PLOS | CURRENTS TREE OF LIFE

Cocos: Constructing multi-domain protein phylogenies

June 9, 2011 · Tree of Life

Max Homilius, John Wiedenhoeft[1], Sebastian Thieme[2], Christoph Standfuß[3], Ivan Kel[3], Roland Krause[4]

1 Doctoral Student, Teaching Assistant, Chemnitz, **2** Department of Vertebrate Genomics, Max Planck Institute for Molecular Genetics Berlin, Free University of Berlin, Germany, **3** Dept. of Computer Science, Free University of Berlin, Germany, **4** Dept. of Computer Science, Free University of Berlin, Germany, Berlin, Germany

Homilius M, Wiedenhoeft J, Thieme S, Standfuß C, Kel I, Krause R. Cocos: Constructing multi-domain protein phylogenies. PLOS Currents Tree of Life. 2011 Jun 9 [last modified: 2012 Apr 4]. Edition 1. doi: 10.1371/currents.RRN1240.

Abstract

Phylogenies of multi-domain proteins have to incorporate macro-evolutionary events, which dramatically increases the complexity of their construction.
We present an application to infer ancestral multi-domain proteins given a species tree and domain phylogenies. As the individual domain phylogenies are often incongruent, we provide diagnostics for the identification and reconciliation of implausible topologies. We implement and extend a suggested algorithmic approach by Behzadi and Vingron (2006).

Funding Statement

The authors have no support or funding to report

Introduction

Domains characterize proteins structurally, evolutionarily and functionally[1] . More than half of the proteins in prokaryotes and about 80 percent of the proteins in eukaryotes are composed of multiple domains[2] . About 200 domains in eukaryotes occur in diverse architectures[3] and provide a challenge for phylogenetic inference, as proteins can be composed of non-homologous elements. Evolutionary events such as the fusion of proteins or the loss of domains need to be considered in phylogenetic analyses of multi-domain proteins (MDPs).

Behzadi and Vingron put forward an iterative procedure (BV) to reconstruct ancestral domain compositions using the phylogenetic relationship within domain families[4] . See Figure 1 for an overview. Their algorithm minimizes the number of the macro-evolutionary events protein fusion and domain loss using a set-theoretic formulation and is independent of the order of the domains in the proteins.

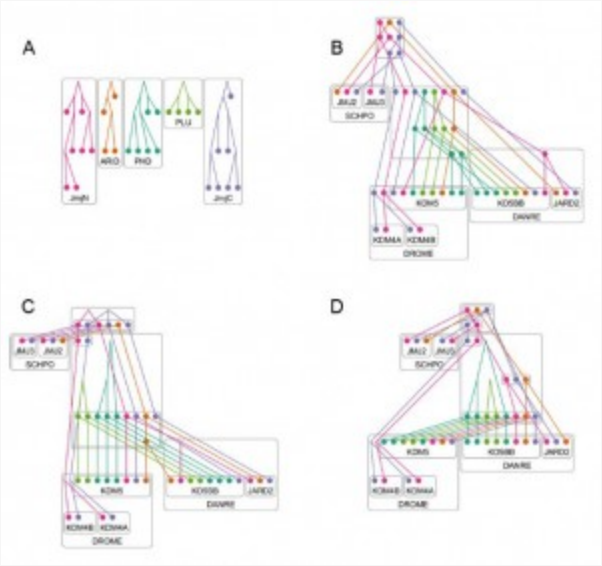

Figure 1. Example and principal approach. Reconstruction of the ancestral domain composition for JMJ-associated proteins in Drosophila melanogaster, Schizosaccharomyces pombe and Danio rerio . (A) Phylogenies are inferred for each domain. (B) Domain trees are embedded in the species tree. (C) Domains are partitioned into proteins. (D) Inconsistencies in the domain trees are curated.

The algorithm consists of the following steps:

1. Input a species tree, a set of domain trees and extant domain compositions, i.e. a partition of the domain trees' leaf set.
2. Recursively map domain tree nodes to species tree nodes at the least common ancestor (LCA) of all child domain nodes, starting at the leaf level.
3. Recursively walk through the species tree nodes (bottom-up). In all child species, the domains are already partitioned. Establish correspondence between domain nodes in the child species and those in the current species (relabeling of domain nodes), then find an optimal partition in the current species which is closest to all child partitions in terms of the weighted number of fusions and deletions.

Previous analyses of MDPs have decided against the use of phylogenetic trees for domains[5] or relied on establishing phylogenies only for domain trees with high internal bootstrap support[6] . Recently, an alternative approach for the reconstruction of MDPs including domain trees was proposed[7] .

We implemented BV, tested it and and identified critical issues that need to be addressed for successful reconstruction of phylogenies of MDPs using BV. Due to the large number of possible domain combinations a good set of partitions cannot be found by brute-force enumeration. We implemented a heuristic called weak edge erosion, which yields close to optimal solutions faster than simulated annealing suggested by[4] . In the practical application it showed that most domain trees are incongruent to each other and the species tree. We implemented a simple procedure to detect and rectify problematic cases.

In the following, we present our findings in detail, provide an implementation and show how to use it in practice. First, BV and the individual improvements are introduced formally.

Methods

The algorithm by Behzadi and Vingron

The algorithm BV uses the information of domain trees and the composition of extant proteins to infer the domain composition of ancestral proteins [4]. The original publication contains a worked example recommended for further study. Let S be a species tree in which xy is the parent species of x and y. Each species, e.,g. x, is assigned a set D^x of domains that belong to a domain family $A,B,\ldots,Z$. Let $R^x=\{P_1^x,\ldots,P_k^x\}$ be a partition of D^x called a domain composition family of domain compositions $P_1^x,\ldots,P_k^x$. As these are sets, the ordering of domains within a gene is ignored.

Phylogenetic trees are inferred for each domain family individually. Reconciling domain trees for each domain family using the known species tree assigns domain nodes to each species node xy in S. The nodes which have a direct child in at least one of the descendant species are the elements of D^{xy}. Let $\widetilde{D}^x$ be the set D^x relabeled in a way that each domain in x receives the name of its ancestral domain in xy, and let $\widetilde{R}^x$ be the according domain composition family. If there is a duplication event in x, the two domains map to the same ancestral domain in xy, thus $|\widetilde{D}^x|\leq|D^x|$. Since R is known only for the leafs, the problem is to find partitions for the inner nodes, i.e.domain composition families of ancestral species that minimize some cost

$$\delta(R^{xy},\widetilde{R}^x)+\delta(R^{xy},\widetilde{R}^y)$$

which can be arbitrarily defined.

Behzadi and Vingron suggest an additive measure given by

$$\delta(R^{xy},\widetilde{R}^x)=\sum_i\delta(R^{xy},\{\widetilde{P}_i^x\})$$

thus we solve

$$\arg\min_{R^{xy}}\left\{\sum_i\delta(R^{xy},\{\widetilde{P}_i^x\})+\sum_i\delta(R^{xy},\{\widetilde{P}_i^y\})\right\}$$

where $\delta(R^{xy},\{\widetilde{P}_i^x\})$ measures the number of deletions and merges that are necessary to transform the elements of R^{xy}, the domain compositions of the parent, to the child domain composition $\widetilde{P}_i^x$. Ignore all P_j^{xy} that do not contain any domain in $\widetilde{P}_i^x$ and store the indices of the remaining in

$$J:=\{j|P_j^{xy}\cap\widetilde{P}_i^x\neq\emptyset\}$$

The number of merges to have all domains of $\widetilde{P}_i^x$ in one set is therefore $|J|-1$. The resulting set could contain other domains that need to be deleted as they are not in the child domain composition; their number is

$$|\bigcup_{j\in J}P_j^{xy}-\widetilde{P}_i^x|=\sum_{j\in J}|P_j^{xy}-\widetilde{P}_i^x|$$

Assigning costs for union and deletion yields the partitioning score

$$S:=\delta(R^{xy},\{\widetilde{P}_i^x\})=c_U\cdot\underbrace{(|J|-1)}_{\#\text{ unions}}+c_D\cdot\underbrace{\sum_{j\in J}|P_j^{xy}-\widetilde{P}_i^x|}_{\#\text{ deleted elements}}t$$

The entire tree is reconstructed in a bottom-up pass including the root.

Heuristics for partitioning

As the number of possible partitions for domains into genes grows rapidly, complete enumeration is only

possible for the simplest cases and not suitable for real applications. For BV, it was suggested to use simulated annealing to solve the partitioning problem [4] . We explored a deterministic algorithm we call weak edge erosion (see below) to find a suitable partition

Weak Edge Erosion

Weak edge erosion is a hierarchical graph clustering method based on the idea of attacking a network of affinities between elements at its weakest points and recursively create clusters by separating network components. The affinity graph is defined as follows:

Consider an undirected loop-free graph G with a vertex set $V(G) := D^{xy}$. An edge between vertices v_1, v_2 is assigned a cost according to the number of sets in the child partitions that contain $\{v_1, v_2\}$ as a subset after relabeling. Note that each set induces a clique in this affinity graph. The edge weight measures the affinity of two vertices to occur together in a set. Thus removing edges corresponds to breaking the affinity between elements. To find a good approximation for the partitioning problem, we let $G := G_N$ store G_N in a tree node N, cut the graph and store the resulting components G_i in child nodes i of N. These nodes are processed recursively until the vertex set in a node scores 0 with the score S. Then the nodes are checked in a bottom-up pass as to whether their subset or the subsets of their children score better, and this solution is passed to the parent.

Cluster boundaries are induced by regions sparse in edges. But dividing a set to create partitions translates to cutting through a large number of edges in a clique, so minimal cuts and related concepts of connectivity are of limited use. Particularly, min-cut tends to separate single vertices from cliques, thus creating a suboptimal partition.

To obtain meaningful cuts for our purpose, we introduce the concept of weak edges. Let each vertex be labeled by the sum of weights of all its incident edges. From the perspective of an edge, high vertex weights mean that the vertices have a strong connection to a set of other vertices, so edges are regarded to become weaker as their vertex weights grow. We thus define a total order of weakness: let $e_m, e_n \in E(G)$ be two edges, and $v^-, v^+ \in V(G)$ be two vertices such that $w(v^-) \leq w(v^+)$. e_m is weaker than e_n , denoted by $e_m \prec e_n$, if the corresponding edge has a lower weight:

$$w(e_m) < w(e_n) \Rightarrow e_m \prec e_n$$

This first condition accounts for that we prefer weaker affinities to be violated. If the edge weights are equal, we want to exploit the weakening effect of high vertex weights, thus the weakness order is determined by the heavier vertices:

$$w(v_m^+) > w(v_n^+) \Rightarrow e_m \prec e_n$$

If these are equal as well, the relation between the lighter vertices decides:

$$w(v_m^-) > w(v_n^-) \Rightarrow e_m \prec e_n$$

The concept of edge weakness is illustrated in Figure 2. As the cost function is additive, we can find an approximate solution to the partitioning problem by splitting D^{xy} into nested P_N^{xy} recursively and combine the local results. We use Algorithm 1 to find a good partition R_N^{xy} of $V(G) = D^{xy}$ with a cost C_N. The weakest edges are removed until the graph is decomposed into two or more components. A cut tree of G is built such that a node contains G and its children contain the connected components. See Figure 3 for an example.

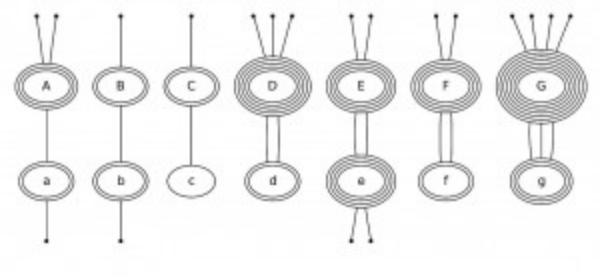

Figure 2. Example of edge weakness. Edges are sorted from left to right in decreasing order of weakness.

Weights are depicted by the multiplicity of lines. The edge (A,a) has a weight of 1, node A has a combined weight of incident edges of 3, and a of 2. The upper row contains the heavier nodes incident to the central edge. The lower the edge weight, the weaker this edge, e.g.

$$w(C,c)<w(D,d)\Rightarrow(C,c)\prec(D,d)$$

If edges tie, such as (A,a), (B,b) and (C,c), the heavier nodes decide:

$$w(A)>w(B)\Rightarrow(A,a)\prec(B,b)w(A)>w(C)\Rightarrow(A,a)\prec(C,c)$$

If the upper nodes tie as well, as for $w(B)=w(C)$, the weights of the lighter nodes decide:

$$w(b)>w(c)\Rightarrow(B,b)\prec(C,c)$$

Walking through the cut-tree in a bottom-up procedure, we decide whether the block in the parent node or the two set blocks in its children yield a better score S, and pass this local solution upwards until the root contains an approximate solution for the partitioning problem.

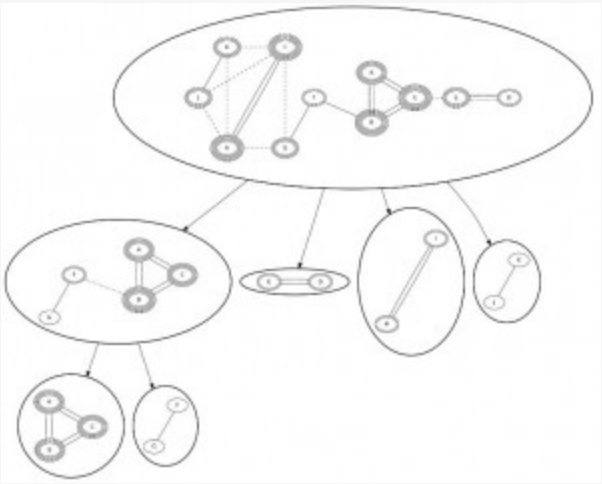

Figure 3. A worked example of a cut tree for the sets $\{A,B,C\}$, $\{A,B,C\}$, $\{C,E\}$, $\{D,E\}$, $\{D,E\}$, $\{B,F\}$, $\{F,G,H,I\}$, $\{H,I,J,K\}$

Each node represents a set block of the vertices it contains. Vertex weights are denoted by the number of peripheries. Note that this weight changes during the process according to the sum of incident edges (e.g. in vertex I). The weakest edges that are attacked during an erosion are denoted by dashed lines. Final erosions of the remaining cliques in the leafs are omitted, since they do not yield a better score. In the root, the weakest edges are $\{F,H\}$ and $\{F,I\}$, as their weights are 1, I and H have the maximum number of incident edge weights:

$$w(v^+)=w(I)=w(H)=6$$

and F is the maximum lighter vertex adjacent to I and H :

$$w(v^-)=w(F)=4$$

However, this does not suffice to disconnect the graph, so vertex weights are recalculated:

$$w(H)=w(I)=5,\ w(F)=2$$

and all edges with a weight of 1 are removed, $w(v^+)=5$ and $w(v^-)=3$. Relying on min-cut instead of erosion would fail to create set blocks $\{F,G\}$, $\{H,I\}$ and $\{J,K\}$, as it would cut off G , J and K as single vertices.

The complete procedure is summarized in Algorithm 1:

Algorithm 1 CREATECUTSUBTREE

Require: cut-tree node N, node-specific affinity graph G_N, relabeled domain composition families $\tilde{R}^x$ and $\tilde{R}^y$ of the child species

$R_N^{xy} \leftarrow \{V(G_N)\}$
$C_N \leftarrow \delta(R_N^{xy}, \tilde{R}^x) + \delta(R_N^{xy}, \tilde{R}^y)$
if $C_N > 0$ **then**
  while G_N is connected **do**
    erode weakest edges and recalculate the weights of affected vertices
  for all connected components G_i **do**
    add child i to cut-tree node n
    $(R_i^{xy}, C_i) \leftarrow$ CREATECUTSUBTREE(i, G_i)
  $R_I^{xy} \leftarrow \bigcup_i R_i^{xy}$
  $C_I \leftarrow \delta(R_I^{xy}, \tilde{R}^x) + \delta(R_I^{xy}, \tilde{R}^y)$
  if $C_N > C_I$ **then**
    $R_N^{xy} \leftarrow R_I^{xy}$
    $C_N \leftarrow C_I$
return (R_N^{xy}, C_N)

Simulated annealing

The original authors of BV propose simulated annealing to solve the underlying partitioning problem[8]. We implemented a dynamic cooling schedule, which sets most parameters automatically[9]. Starting from a valid domain configuration, the fusion and fission of groups of domains and the swap of individual domains are used to generate related domain compositions. The simulated annealing procedure is then used to minimize the score S of the domain composition.

Identification and curation of implausible domain trees

Due to their short length, the inferred domain phylogenies often disagree with each other and the species tree. The BV algorithm was proposed for ideal data and does not consider errors in the underlying domain topologies. A practical consequence of such errors are that the order of speciation and duplication events between adjacent domains do not agree. These conflicts can lead to duplicate nodes in the reconstructed composition, for example nodes that have successors within the same protein. The BV algorithm will then produce MDPs with a high partition score due to additional copies of the conflicting domains. Our algorithm aims to produce a partition with an improved score by applying nearest neighbor interchanges on the conflicting domain trees. For each modification the ancestral composition is reconstructed and the modification with the lowest score S is used as a correction.

Simulation of MDP phylogenies

Species trees were generated following the Yule-Harding model. An initial domain composition of three families in a single protein was placed in the root and passed to its children, whose protein domain compositions underwent evolutionary events of genes (fusion, duplication) and domains (gain, duplication, loss). Subtree pruning and regrafting (SPR) operations were applied on the domain trees to create perturbed input data.

Construction of domain phylogenies

For an empirical evaluation we ran global models of the PFAM database[10] with HMMER's hmmpfam and hmmalign [11] against the UniProt/Swiss-Prot database [12] to identify domains and construct alignments. Maximum Likelihood trees were inferred from the alignments using PhyML[13]. The trees were rooted using

Notung [14].

Results and Discussion

As we cannot obtain true ancestral multi-domain protein compositions, we relied on simulations to test the validity of the algorithm and to inspect its performance when errors are introduced. The practical performance was evaluated on known instances of MDPs, available on the accompanying website. A brief, non-trivial example is presented in Fig. 1.

Simulations

To assess the performance of the algorithm on controlled input, species and domain phylogenies were simulated. The reconstructed domain compositions were compared to the original compositions using the partition distance measure, which ranges between zero and the size of the compositions[15]. Most ancestral partitions can be reconstructed perfectly but not all events can be mapped correctly. For example, the gain of a new domain family and the subsequent loss in one of its immediate children cannot be reconstructed accurately. The high standard deviation suggests that there are a few compositions which differ very much from their simulated counterpart. SPR operations lower the quality of the reconstruction.

Taxa	0 SPR	1 SPR	2 SPR	5 SPR	10 SPR
10	0.06 (0.35)	0.10 (0.40)	0.13 (0.45)	0.23 (0.66)	0.34 (0.86)
30	0.09 (0.47)	0.11 (0.50)	0.13 (0.52)	0.16 (0.55)	0.22 (0.65)
50	0.09 (0.46)	0.09 (0.46)	0.10 (0.48)	0.13 (0.50)	0.16 (0.54)

Table 1. Average partition distance and standard deviation of reconstructed domain compositions. The simulated trees had 10, 30 and 50 taxa and up to ten SPR operations were applied. The distance was calculated between simulated and reconstructed domain compositions of all ancestral species and normalized by the number of taxa. Each experiment was repeated 50 times.

An automated correction of incongruent domain phylogenies is effective for cases with minor errors. Manual curation is advised if domain phylogenies cannot be inferred reliably. To this end, internal nodes with high scores are tagged for assessment.

Performance

The run times of our implementation are short. We simulated 50 replicates of input data containing 50 taxa and no regrafting operations and ran the implementation under Linux on an AMD 64 X2 3200+ processor. The erosion heuristic inferred a MDP reconstruction in 2.06s±0.25s, while the simulated annealing procedure took 153s±22s to achieve comparable solutions. In practice, the calculation of reliable domain families bounds the performance.

Application

JMJ domains are found in proteins involved in chromatin remodeling complexes often together with domain families such as ARID, PHD and PLUN-1 [16] . They provide a concise example, parts of which are presented in Fig. 1. Only three species were selected for display; using a larger number of species is advisable for the inference of individual domain phylogenies.

Conclusion

Our solution for the inference of phylogenies of multi-domain proteins provides a simple and easy-to-use interface. The weak edge erosion heuristic provides considerable speed-up over simulated annealing while maintaining comparable solution quality. Beyond the application on MDPs, such methods could be applied to reconstruction of partial homologous units such as bacterial operons or protein complexes. Future work will be directed at improving the quality of the tree reconciliation.

Availability and requirements

The program was successfully tested under Python 2.6 and 2.7 on Windows, Mac OS X and Linux. It receives input for species and domain trees as well as parameters in Nexus format. The output can be visualized using GraphViz. The source code with additional figures and examples can be found at http://virulence.molgen.mpg.de/cocos/ and is freely available under a BSD license.

Author contributions

Implementation: MH and JW. Weak edge erosion: JW. Domain analysis: ST, CS, RK. Example data: IK, CS. Simulations: MH and RK. Wrote the paper: MH, JW and RK.

Competing interests

The authors have declared that no competing interests exist.

References

1. Doolittle RF. The origins and evolution of eukaryotic proteins. Philos Trans R Soc Lond B Biol Sci. 1995 Sep 29;349(1329):235-40. PubMed PMID: 8577833.

2. Apic G, Gough J, Teichmann SA. An insight into domain combinations. Bioinformatics. 2001;17 Suppl 1:S83-9. PubMed PMID: 11472996.

3. Basu MK, Carmel L, Rogozin IB, Koonin EV. Evolution of protein domain promiscuity in eukaryotes. Genome Res. 2008 Mar;18(3):449-61. Epub 2008 Jan 29. PubMed PMID: 18230802; PubMed Central PMCID: PMC2259109.

4. Behzadi B, Vingron M. Reconstructing Domain Compositions of Ancestral Multi-domain Proteins. Lecture Notes in Computer Science, 2006; Volume 4205/2006, 1-10
REFERENCE LINK

5. Song N, Sedgewick RD, Durand D. Domain architecture comparison for multidomain homology identification. J Comput Biol. 2007 May;14(4):496-516. PubMed PMID: 17572026.

6. Forslund K, Henricson A, Hollich V, Sonnhammer EL. Domain tree-based analysis of protein architecture evolution. Mol Biol Evol. 2008 Feb;25(2):254-64. Epub 2007 Nov 19. PubMed PMID: 18025066.

7. Wiedenhoeft J, Krause R, Eulenstein O. The Plexus Model for the Inference of Ancestral Multi-Domain Proteins. IEEE/ACM Trans Comput Biol Bioinform. 2011 Jan 27. [Epub ahead of print] PubMed PMID: 21282868.

8. Finn RD, Mistry J, Tate J, Coggill P, Heger A, Pollington JE, Gavin OL, Gunasekaran P, Ceric G, Forslund K, Holm L, Sonnhammer EL, Eddy SR, Bateman A. The Pfam protein families database. Nucleic Acids Res. 2010 Jan;38(Database issue):D211-22. Epub 2009 Nov 17. PubMed PMID: 19920124; PubMed Central PMCID: PMC2808889.

9. Huang, MD and Romeo, F. and Sangiovanni-Vincentelli, A. An efficient general cooling schedule for simulated annealing. Proceedings of the IEEE International Conference on Computer-Aided Design 1986.

10. Eddy SR. A new generation of homology search tools based on probabilistic inference. Genome Inform. 2009 Oct;23(1):205-11. PubMed PMID: 20180275.

11. UniProt Consortium. The Universal Protein Resource (UniProt) 2009. Nucleic Acids Res. 2009 Jan;37(Database issue):D169-74. Epub 2008 Oct 4. PubMed PMID: 18836194; PubMed Central PMCID: PMC2686606.

12. Guindon S, Gascuel O. A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood. Syst Biol. 2003 Oct;52(5):696-704. PubMed PMID: 14530136.

13. Chen K, Durand D, Farach-Colton M. NOTUNG: a program for dating gene duplications and optimizing gene family trees. J Comput Biol. 2000;7(3-4):429-47. PubMed PMID: 11108472.

14. Gusfield D. Partition-distance: A problem and class of perfect graphs arising in clustering, Information

Processing Letters, Volume 82, Issue 3, 16 May 2002, Pages 159-164, ISSN 0020-0190, DOI: 10.1016/S0020-0190(01)00263-0.
REFERENCE LINK

15. Balciunas D, Ronne H. Evidence of domain swapping within the jumonji family of transcription factors. Trends Biochem Sci. 2000 Jun;25(6):274-6. PubMed PMID: 10838566.

16. Chen K, Durand D, Farach-Colton M. NOTUNG: a program for dating gene duplications and optimizing gene family trees. J Comput Biol. 2000;7(3-4):429-47. PubMed PMID: 11108472.
REFERENCE LINK