,D}(r \cdot t_1) \times P_{H,P}(r \cdot t_2) \times P_{H,H}(r \cdot t_3) \times P_{D,T}(r \cdot t_4) \times P_{D,H}(r \cdot t_5)$$

prob that amino acid D will be replaced by T along branch length t_4 given rate r

- now we calculate the rate r that maximizes this expression

Identifying functional regions in proteins

- actually we can do this without assuming particular amino-acid assignments at the internal nodes



$$P(data \mid r) = \sum_{X,Y} \begin{pmatrix} \pi_X \times P_{X,D}(r \cdot t_1) \times \\ P_{Y,P}(r \cdot t_2) \times P_{Y,H}(r \cdot t_3) \times \\ P_{X,T}(r \cdot t_4) \times P_{X,Y}(r \cdot t_5) \end{pmatrix}$$

sum over all possibilities

- as before calculate the rate *r* that maximizes this expression

Identifying functional region in proteins

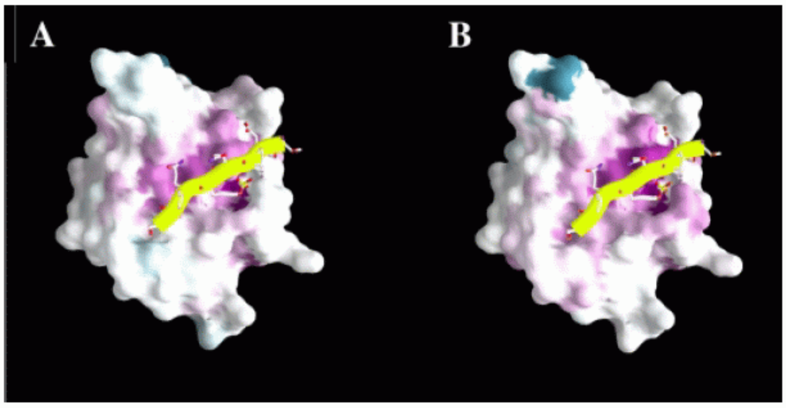


Fig. 2. The peptide-binding groove of the SH2 domain. The structure of the SH2 domain in complex with the C-tail of the tyrosine kinase domain (PDB entry 1fmk, Xu *et al.*, 1997) and MSA of 233 homologues were used. The conservation pattern obtained using MP-ConSurf (**A**) and Rate4Site (**B**) is color-coded onto the molecular surface of the domain: dark violet corresponds to maximal conservation, white corresponds to average conservation level and dark turquoise to maximal variability. The peptide is shown as a balls-and-sticks model with a yellow tube tracing its backbone. The picture was drawn using GRASP (Nicholls *et al.*, 1991). The domain boundaries were set to Trp148 to Pro246 (Al-Lazikani *et al.*, 2001).

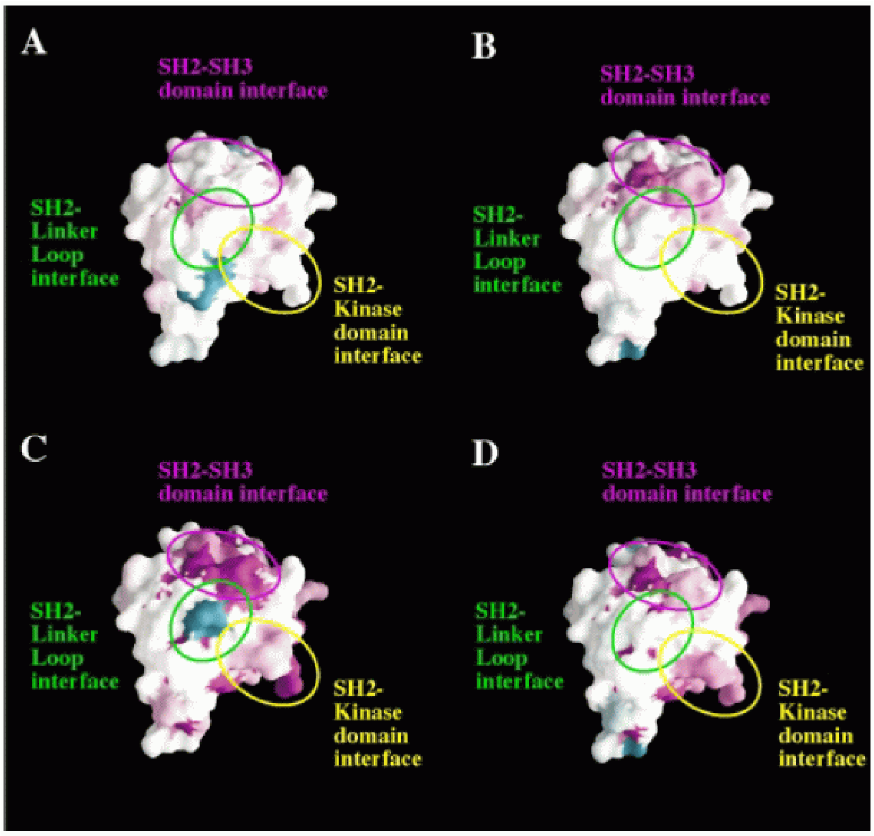
Identifying functional region in proteins

MP-ConSurf method

Rate4Site method

Rates estimated using 233 sequences

Rates estimated using 34 sequences


A
SH2-SH3 domain interface
SH2-Linker Loop interface
SH2-Kinase domain interface
B
SH2-SH3 domain interface
SH2-Linker Loop interface
SH2-Kinase domain interface
C
SH2-SH3 domain interface
SH2-Linker Loop interface
SH2-Kinase domain interface
D
SH2-SH3 domain interface
SH2-Linker Loop interface
SH2-Kinase domain interface