High throughput prediction of inter-protein coevolution

Vid Sustar

Master´s Thesis

Master's Degree Programme in Digital Health and Life Sciences

Department of Future Technologies

University of Turku

October 2018

Abstract

UNIVERSITY OF TURKU

Department of Future Technologies/Faculty of Science and Engineering

SUSTAR, VID: High throughput prediction of inter-protein coevolution

Master´s thesis, 43 p., 7 p. appendices

Bioinformatics

October 2018

Inter-protein co-evolution analysis can reveal in/direct functional or physical protein interactions. Inter-protein co-evolutionary analysis compares the correlation of evolutionary changes between residues on aligned orthologous sequences. On the other hand, modern methods used in experimental cell biological research to screen for protein-protein interaction, often based on mass spectrometry, often lead to identification of large amount of possible interacting proteins. If automatized, inter-protein co-evolution analysis can serve as a valuable step in refining the results, typically containing hundreds of hits, for further experiments. Manual retrieval of tens of orthologous sequences, alignment and phylogenetic tree preparations of such amounts of data is insufficient. The aim of this thesis is to create an assembly of scripts that automatize high-throughput inter-protein co-evolution analysis.

Scripts were written in Python language. Scripts are using API client interface to access online databases with sequences of input protein identifiers. Through matched identifiers, over 85 representative orthologous sequences from vertebrate species are retrieved from OrthoDB orthologues database. Scripts align these sequences with PRANK MSA algorithm and create corresponding phylogenetic tree. All protein pairs are structured for multicore computation with CAPS programme on CSC supercomputer. Multiple CAPS outputs are abstracted into comprehensive form for comparison of relative co-adaptive co-evolution between proposed protein pairs.

In this work, I have developed automatization for a protein-interactome screen done by proximity labelling of B cell receptor and plasma membrane associated proteins under activating or non-activating conditions. Applying high-throughput co-evolutionary analysis to this data provides a completely new approach to identify new players in B cell activation, critical for autoimmunity, hypo-immunity or cancer. Results showed unsatisfying performance of CAPS, explanation and alternatives were given.

KEYWORDS: inter-protein, co-evolution, interaction, high-throughput, B cell activation

Acknowledgements

I would like to thank my supervisors Martti Tolvanen and Pieta Mattila for introducing me to the exciting topic of my thesis. I especially thank Martti for his time, advices and spreading his enthusiasm and motivation in the field of evolutionary bioinformatics. I thank Pieta for her advices in the field of Immunology and overall support of my bioinformatics studies. I am grateful to late professor Mario A. Fares and to dr. Brian Caffrey, the creators of CAPS for their guiding instructions for parallel usage of CAPS. I would also like to acknowledge CSC – IT Center for Science, Finland, for generous computational resources.

Special thanks goes also to colleagues Luqman Awoniyi for provision of MS protein hits and Petar Petrov for sharing his extensive CAPS and Linux experience. I also thank all other work colleagues that had to endure my long explanations of the master thesis.

Last, but not least I thank my life companion Mira for patience and allowing me to sacrifice some of our family life for my pursuit of bioinformatics.

Contents

1 Introduction

1.1 B cell activation

Through evolution multicellular organisms had to create protection against invading microorganisms by developing specialised immune cells. The oldest immune system is the innate immune system in the form of macrophages that swallow invading bacterial pathogens and natural killer cells that destroy virally infected own cells (Janeway, 2001).

Innate immune system reacts immediately, but may be insufficient. Improved effectivity of immune system against recurring pathogens was achieved through adaptive immune system which works hand in hand with innate immunity. Adaptive immunity appeared at around 500 million years ago in vertebrate animals. From this point on in evolution cells of innate system such as macrophages swallow pathogens and present antigens (hence antigen presenting cells) to cells found in special secondary lymphoid organs (Batista, 2009, Figure 1A). These cells are called B and T cells, some of which poses unique cognate antigen receptors (Flajnik, 2010). B cells with matching receptors are activated and undergo clonal expansion into memory and antibody producing plasma B cells, adapted to specific antigen carrying pathogen. Upon binding, antibodies can either activate macrophages or directly incapacitate the pathogenic process depending on the type of the antigen and antibody (Borghesi, 2006, Figure 1B).

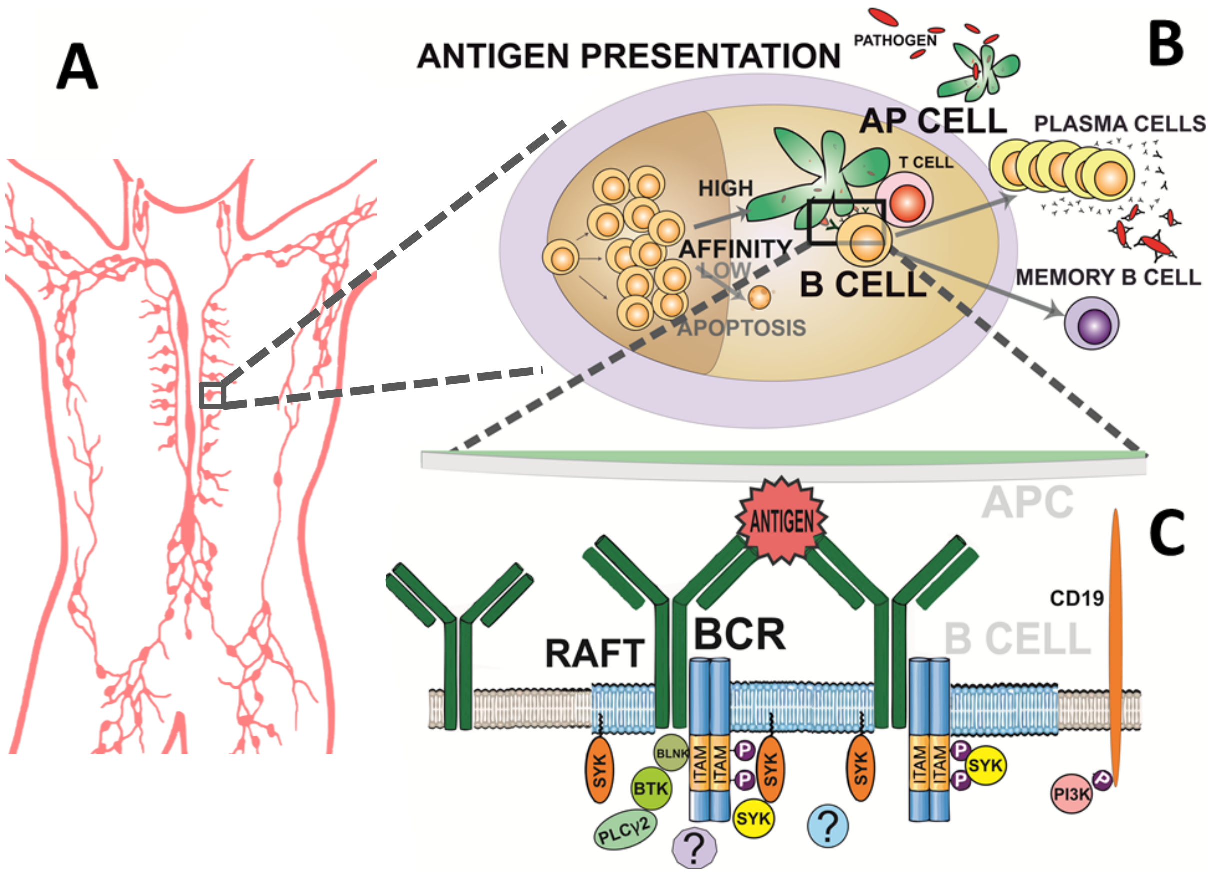


Figure 1. Presentation of the importance of B cell activation. (A) The location of B cells in lymph nodes (violet)[1] with (B) magnification where antigen presenting cells (APC) present the antigen to cognate few B cells that are expanded to memory B cells and antibody releasing B cells to fight the antigen carrying bacteria. [2] (C) magnification of APC-B cell interface; BCR relocation to lipid rafts after engagement with antigen, signalosome formation that leads to above B cell differentiation. (C) adapted from (Slupsky, 2015).

---

[1]https://visualsonline.cancer.gov/retrieve.cfm?imageid=3237&dpi=300&fileformat=jpg
[2]https://upload.wikimedia.org/wikipedia/commons/6/60/Germinal_center.svg

B cell activation is a crucial and finely tuned process in this sequence of events. B cell receptors (BCR, membrane bound immunoglobulin) are residing in plasma membrane (PM). Plasma membrane is composed of subdomains of distinct compositions and characteristics. Thicker, less fluid subdomain enriched with glycosphingolipids, saturated long fatty acid phospholipids and cholesterol and long transmembrane region proteins are called lipid rafts. IgM type BCR upon binding of antigen transition from non-raft into raft PM subdomains, where they aggregate into clusters and transduce the signal into cell interior via BCR intracellular α/β subunits. BCR clustering activates the Src family kinases Lyn, Syk and Btk tyrosine kinases. 'Signalosome' is formed composed of the BCR, tyrosine kinases, adaptor proteins CD19 and BLNK, and signalling enzymes PLCγ2, PI3K, and Vav. Membrane phospholipid phosphoinositide (4,5)P2 (PIP2) serves as a docking place for cortical actin adapter proteins. PLCγ2 cuts PIP2 into inositol 1,4,5-trisphosphate (IP3), which opens intracellular calcium storages and diacylglycerol which activates protein kinase C (Figure 1C). Overall, BCR signalosome activates multiple signalling cascades that involve kinases, GTPases, and transcription factors which affect cell metabolism, gene expression, and cytoskeletal reorganization (Dal Porto, 2004)

When B cell activation is abnormally upregulated it can lead to increased propagation of B cells expressed in several types of cancer: non/Hodgkin, follicular, lymphocytic , Burkitt's, or diffuse large B-cell lymphomas. It can also lead to autoimmune diseases: multiple sclerosis, rheumatoid arthritis and systemic lupus erythematosus. When B cells are underactivatable it can lead to immune deficiencies, type 2 diabetes and periodontal disease (Carter, 2006).

Understanding B cell activation through identification of the involved proteins and their interactions in pathways is therefore crucial in fighting these diseases.

1.2 Identification of new protein involved in B cell activation

One approach of identifying involved proteins is by mass spectrometry. Mass spectrometry is a technique for measuring the relative charge and mass of charged particles, in this case proteins. Proteins are purified from lysed cells and digested by proteases into short peptides. Peptides are transformed into gas phase and charged through ionisation by lasers or electro spraying. Charged peptides are accelerated through nonlinear electric or magnetic field toward detector. Detector detects the time of flight of the peptides, where smaller peptides are faster (Wysocki, 2005). The output is spectra specific to peptide sequence fingerprints found in databases from which peptides can be identified. In case of tandem mass spectrometry each peptide is further fragmented into an array of sub-peptides which are also analysed. From the array of sub-peptide spectra it is easier to directly determine the amino acid sequence of a peptide. Proteins are identified through statistically significant presence of their characteristic peptides in the acquired data.

Human B cells contain tens of thousands of proteins in different compartments performing their own functions. Only small subset of these proteins is involved in B cell activation. It is desirable to narrow down the array of proteins fed to MS as closest as possible to the B cell activation subset. Isolating - pulling down of only a narrow subset of proteins is possible via proximity labelling of proteins. Example of proximity labelling is usage of ascorbate peroxidase (APEX) bound to compartment of interest via lipid tails of compartment characteristic protein. In presence of hydrogen peroxide, APEX converts biotin phenol into reactive radicals that react and bind to proteins within a radius of 50 nm in about a minute, which allows not only spatial, but also temporally controlled biotinylation of proteins near APEX (Rhee, 2013). Biotinylated proteins can be selectively pulled down from cell lysate by binding to streptavidin coated magnetic beads. The biotin-streptavidin bound proteins are then washed of by higher ionic strength solution and further purified by gel electrophoresis. Purified protein sample from gel electrophoresis is then processed for MS. One can target APEX to raft or nonraft membrane compartments by adding myristoyl-palmitoyl-palmitoyl or just myristoyl lipidation sequence (respectively) to APEX sequence. Myristoyl fatty acid chain is shorter than palymitoyl chain which predominantly resides in thicker lipid raft regions of plasma membrane, into which IgM BCR shift upon B cell activation. Since also temporal control of APEX reactions is possible one can study the transition of proteins between non-raft and raft regions in the course of B cell activation.

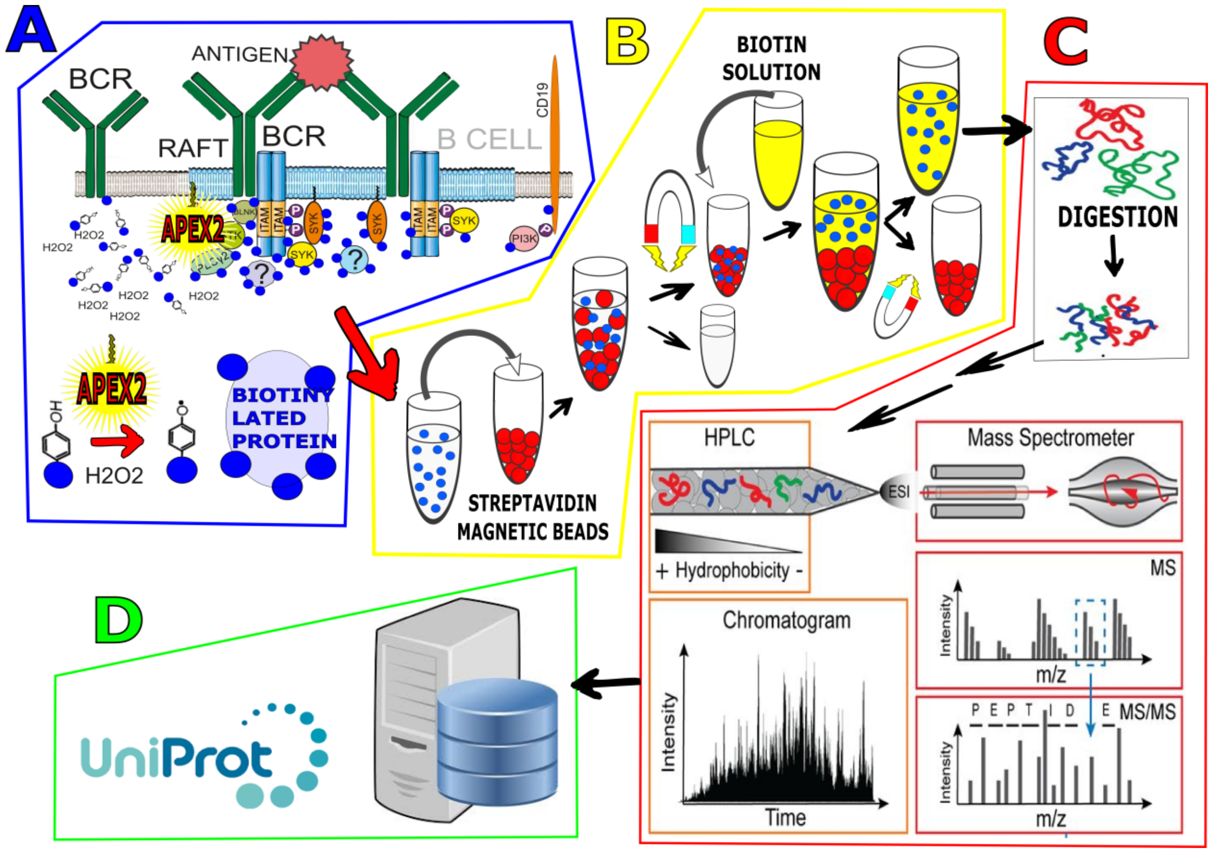


Figure 2. The pipeline of proximity based labelling and protein identification. (A) Specific cell compartment targeting APEX in the presence of hydrogen peroxide converts biotin phenol into biotin phenol radicals that react and label nearby proteins. (B) Biotinylated proteins are pulled down and washed from cell lysate with streptavidin coated magnetic beads. (C) Biotinylated proteins are processed for and analysed with protein mass spectrometry[5] (D) Finally the proteins are identified in databases based on MS peptide sequences. Adapted from (He(Heap, 2017).

1.3 Implied functional relationship of activation proteins through co-evolution analysis

By identification of proteins close to compartments where B cell activation occurs one cannot yet truly confirm their true involvement in the process of B cell activation or what kind of interaction with other known B cell activation proteins they might have. To further refine the results of MS and find more exact roles of all identified proteins one can employ different biochemical techniques to identify more stable interactions. Other way is to use different fluorescent microscopy techniques, including FRET where 10 nm range inter-protein proximity is detected. However, microscopic techniques, especially FRET, are typically highly laborious and cannot be applied to very many MS hits. Thus, further narrowing down of the MS hits is required, and is currently often done subjectively based on other existing data or literature. An unbiased refinement to objectively narrow down the list of hits to most potential new players would be essential and would critically facilitate new interesting findings and research lines.

With the advent of increasing computer capabilities and ever larger biological sequence databases there is also a greater array of bioinformatics tools that can be much faster and cheaper in comparison to conventional wet-lab methods. Among these tools is inter-protein co-evolution analysis, which can inform us about directly physically or at least indirectly functionally interacting proteins. The following section (2 Background) presents the theoretical background of protein co-evolution analysis.

1.4 Aim of the thesis

The aim of this thesis is to create a script or assembly of scripts that will:

- automatize and scale up the retrieval of orthologous sequences for hundreds of proteins identified from mass spectrometry or elsewhere

- create multiple sequence alignments and phylogenetic tree presentations for the retrieved orthologues of each protein

- Prepare these as input files for multicore parallel run of CAPS – coevolution analysis programme on supercomputer

- combine CAPS multiple output files into comprehensible readout of adaptive co-evolution

Together, the generated toolbox is aimed to significantly facilitate the narrowing down the lists of potential interesting proteins in a given high-throughput approach to only those proteins that show co-evolution and, thus, are more likely to function together in a given cellular pathway.

2 Background

2.1 Co-evolution

Co-evolution is an inseparable phenomenon to the evolution of species through selective environmental pressure. Part of environment are other species. When two or more species form interacting dynamic ecological relationship, a change in one species demands adaptive evolutionary response from interacting species. Although the term co-evolution was first used by Ehrlich and Raven in 1964 (Ehrlich, 1969), actually already Darwin, the father of theory of evolution, presented this idea of mutually influencing species in 1862 with an example of orchid blossoms and the corresponding lengths of the tongues of their insect pollinators (Darwin, 1862). Other more general examples of closely interacting pairs of species are predator and prey, symbionts, parasite-host etc.

In 20th century also the biochemical basis of evolution became evident with the discovery of deoxyribonucleic acid (DNA) as a carrier of inheritable adaptive change through generations (Watson, 1953). Practically every known non-viral organism contains DNA. DNA is coding thousands of proteins - molecular machinery that performs a vast array of functions in each cell forming an organism (Crick, 1958).

2.2 Protein co-evolution

Most proteins interact with other proteins to achieve higher functions (De Las Rivas, 2010). As there is co-evolution of closely interacting species one can naturally expect some form of co-evolution in closely interacting proteins, from which lastly organisms and species arise.One can further define co-evolution as similarity in evolutionary histories measurable by similarity of phylogenetic trees and introduce a term co-adaptation as a co-evolutionary influence of one protein on another's evolutionary history (Pazos, 2008). Co-adaptation requires and is therefore informing us of a close relationship between proteins. Besides co-adaptation the other major source of protein co-evolution is phylogenetic coevolution.

If one wants to reveal inter-protein interactions, the phylogenetic co-evolution of proteins needs to be separated from co-adaptive co-evolution (Pazos, 2013).

Protein co-evolution can be measured on several levels. Proteins are a few tens to few thousands long sequences of amino acids, where each site can be composed of 20 different amino acids (Brocchieri, 2005). Each amino acid is coded by a codon, a sequence of 3 amino acids, composed of 4 possible different amino acids (Crick, 1958, Nirenberg, 1961). Each of 20 different amino acids contains distinct side chains that give them characteristic polarity, hydrophobicity and size which affect the interaction between amino acids in a protein chain, forming a distinct folding and final shape and function of the protein. A change or a mutation in the composition of the amino acids can have varying effects on the shape and functionality of a protein depending on the similarity of the replaced amino acid and the position of the change (King, 2013).

When for example positively charged amino acid is replaced by negatively charged amino acid, it will repel nearby negatively charged amino acids that may be in the same protein or in the interacting protein. Such a change can be compensated, if the interacting amino acids in turn also mutate to opposing charges, this is a case of co-adaptation (Gobel, 1994, Mateu, 1999, Shim Choi, 2005).

Co-adaptive correlated change can be detected in multiple sequence alignment (MSA) (Pazos, 2008). MSA is as its name implies an alignment of three or more evolutionary related sequences, where each sequence is an orthologous sequence from separate species and equivalent residues are placed in the same column. This way the evolutionary changes in a residue can be easily tracked along the column. If there is a simultaneous change in evolution, possible co-adaptation of interacting proteins, it can be recognised in concurrent correlated changes in another residue at the same stage of evolution, across many species in the MSA (Pazos, 1997). One can calculate the correlation in the change of two residues in a weighed blossum matrix accounting for the intensity of the change (ie. negatively charged AA into positively charged vs hydrophobic AA into other hydrophobic AA) (Fares, 2006). In order to be able to recognise concurrent changes in a residue it is critical to obtain proper MSA with no misalignments.

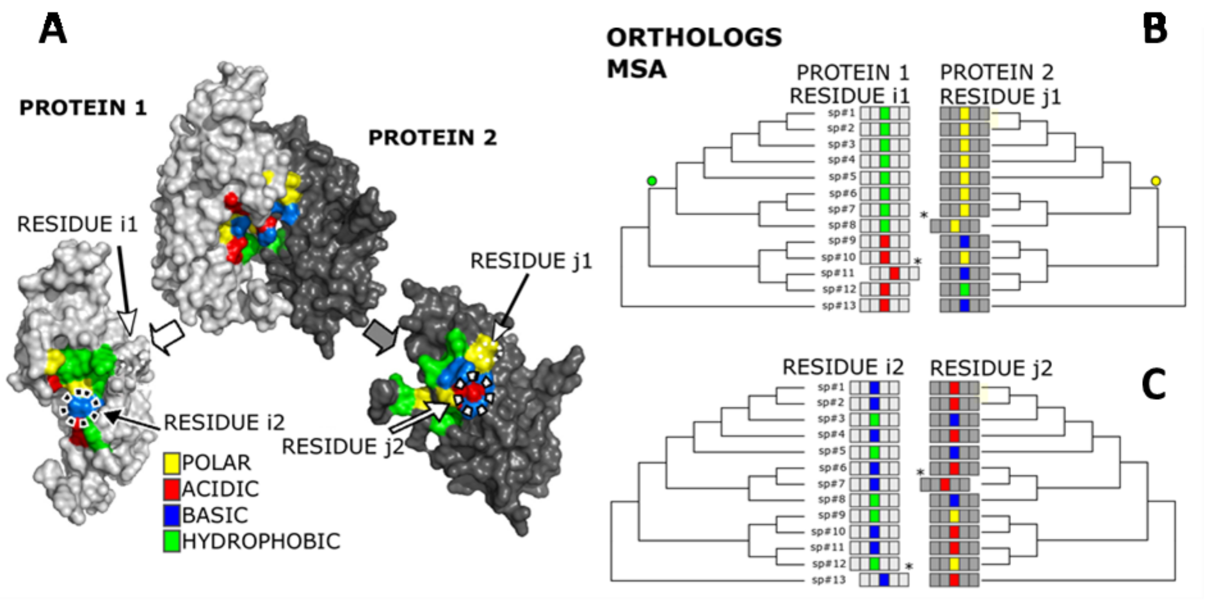


Figure 3.

Example of coevolving residues in two interacting proteins. (A) 3D structure of proteins with marked selected residues and their properties legend. (B,C) orthologous sequences alignments with selected residues and their correlation. (B) Residues i1 and j1 were subjected to phylogenetic co-evolution whereas (C) residues i2 and j2 were subjected to co-adaptive co-evolution that shows co-evolution even if we remove random clade. Asterisks show the misalignments to emphasize importance of their avoidance. (Adapted from Madaoui, 2008)

2.3 Orthologous genes and multiple sequence alignment (MSA)

The sequences required for MSA are obtained from orthologue databases. The orthologues databases are built based on algorithms that compare the homologous sequences of DNA genes of different species. Gene duplications (paralogues) or deletions through evolution result in several or false orthologue candidates (Graur, 2000). In addition there is a possibility of different protein splice variants in different species. With the automatization and upscaling of genomics the complexity of algorithm based determination of orthologues in genomic data and the possibility for their inaccuracies has also increased (Jensen, 2001).

The algorithms for ortholog identification identify genes in the genomic data by either *de novo* methods that use transcriptional initiation, termination and splice patterns or by *trans-*

alignment methods where cDNA sequences of best annotated species (human) are used for comparative search of similar gene sequences (Jensen, 2001).

Multiple Sequence alignment algorithms are used for alignment of acquired orthologous sequences in order to quantify the evolutionary conversion of individual AA residues – as a column in MSA (Figure 3, B,C). There are several MSA algorithms each with its own benefits and drawbacks. Clustal omega is an improved iteration of Clustal MSA algorithms and is suitable for medium-large alignments. It uses HHalign method, seeded guide trees and hidden markov models profile-profile techniques to generate alignments (Sievers, 2011). T-Coffe is suitable for small alignments. T-Coffe stands for Tree-Based Consistency Objective Function for Alignment Evaluation and attempts to overcome the problems of progressive alignment methods (Notredame, 2000). Muscle is used for medium sized alignments. Muscle stands for MUltiple Sequence Comparison by Log-Expectation (Edgar, 2004).

PRANK optimally estimates insertions and the number of deletion events and can therefore be also used for more distantly related sequences. Unlike other alignment programs PRANK uses maximum likelihood methods used in phylogenetics and correctly estimates evolutionary distances between sequences (Löytynoja, 2008). The drawback of PRANK is long running time. PRANK uses guide phylogenetic tree to distinguish insertions from deletions and any errors in the tree will result in errors in alignment (Löytynoja, 2014).

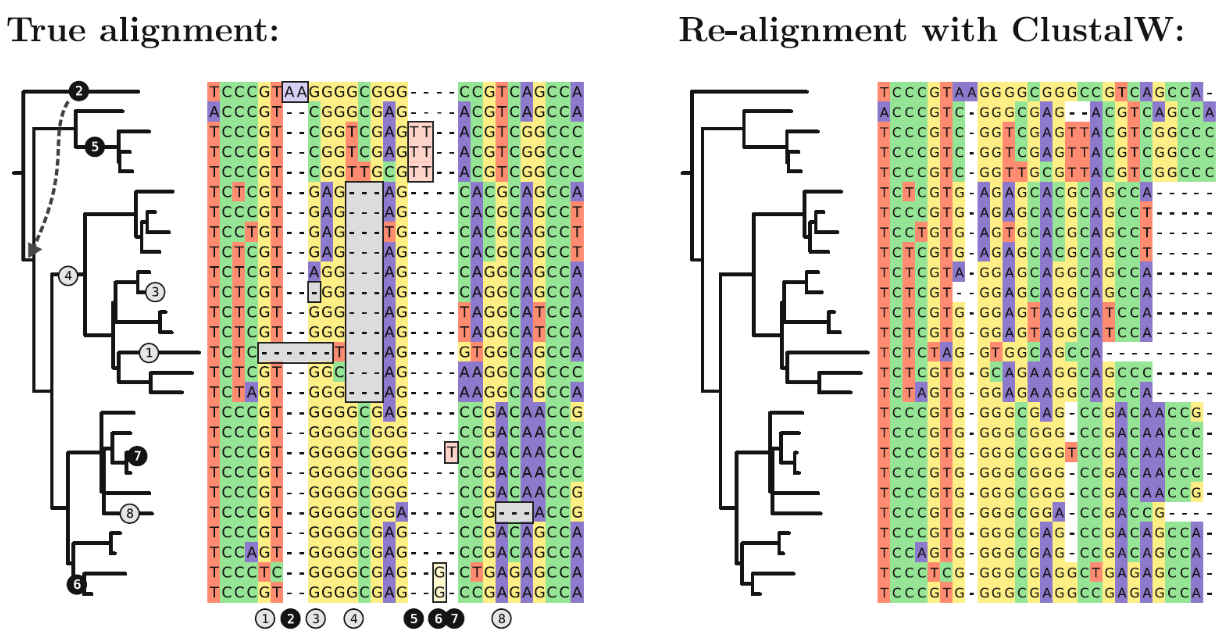


Figure 4. Comparison of phylogeny aware PRANK MSA algorithm (left) and the classical progressive algorithm ClustalW (right). The framed patterns in sequences correspond to the numbered evolutionary events of insertions and deletions on the phylogenetic tree on the left side. In comparison the classical progressive alignment algorithm on the right does not resolves insertion and deletions correctly (Löytynoja, 2014).

2.4 Methods for protein residue co-evolution quantification

Most basic co-evolution prediction algorithms are based on interdependent amino acid frequencies or the detection of similar patterns of amino acid substitutions in two columns of the MSA - pairs of residue positions. These are inter-residue co-evolution analysis algorithms, which can be used beside inter-protein analysis also for intra-protein co-evolution analysis, that inform us of closely interacting residues within a protein 3D structure. The similarity in the AA substitution patterns can be calculated directly by linear correlation. Most common name for this approach is **McLachlan-based substitution correlation (McBASC)** (Göbel, 1994). The drawback of this approach is that is does not take into account background phylogenetic divergence. The **co-evolution analysis using protein sequences (CAPS)** uses phylogenetic information to check and qualify only the correlations that persist when individual clades are removed from original MSA. The drawback of CAPS in comparison to McBASC is high computational demand.

Mutual information is another approach, which instead of calculating substitution correlations, uses all amino acid frequencies within one residue position to predict the probability of amino acid of another residue and vice versa.

Residue correlation usually does not happen only in pairwise fashion but forms a network. This is approached by **Direct coupling analysis (DCA)** which creates a global statistical model based on position-specific variability and inter-position coupling in MSA (Weigt, 2009). DCA model is transformed into mutual information-based formulation with Heuristic methods. Another set of analytical algorithms for residue co-evolution is focused on finding groups of residues that have a protein family-dependent conservation pattern, which also shows correlated mutational patterns. These residue positions are named **specificity-determining positions (SDP**s). SDPs mutate in coordinated fashion in the context of subfamily divergence. For example in the duplication of an enzyme the subfamily of these proteins can adapt to a new set of substrates by mutating just a group of residues that often form 3D clusters on the interfaces between enzyme-substrate or receptor-ligand. There are several methods to detect SPDs. **Sequence Space** uses principal component analysis (PCA) vector representations of the MSAs to detect AA (SDP) patterns of protein subfamilies and is the basis for other methods that detect SDPs in MSAs (Casari, 1995). **S3det**, automatizes detection of protein subfamilies and SDPs in MSAs. Through automatization and upscaling it was possible to show the relation between SDPs and binding sites of interactors and substrates (Rausell, 2010). **Mutational behaviour (MB)** correlates residue substitution in MSAs to whole sequences variation (del Sol Mesa, 2003). **Evolutionary Trace (ET)** uses similarities of the sequences that split in each tree branch and measures the time these similarities become conserved. The distance in time from origin of the phylogenetic tree to the fixation of conserved sequences is used to determine subfamilies. Differential sequence conservation analysis informs of SDPs – protein binding sites. This way MSAs do not need to be perfectly partitioned into protein subfamilies (Lichtarge, 1996, Mihalek, 2004). **Combinatorial entropy optimization (CEO)** uses combinatorial exploration to first create optimal partitioning of subfamilies, while residue entropies are used to determine SDPs (Reva, 2007).

Similar to SPD searching algorithms there are also **statistical coupling analysis (SCA**) algorithms which don´t require specificity of found residues to certain protein subfamilies (Lockless, 1999).

Co-evolution can be also measured at protein level. It was observed that for example ligand and receptor pairs tend to have similar phylogenetic trees. **MirrorTree** calculates inter-orthologue distance matrices from the MSA-derived phylogenetic trees or directly from the MSAs. Linear correlations between distance matrices serve as approximation of phylogenetic similarities between two proteins (Pazos, 2001). **Tree of life - Tol Mirror tree** is a method that uses evolutionary relationship between species to reduce this type of tree similarity background in inter-orthologue distance matrices (Pazos, 2005). **ContextMirror** on the other hand uses information from the whole proteome of interest to compensate for this and other possible types of backgrounds (Juan, 2008). Mirror tree methods depend strongly on the selection of species in the tree. This problem can be circumvented by automatic selection of species where co-evolution is the highest as by **Matrix Match Maker (MMM)** (Tillier, 2009). **Phylogenetic profiles** uses simple similarity of the presence or absence patterns of the two proteins in different species to infer inter-protein co-evolution (Pellegrini, 1999).

It is also possible to combine both intra and inter protein residue correlations to estimate possible interactions between proteins, as does for example **in silico two-hybrid (i2h)** method (Pazos, 2002)

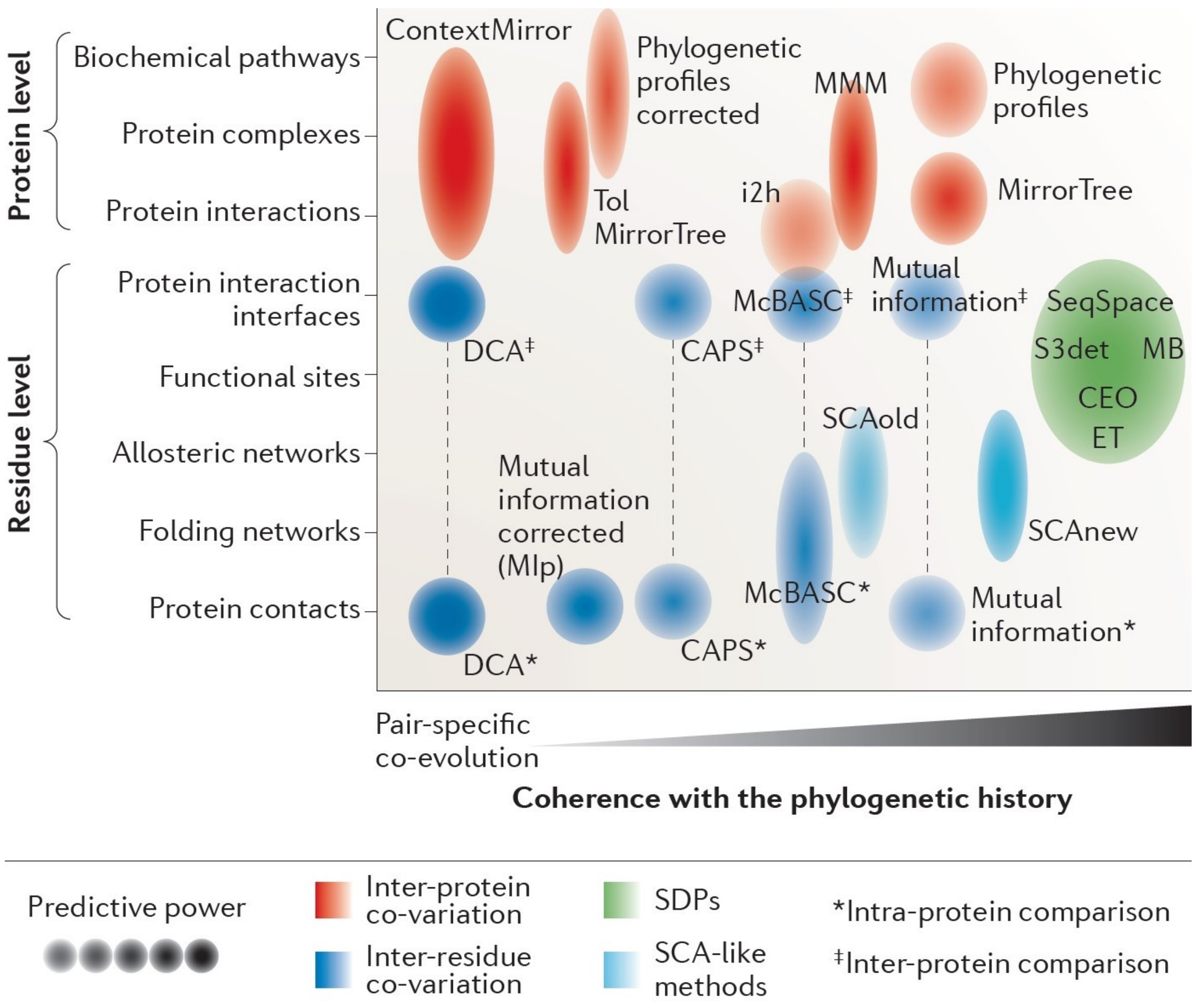


Figure 5. Comparison of different co-evolution analysis methods. Coloured by main categories as covered in the text. Adapted from (Juan, 2013).

2.5 Co-evolution analysis using protein sequences (CAPS)

In this thesis, co-evolution analysis using protein sequences (CAPS) has been chosen as the method to estimate inter-protein co-evolution. CAPS is a conservative algorithm for detection of residue coevolution that can disentangle interaction from stochastic and phylogenetic co-evolution. As it is based on residue co-evolution it can be used for intra or inter protein co-evolution (Fares, 2006). When used for inter protein co-evolution one can of course determine exactly which parts of two molecules were co-evolving or in case of physical interaction directly interacting. The drawback of CAPS is that computation time is exponentially dependent on the length of the sequences – the number of residues in the alignment, since it calculates correlations for each of the possible pairs of residues.

Residues in MSA of orthologous sequences can undergo mutational transition (as in Figure 3 B, C). The variance of the transitions can be used to infer correlations – co-evolution between two residue sites. CAPS calculates the residue conversion variance correlations. It corrects the correlations for the time of each species (sequence) divergence.

Following is the mathematical basis of the CAPS algorithm. More in depth explanation can be found in original article (Fares, 2006a). CAPS compares the conversion probabilities between two residues using Blocks Substitution Matrix (BLOSUM; Henikoff, 1992). BLOSUM substitution matrices account for divergence probabilities between each AA. However, there are cases like pre-speciation duplications that fixate certain residues into more conserved than more recent sequence divergences. To take this into account the probability of residue conversion from e to k amino acid from sequence i to j – $(\theta_{ek})_{ij}$ is based on Blosum e->k conversion probability (B_{ek}) divided by time since divergence between the sequences in question as in (1):

$$(\theta_{ek})_{ij} = (B_{ek}t^{-1})_{ij} \qquad (1)$$

From this, the average probability for all the sequences (whole column in Figure 3 B, C) is calculated (2)

$$\overline{\theta}_C = \frac{1}{T}\sum_{S=1}^{T}(\theta_{ek})_S \qquad (2)$$

,where S is each pairwise comparison and T is the total number of pairwise sequence comparisons.
T is calculated as (3) where N is the total number of sequences in the alignment:

$$T = \frac{N(N-1)}{2} \qquad (3)$$

Variability of each transition is calculated as the squared difference of transition probability to average probability as in (4):

$$\widehat{D}_{ek} = \left[(\theta_{ek})_{ij} - \bar{\theta}_C\right]^2 \tag{4}$$

From this the mean variability is calculated as in (5):

$$\overline{D}_C = \frac{1}{T}\sum_{S=1}^{T}[(\theta_{ek})_S - \bar{\theta}_C]^2 \tag{5}$$

To assess co-evolution between residue positions A and B (as between i1 and j1 or i2 and j2 in Figure 3B and Figure 3C) the correlation ρ_{AB} in AA variability relative to mean variability is calculated for each residue transition (as between instances within the columns i1 and j1 column of Figure 3 B, C). The correlation ρ_{AB} is calculated as in (6):

$$\rho_{AB} = \frac{\frac{1}{T}\sum_{S=1}^{T}[(\widehat{D}_{ek})_S - \overline{D}_A][(\widehat{D}_{ek})_S - \overline{D}_B]}{\sqrt{\frac{1}{T}\sum_{S=1}^{T}[(\widehat{D}_{ek})_S - \overline{D}_A]^2 \frac{1}{T}\sum_{S=1}^{T}[(\widehat{D}_{ek})_S - \overline{D}_B]^2}} \tag{6}$$

To estimate whether that particular ρ_{AB} correlation is significant, correlations for K number of re-sampled random pairs of sites (different columns in Figure 3 B, C) is calculated (1-6) and mean $\bar{\rho}$ (7):

$$\bar{\rho} = \frac{1}{K}\sum_{l=1}^{K}\rho_l \tag{7}$$

and $V\,(\rho)$ variance of correlation coefficients are calculated (8):

$$V\,(\rho) = \frac{1}{K}\sum_{l=1}^{K}(\rho_l - \bar{\rho})^2 \tag{8}$$

The test of significance is made by comparing the individual correlation coefficients? to normalised Z distribution as in (9):

$$Z = \frac{\rho_{AB} - \bar{\rho}}{\sqrt{V(\rho)}} \tag{9}$$

Statistical power of the test is optimised by all the sites that have:

$$\overline{D}_C > \Theta - 2\sigma_\Theta \tag{10}$$

Θ stands for parametric value of $\overline{D}_C$ and is calculated as:

$$\Theta = \frac{1}{L}\sum_{s=1}^{L}(\overline{D}_C)_s \tag{11}$$

whereas σ_Θ is the standard deviation of Θ.

There are several contributors to co-evolution: stochastic covariation, inter-molecular interaction and phylogenetic convergence. The goal is to filter out all but the inter-molecular interaction components.

Stochastic covariation is removed by testing for significance of the simulated data.

Testing for phylogenetic component of co-evolution is made by running the above equations while removing individual clades (clusters of species – orthologous sequences with more recent common ancestor). For this provision of phylogenetic tree matching, the species of orthologous sequences is needed. If the correlation coefficient remains statistically significant even after iterative removal of individual clades, the inter-molecular co-evolution component is significant.

Lastly three Chi-squared tests are made to validate which of the pairs are interacting and are involved in protein-protein interactions. O_i is the observed number of AA pairs predicted as co-evolving, $E_i$is the expected number of co-evolving AA pairs and n is the number of possible outcomes.

$$\chi^2 = \sum_{i=1}^{n}\frac{(O_i - E_i)^2}{E_i} \tag{12}$$

This test is repeated 3 times with different E_i which are:

1. E_i is the mean number of co-evolving pairs in a hundred pairs of alignments with same phylogenetic distances.

2. E_i is the the mean number of co-evolving pairs for all pairwise inter-molecular analysis

3. E_i is the the mean number of co-evolving pairs for all pairs of one protein against all other proteins

3. Materials and methods

3.1. Materials

3.1.1. List of proteins

The script used a list of Uniprot protein identifiers as input. These proteins can be a list of any proteins of interest.

Examination of effect of different input parameters on CAPS output was done together with checking of running CAPS in two protein per folder mode. For this the MSA and TOL of the following proteins were used: Toll like receptor 1, 2, 6 (TLR 1,2,6) and CD79a and CD79b.

To test the reliability of the script the MSA and TOLs of three proteins of B cell activation (CD79a, CD79b and Lyn) and two proteins located in mitochondrial matrix (ATP-dependent-Clp-protease-proteolytic-subunit - Clpp, Glutaredoxin-related-protein-5 – Glrx5) were used.

The final input data were MS hits from APEX2 biotin phenol proximity labelling. APEX2 was targeting B cell non/raft plasma membrane regions pre and after B cell activation. B cells were mouse A20 D1.3 cell line, where D1.3 stands for HEL specific IgM BCR expressed on A20 cells. In total there were 538 input protein Uniprot IDs used.

3.1.2. Online databases

Reference sequences of Uniprot ID MS hits were obtained from Universal Protein Resource (UniProt) database (Chen, 2017): http://www.uniprot.org/

Orthologous sequences to Uniprot reference sequences were obtained from OrthoDB database comprehensive catalog of orthologs, i.e. genes inherited by extant species from their last common ancestor (Zdobnov, 2017): http://www.orthodb.org/

Tree of life – phylogenetic tree of vertebrate species used in this study was obtained from TimeTree. Timetree is a database of the tree-of-life and its evolutionary timescale (Hedges, 2015): http://www.timetree.org/.

85 vertebrate species used were chosen as a balance between species with highest number of orthologues (top coverage in the sample of 200 chosen proteins), while maintaining representations of all main vertebrate clades in OrthoDB database. The list of 85 species and their phylogenetic tree of life is depicted in (Figure 6).

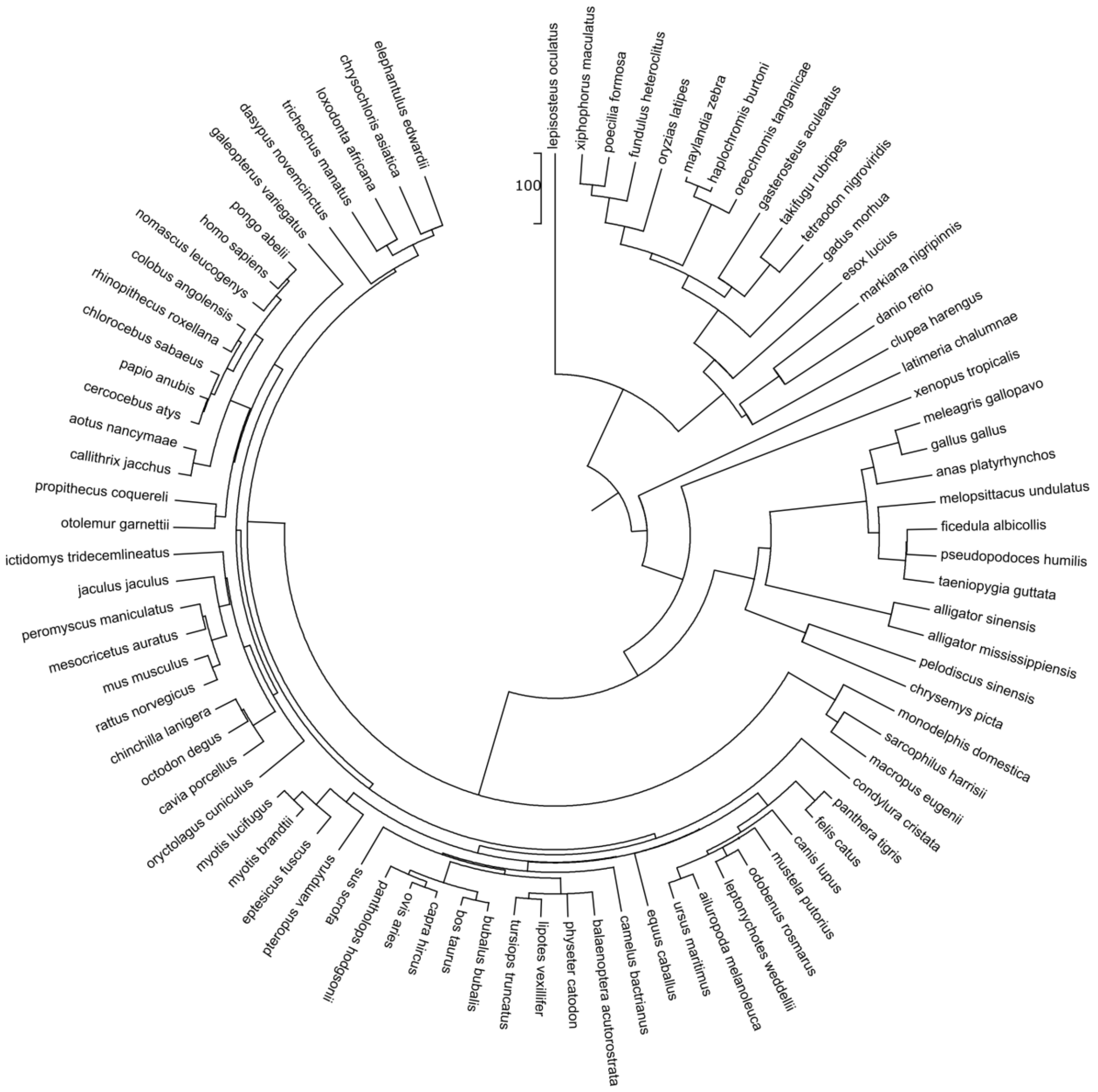


Figure 6. Tree of life of 85 vertebrate species of which orthologues were used in the study obtained from TimeTree.org

3.1.3. Software

The scripts that represented the framework of the pipeline for retrieval of sequences and communication with other programs were written in Python (Python Software Foundation. Python Language Reference, version 2.7. Available at http://www.python.org, van Rossum, 1995) with Python libraries used Biopython (Cock, 2009), lxml (ver. 4.1.0, Behnel, Faassen, Bicking, available at http://lxml.de/)

Local BLAST+ for finding corresponding OrthoDB IDs to Uniprot IDs (Altschul, 1990, Altschul, 1997, Camacho, 2009)

Software for co-evolution analysis was Coevolution analysis using protein sequences, CAPS version 2 (Fares, 2006a, Fares 2006b)

CAPS output was compared to Cytoscape, version 3.6.1 (Shannon, 2003)

Commands for controlling the parallel multicore execution of CAPS were written in BASH Free Software Foundation (2007). Bash (3.2.48) [Unix shell program]. Retrieved from http://ftp.gnu.org/gnu/bash/bash-3.2.48.tar.gz

Computations were performed on local and online Taito supercomputer of CSC – IT Center for Science, Finland.

3.2 Methods

3.2.1. General overview

The scripts aim to

- Read the input of several hundred Uniprot protein identifiers

- Convert Uniprot IDs into OrthoDB protein identifiers

- Retrieve appropriate orthologue sequences from OrthoDB (with high enough identity, length similarity, vertebrate clade, with relatively homogenous spread of representative species, good quality sequences without missing AAs)

- Align these sequences with modern aligning algorithm

- Create individual folders containing aligned sequences for all possible protein-protein coevolving pairs

- Command CAPS algorithm to run number of parallel analyses equal to the number of the cores the computing processor has - the computer being CSC supercomputer.

- The final script could be used for co-evolutionary analysis of any set of hundreds of proteins of interest.

In practice several sub-steps need to be created and actual flow of the script is presented in (Figure 8). Each step is further explained in a separate paragraph. The codes for each section can be found in Appendices.

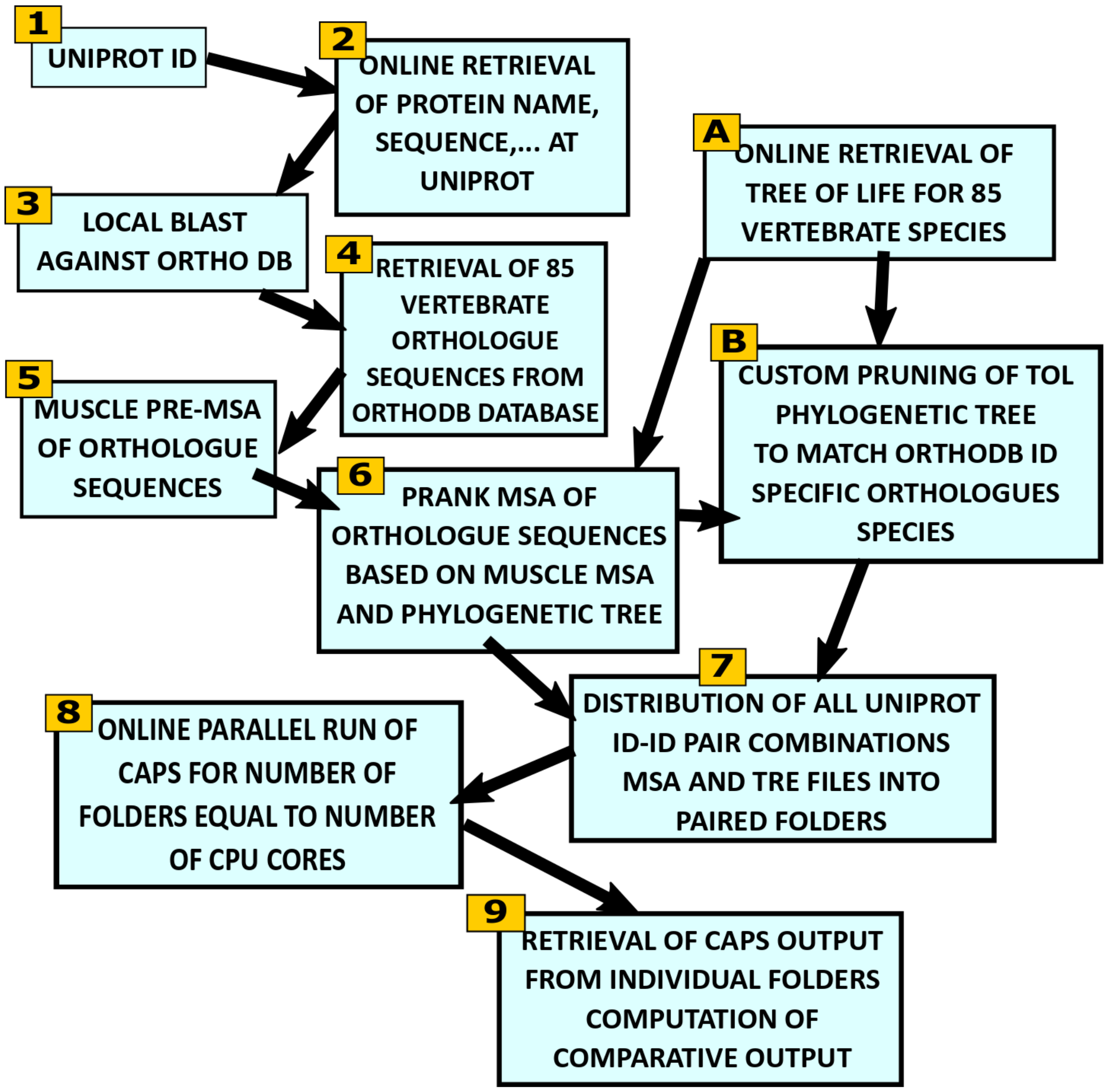

1
UNIPROT ID
2
ONLINE RETRIEVAL OF PROTEIN NAME, SEQUENCE,... AT UNIPROT
3
LOCAL BLAST AGAINST ORTHO DB
4
RETRIEVAL OF 85 VERTEBRATE ORTHOLOGUE SEQUENCES FROM ORTHODB DATABASE
5
MUSCLE PRE-MSA OF ORTHOLOGUE SEQUENCES
6
PRANK MSA OF ORTHOLOGUE SEQUENCES BASED ON MUSCLE MSA AND PHYLOGENETIC TREE
A
ONLINE RETRIEVAL OF TREE OF LIFE FOR 85 VERTEBRATE SPECIES
B
CUSTOM PRUNING OF TOL PHYLOGENETIC TREE TO MATCH ORTHODB ID SPECIFIC ORTHOLOGUES SPECIES
7
DISTRIBUTION OF ALL UNIPROT ID-ID PAIR COMBINATIONS MSA AND TRE FILES INTO PAIRED FOLDERS
8
ONLINE PARALLEL RUN OF CAPS FOR NUMBER OF FOLDERS EQUAL TO NUMBER OF CPU CORES
9
RETRIEVAL OF CAPS OUTPUT FROM INDIVIDUAL FOLDERS COMPUTATION OF COMPARATIVE OUTPUT

Figure 8. Pipeline of the script(s) for high throughput co-evolution analysis with CAPS with marked steps

3.2.2. Step A: TimeTree TOL of 85 vertebrate species

List of 85 vertebrate species with relatively highest orthologue coverage in OrthoDB database while maintaining representation of major vertebrate clades is used for the creation of phylogenetic tree of life. Tree of life shows all the divergences from common ancestor leading to each individual species. Each divergence and time distance from one to other divergence is calculated based on comparative similarities of whole genomes of the species (as in Figure 6). The tree of life is saved as Newick standard format file with extension ".dnd"

3.2.3 Step 1: Uniprot ID

Script input is provided as a list of protein hits in the form of Uniprot identifiers. Each Uniprot identifier is a string readable by Python. Script loops through Steps 1-2-3-4-5-6-B for each Uniprot identifier with a simple for loop.

3.2.4. Step 2: Online retrieval of protein sequence at Uniprot

Representational State Transfer (REST) Application Programming Interface (API) allows access to Uniprot data with structured Uniform Resource Locator (URLs).

The URL is formed and output is retrieved with Python "urllib2" extensible library for opening URLs, for example: "http://www.uniprot.org/uniprot/"uniprotID"/xml". Xml output was parsed with lxml library commands to obtain protein main, alternative, gene names and AA sequence.

3.2.5. Step 3: Local BLAST against OrthoDB

Since the cross-references pointing from one database identifier to identifiers from other databases may not be most accurate or regularly updated, a local BLAST search with Uniprot sequence is made against OrthoDB sequence database with OrthoDB identifiers. Biopython BLAST library Bio.Blast.Applications with NCBI BLAST command line is used.

3.2.6. Step 4: Retrieval of orthologue sequences

OrthoDB database is accessed via REST API. Two structured URL inputs are sent. The URL input contains the OrthoDB identifier of a mouse protein sequence from previous step and a vertebrate clade identifier, so the retrieval is made for all the orthologues to mouse within vertebrates. The URL input contains also the format of the output which is either fasta – cointaining all the fasta sequences of orthologues with OrthoDB id tags or tab – containing the OrthoDB identifiers, species names and other information. Both tab and fasta files are parsed into individual orthologue entries. Because further operation demands tracking of species names each fasta header of each fasta sequence is modified with Python script to contain species name. In OrthoDB single species might contain several homologous hits to the original query because of gene duplication etc. In this case the Python script selects from two homologs the one with the highest similarity to the original mouse protein sequence. In order to achieve this, the script iterates through already retrieved sequences (of the same protein orthologues) with species in the header. In case same species name is found it compares the two sequences. Retrieved filtered orthologue sequences with modified fasta headers are saved in a fasta file to be used in next step. While retrieving input URL only contains the vertebrate clade identifier, and cannot contain the list of species, the output in the final file of this step might contain also vertebrate species that we have not chosen in the 85 vertebrate TOL in step A.

3.2.7. Step 5: MUSCLE pre-MSA

Multiple sequence alignment of the sequences retrieved from OrthoDB is first made with BioPython command line that controls the MUSCLE MSA program outside of Python. It aligns all the sequences in the fasta file, also those not in the 85 species TOL phylogenetic file from step A. The MUSCLE pre-alignment is done in order to speed up the next computationally more demanding PRANK alignment.

3.2.8. Step 6: PRANK MSA

PRANK MSA is made with Biopython library PrankCommandline. PRANK MSA uses phylogenetic information in order to create phylogenetically harmonised distribution of gaps in the alignment in accordance to evolutionary occurring insertions and deletions. The phylogenetic information needed is the TOL phylogenetic file from step A. The other input is the muscle pre-aligned MSA from step 5, the command line needs to contain “pre-aligned=True”. Because TOL might contain vertebrate species that are not in that particular retrieved OrthoDB file and vice versa, the PRANK command line needs to contain command to “prunetree=True”, and “prunedata=True”. These commands were originally not in the PRANK command file “\Bio\Align\Applications_Prank.py” contained in Biopython so they had to be added in the latest version of PRANK (http://wasabiapp.org/software/prank/). The following is the code added to _Prank.py self-class:

“self.parameters = [… _Switch(["-prunetree", "prunetree"]),…”

The resulting PRANK MSA file only contains orthologous sequences of species that are in both MUSCLE OrthoDB retrieved species and TOL phylogenetic tree file with 85 vertebrate species from step A.

3.2.9. Step B: Custom pruning of TOL

CAPS program requires individual phylogenetic tree of life files corresponding to each MSA of particular protein orthologues in order to perform exclusion of phylogenetic component to co-evolution. The phylogenetic tree file must match exactly the number and form of species names in the MSA. Therefore the copy of original tree of life for 85 vertebrate species from step A with a matching name to the MSA file is made. This new phylogenetic file is pruned of any excess species that are not present in MSA output from PRANK in order to match it. Commands from Phylo sub library of BioPython are used for this.

3.2.10. Step 7: Distribution of files

CAPS in its original form uses only one CPU core and is incapable of parallel computing that would greatly speed up the computation. CAPS creates all the combinations of files present in one folder and runs each pair one after another using one core throughout until all the pairs are processed. To circumvent this, the files of interest can also be paired with their partners into individual folders, so CAPS will in each instance compute just the co-evolution between the two files in the folder and as many instances as there are CPU cores are called in parallel.

The orthologues PRANK MSA and corresponding TOL of each input Uniprot identifier coming from the loops of step 2-3-4-5-6-B need to be distributed into separate folders. This is done by the separate Python script that uses “itertools” library. For each pair of proteins separate subfolders are created, one containing two MSA fasta “.fas” files and the other containing phylogenetic “.tre” files.

3.2.11. Step 8: Multicore parallel run of CAPS

BASH command list for the parallel run of CAPS in each folder is created. Archive file of all the folders is made and uploaded to Taito computer of CSC. Taito supercomputer can schedule maximally 700 commands in one instance, therefore the BASH command list needs to be divided accordingly.

Each command calls CAPS from the location of files. Several commands are specified. "-F fasfolder/" specifies the subfolder containing FASTA MSA file. "-inter" is a command for inter-protein co-evolutions (default is intra-protein). "-T trefolder/" specifies subfolder containing phylogenetic tree file. One can add optional commands like "-g 0.6" which will discard any column in FASTA MSA file with >60% gaps. Other optional command is "-H mus_musculus" which will acknowledge the sequence with mus_musculus as the reference sequence when giving the position of co-evolving residue pairs (of course it can be changed to any other FASTA header). "-c" specifies converging command, here CAPS will run as many randomised sampling simulations so that normalised distribution of co-variation correlation coefficients is large enough that probability of alpha one error is insignificant and does not change anymore. Other commands are possible, but are hidden in CAPS.cpp source file code.

Another BASH script is made that will automatically run through each of the CAPS command sublists. After the run, CAPS creates an output into each folder. To extract the output files and separate them from input files in the folders another BASH command list is used for moving the output files in one common folder. Also this "moving" command list needs to be partitioned, however the same BASH script can be used for their automatic sequential execution since the number of commands matches those for running of CAPS. The folder with all joined output can be archived and downloaded from CSC Taito supercomputer.

3.2.12. Step 9: Extracting and computing multiple CAPS output

CAPS creates several types of output files. One file contains the sequences used, and the positions of all the residue pairs that have statistically significant transition variance correlation. Based on this file one can track interacting residue pair in the sequence and make additional calculations based on the biochemical properties of the AAs. Also, one can position the residue pairs in the crystal 3D structure of the proteins, or validate the possible pairs, if it is the case of physical interaction between more commonly known interacting domains. If one would provide ProteinDataBase (PDB) 3D crystal structure files of the proteins CAPS could also automatically mark the found co-evolving residue pairs on those.

The other type of output file contains the abstract of this information, number of interacting residue pairs, sum of all correlation coefficients in co-evolving pairs, cut off threshold,

threshold simulation r, average r, average significant r, file one tree length, file two tree length, gap threshold, bootcutoff, distance coefficient and finally also the conclusive statement whether there is a relative co-evolution between the given pair of proteins in both directions (“YES/NO”).

This final “YES/NO” statement is calculated based on a Chi-squared test that takes into account all the files in the folder. Since in our case there are always two compared proteins, the co-evolution in regard to the average from this same pair is never higher and the co-evolution statement is always negative.

Through python script the former values are extracted from each file of all the pairs of proteins and printed in one file. Chi-squared test is performed to test if the number of observed co-evolving residue pairs is above the expected value. Expected value is calculated based on the average co-evolution coefficient per residue pair for all the protein-protein pairs involving one of the two proteins (meaning also the pairs with other proteins). There are two final statements, one showing the significance of protein 1 -> protein 2 co-evolution in regards to all the co-evolution of protein 1 toward any other calculated protein co-evolution and then another significance of protein 2-> protein 1 in in regards to all the co-evolution of protein 2 toward any other calculated protein co-evolution.

The calculations are then made according to equation 12 in section 2.5.

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$$

Where E_i is the mean number of co-evolving pairs for all pairs of one protein against all other proteins and is calculated as follows:

$$E_i = \left(\frac{\sum_{j=1}^{N} \left(\frac{n_{\rho_{sig}}}{\rho_t} \right)}{N} \right) \cdot \rho_t \qquad (13)$$

And where $n_{\rho_{sig}}$ is the number of pairs of residue positions with statistically significant correlation coefficient and ρ_t is the sum of all correlation coefficients within a protein pair

$$\rho_t = \sum_{AB=1}^{X,Z} \rho_{AB} \qquad (14)$$

Here AB is the current pair of residue positions and X, Z is the last pair of residue positions of which correlation coefficients are calculated.

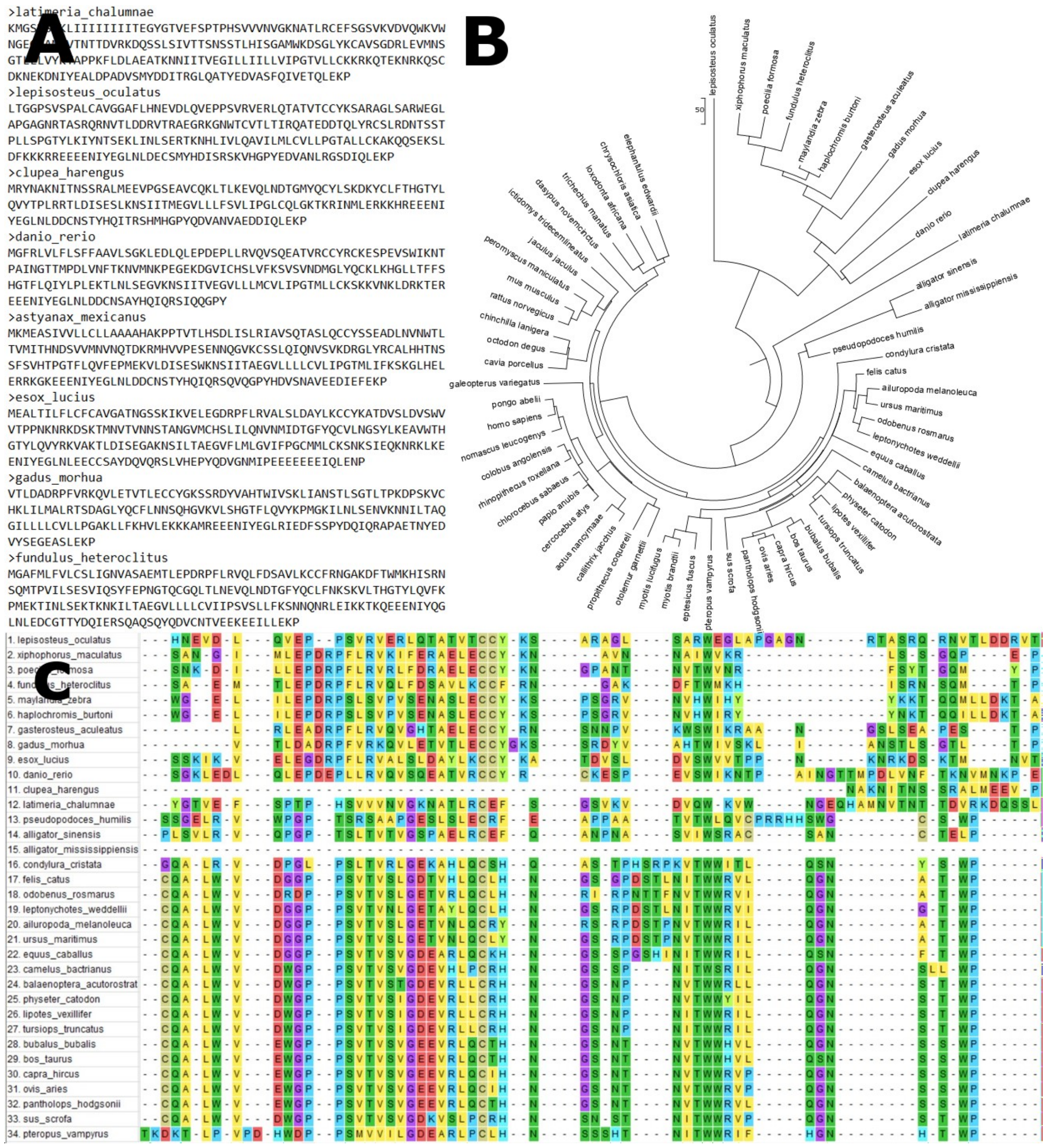
3.2.13. Comparing CAPS results to Cytoscape biochemical interaction data

To assess the correctness of assembled CAPS co-evoltion output, it was compared to Cytoscape biochemical interaction data. The data was obtained from all available online databases automatically determined by Cytoscape for each of the top co-evolving proteins. The data of all databases was merged for each individual protein. Each interaction mapping had one neighbour depth. CD79a or Lyn interaction map versus their top five co-evolving proteins interaction maps were made by merging. The newly established paired merged networks were further merged into one network containing CD79a or Lyn and all their top five co-evolving proteins and their first neighbour interactors. The interactors that did not interact between CD79a (or Lyn) nor any of five top co-evolving proteins (or between these) were erased for better visualisation.

4. Results

4.1 Orthologue retrieval, PRANK MSA and phylogenetic tree pruning

The script is able to retrieve the orthologs from OrthoDB and make PRANK MSA and phylogenetic trees from the provided Uniprot IDs. Figure 9 shows one representative PRANK MSA as visualised in Mega software. Figure 10 shows a corresponding phylogenetic tree



```
>latimeria_chalumnae
KMGS[illegible]KLIIIIIIIIITEGYGTVEFSPTPHSVVVNVGKNATLRCEFSGSVKVDVQWKVW
NGE[illegible]VTNTTDVRKDQSSLSIVTTSNSSTLHISGAMWKDSGLYKCAVSGDRLEVMNS
GT[illegible]LVY[illegible]APPKFLDLAEATKNNIITVEGILLIILLVIPGTVLLCKKRKQTEKNRKQSC
DKNEKDNIYEALDPADVSMYDDITRGLQATYEDVASFQIVETQLEKP
>lepisosteus_oculatus
LTGGPSVSPALCAVGGAFLHNEVDLQVEPPSVRVERLQTATVTCCYKSARAGLSARWEGL
APGAGNRTASRQRNVTLDDRVTRAEGRKGNWTCVTLTIRQATEDDTQLYRCSLRDNTSST
PLLSPGTYLKIYNTSEKLINLSERTKNHLIVLQAVILMLCVLLPGTALLCKAKQQSEKSL
DFKKKRREEEENIYEGLNLDECSMYHDISRSKVHGPYEDVANLRGSDIQLEKP
>clupea_harengus
MRYNAKNITNSSRALMEEVPGSEAVCQKLTLKEVQLNDTGMYQCYLSKDKYCLFTHGTYL
QVYTPLRRTLDISESLKNSIITMEGVLLLFSVLIPGLCQLGKTKRINMLERKKHREEENI
YEGLNLDDCNSTYHQITRSHMHGPYQDVANVAEDDIQLEKP
>danio_rerio
MGFRLVLFLSFFAAVLSGKLEDLQLEPDEPLLRVQVSQEATVRCCYRCKESPEVSWIKNT
PAINGTTMPDLVNFTKNVMNKPEGEKDGVICHSLVFKSVSVNDMGLYQCKLKHGLLTFFS
HGTFLQIYLPLEKTLNLSEGVKNSIITVEGVLLLMCVLIPGTMLLCKSKKVNKLDRKTER
EEENIYEGLNLDDCNSAYHQIQRSIQQGPY
>astyanax_mexicanus
MKMEASIVVLLCLLAAAAHAKPPTVTLHSDLISLRIAVSQTASLQCCYSSEADLNVNWTL
TVMITHNDSVVMNVNQTDKRMHVVPESENNQGVKCSSLQIQNVSVKDRGLYRCALHHTNS
SFSVHTPGTFLQVFEPMEKVLDISESWKNSIITAEGVLLLLCVLIPGTMLIFKSKGLHEL
ERRKGKEEENIYEGLNLDDCNSTYHQIQRSQVQGPYHDVSNAVEEDIEFEKP
>esox_lucius
MEALTILFLCFCAVGATNGSSKIKVELEGDRPFLRVALSLDAYLKCCYKATDVSLDVSWV
VTPPNKNRKDSKTMNVTVNNSTANGVMCHSLILQNVNMIDTGFYQCVLNGSYLKEAVWTH
GTYLQVYRKVAKTLDISEGAKNSILTAEGVFLMLGVIFPGCMMLCKSNKSIEQKNRKLKE
ENIYEGLNLEECCSAYDQVQRSLVHEPYQDVGNMIPEEEEEEEIQLENP
>gadus_morhua
VTLDADRPFVRKQVLETVTLECCYGKSSRDYVAHTWIVSKLIANSTLSGTLTPKDPSKVC
HKLILMALRTSDAGLYQCFLNNSQHGVKVLSHGTFLQVYKPMGKILNLSENVKNNILTAQ
GILLLLCVLLPGAKLLFKHVLEKKKAMREEENIYEGLRIEDFSSPYDQIQRAPAETNYED
VYSEGEASLEKP
>fundulus_heteroclitus
MGAFMLFVLCSLIGNVASAEMTLEPDRPFLRVQLFDSAVLKCCFRNGAKDFTWMKHISRN
SQMTPVILSESVIQSYFEPNGTQCGQLTLNEVQLNDTGFYQCLFNKSKVLTHGTYLQVFK
PMEKTINLSEKTKNKILTAEGVLLLLCVIIPSVSLLFKSNNQNRLEIKKTKQEEENIYQG
LNLEDCGTTYDQIERSQAQSQYQDVCNTVEEKEEILLEKP
```

```
1. lepisosteus_oculatus       ---HNEVD-L---QVEP--PSVRVERLQTATVTCCY-KS-----ARAGL----SARWEGLAPGAGN------RTASRQ-RNVTLDDRVT
2. xiphophorus_maculatus      ---SAN-G-I---MLEPDRPFLRVKIFERAELECCY-KN-------AVN----NAIWVKR--------------LS-S-GQP----E-P
3. poecilia_formosa           ---SNK-D-I---LLEPDRPFLRVRLFDRAELECCY-KN-----GPANT----NVTWVNR--------------FSYT-GQM----Y-P
4. fundulus_heteroclitus      ---SA--E-M---TLEPDRPFLRVQLFDSAVLKCCF-RN-------GAK----DFTWMKH--------------ISRN-SQM----T-P
5. maylandia_zebra            ---WG--E-L---ILEPDRPSLSVPVSENASLECCY-KS-----PSGRV----NVHWIHY--------------YKKT-QQMLLDKT-A
6. haplochromis_burtoni       ---WG--E-L---ILEPDRPSLSVPVSENASLECCY-KS-----PSGRV----NVHWIRY--------------YNKT-QQILLDKT-A
7. gasterosteus_aculeatus     ---------L---RLEADRPFLRVQVGHTAELECCY-RN-----SNNPV----KWSWIKRAA---N------GSLSEA-PES----T-P
8. gadus_morhua               ---------V---TLDADRPFVRKQVLETVTLECCYGKS-----SRDYV----AHTWIVSKL---I------ANSTLS-GTL----T-P
9. esox_lucius                ---SSKIK-V---ELEGDRPFLRVALSLDAYLKCCY-KA-----TDVSL----DVSWVVTPP---N------KNRKDS-KTM----NVT
10. danio_rerio               ---SGKLEDL---QLEPDEPLLRVQVSQEATVRCCY-R------CKESP----EVSWIKNTP---AINGTTMPDLVNF-TKNVMNKP-E
11. clupea_harengus           ----------------------------------------------------------------------NAKNITNS-SRALMEEV-P
12. latimeria_chalumnae       ---YGTVE-F---SPTP--HSVVVNVGKNATLRCEF--S-----GSVKV----DVQW-KVW-----NGEQHAMNVTNT-TDVRKDQSSL
13. pseudopodoces_humilis     --SSGELR-V---WPGP--TSRSAAPGESLSLECRF--E-----APPAA----TVTWLQVCPRRHHSWG--------C-S-WP------
14. alligator_sinensis        --PLSVLR-V---QPGP--TSLTVTVGSPAELRCEF--Q-----ANPNA----SVIWSRAC-----SAN--------C-TELP------
15. alligator_mississippiensis -----------------------------------------------------------------------------------------
16. condylura_cristata        --GQA-LR-V---DPGL--PSLTVRLGEKAHLQCSH--Q-----AS-TPHSRPKVTWWITL-----QSN--------Y-S-WP------
17. felis_catus               --CQA-LW-V---DGGP--PSVTVSLGDTVHLQCLH--N-----GS-GPDSTLNITWWRVL-----QGN--------A-T-WP------
18. odobenus_rosmarus         --CQA-LW-V---DRDP--PSVTVSLGETVRLQCLH--N-----RI-RPNTTFNVTWWRVI-----QGN--------A-T-WP------
19. leptonychotes_weddellii   --CQA-LW-V---DGGP--PSVTVNLGETAYLQCLH--N-----GS-RPDSTLNITWWRVI-----QGN--------G-T-WP------
20. ailuropoda_melanoleuca    --CQA-LW-V---DGGP--PSVTVSLGETVNLQCRY--N-----RS-RPDSTPNVTWWRIL-----QGN--------A-T-WP------
21. ursus_maritimus           --CQA-LW-V---DGGP--PSVTVSLGETVNLQCLY--N-----GS-RPDSTPNVTWWRIL-----QGN--------A-T-WP------
22. equus_caballus            --CQA-LW-V---DGGP--PSVTVSVGDEARLQCKH--N-----GS-SPGSHINITWWRIL-----QSN--------F-T-WP------
23. camelus_bactrianus        --CQA-LW-V---DWGP--PSVTVSVGDEVHLPCRH--N-----GS-SP----NITWSRIL-----QGN--------SLL-WP------
24. balaenoptera_acutorostrat --CQA-LW-V---DWGP--PSVTVSTGDEVRLLCRH--N-----GS-NP----NVTWWRLL-----QGN--------S-T-WP------
25. physeter_catodon          --CQA-LW-V---DWGP--PSVTVSIGDEVRLLCRH--N-----GS-NP----NVTWWYIL-----QGN--------S-T-WP------
26. lipotes_vexillifer        --CQA-LW-V---DWGP--PSVTVSIGDEVRLLCRH--N-----GS-NP----NITWWRIL-----QGN--------S-T-WP------
27. tursiops_truncatus        --CQA-LW-V---DWGP--PSVTVSIGDEVRLLCRH--N-----GS-NP----NITWWRIL-----QGN--------S-T-WP------
28. bubalus_bubalis           --CQA-LW-V---EWGP--PSVTVSVGEEVRLQCTH--N-----GS-NT----NVTWWHVL-----QGN--------S-S-WP------
29. bos_taurus                --CQA-LW-V---EWGP--PSVTVSVGEEVRLQCTH--N-----GS-NT----NVTWWHVL-----QSN--------S-S-WP------
30. capra_hircus              --CQA-LW-V---EWGP--PSVTVSVGEEVRLQCIH--N-----GS-NT----NVTWWRVP-----QGN--------S-S-WP------
31. ovis_aries                --CQA-LW-V---EWGP--PSVTVSVGEEVRLQCIH--N-----GS-NT----NVTWWRVP-----QGN--------S-S-WP------
32. pantholops_hodgsonii      --CQA-LW-V---EWGP--PSVTVSVGEEVRLQCTH--N-----GS-NT----NVTWWRVL-----QGN--------S-S-WP------
33. sus_scrofa                --CQA-LW-V---DWGP--PSVTVSVGDKVSLPCRH--N-----SN-ST----NITWWRVP-----QGN--------S-T-WP------
34. pteropus_vampyrus         TKDKT-LP-VPD-HWDP--PSMVVILGDEARLPCLH--N-----SSSHT----NITWWRIF-----HGN--------H-T-WP------
```

Figure 9. Example of retrieval, TOL and Prank MSA: (A) Retrieved orthologous sequences IgAlpha - CD79a, (B) Tree of Life for 63 vertebrate species both in retrieved CD79a orthologues and in general TOL of 85 chosen vertebrate species (Fig 6) and (C) Corresponding PRANK MSA alignment of retrieved CD79a 63 orthologous vertebrate sequences obtained with the script. Only portion of sequences are shown for better readability.

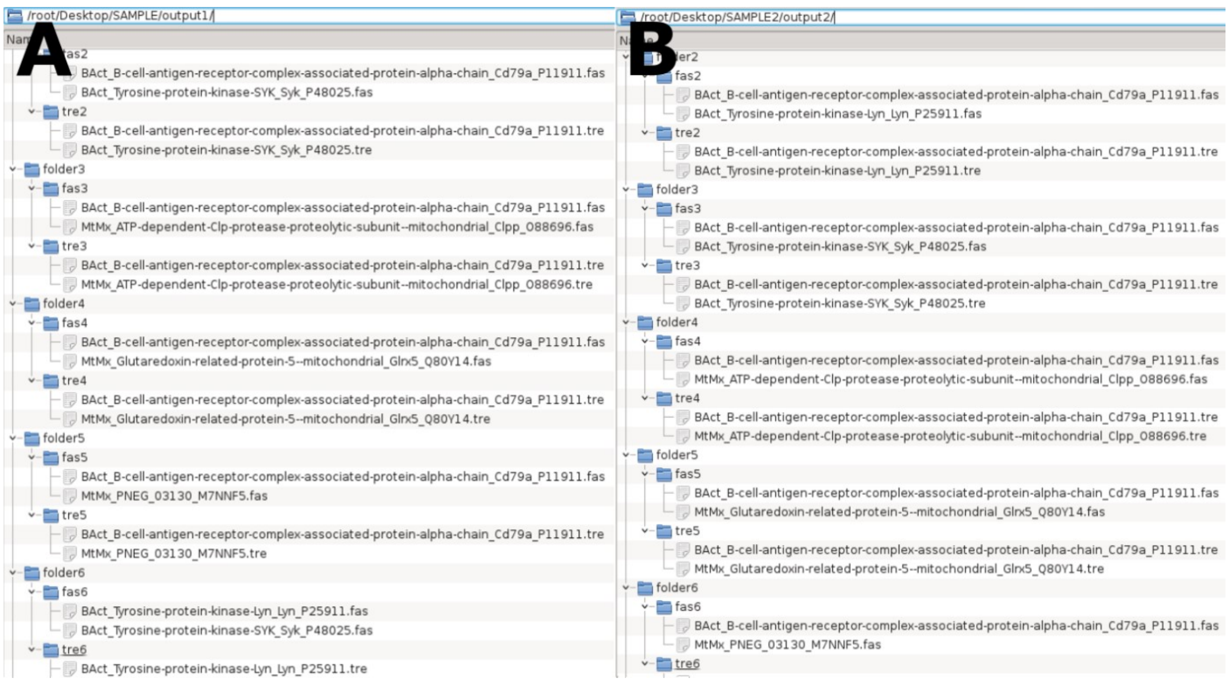
The retrieval of Orthologous sequences, TOL and PRANK MSA creation for individual UniprotID takes on average 1-2 minutes depending on the length of the sequences and number of orthologues found. The retrieval may fail in about 5% of the cases, due to various

4.2 Files distribution into folders

The script was able to distribute the obtained MSA fasta files and corresponding phylogenetic trees paired with all possible interacting protein counterparts into separate folders. This creates

$$\frac{n \cdot (n-1)}{2} \tag{15}$$

of folders for n number of proteins (Figure 10 A). Another alternative is to create pairs of series of proteins versus one particular protein of interest (Figure 10 B). This alternative, more focused approach results in linear increase of resulting folders (n-1). The computational demand is drastically lower compared to the first approach by higher numbers of proteins.



```
/root/Desktop/SAMPLE/output1/
    fas2
        BAct_B-cell-antigen-receptor-complex-associated-protein-alpha-chain_Cd79a_P11911.fas
        BAct_Tyrosine-protein-kinase-SYK_Syk_P48025.fas
    tre2
        BAct_B-cell-antigen-receptor-complex-associated-protein-alpha-chain_Cd79a_P11911.tre
        BAct_Tyrosine-protein-kinase-SYK_Syk_P48025.tre
folder3
    fas3
        BAct_B-cell-antigen-receptor-complex-associated-protein-alpha-chain_Cd79a_P11911.fas
        MtMx_ATP-dependent-Clp-protease-proteolytic-subunit--mitochondrial_Clpp_O88696.fas
    tre3
        BAct_B-cell-antigen-receptor-complex-associated-protein-alpha-chain_Cd79a_P11911.tre
        MtMx_ATP-dependent-Clp-protease-proteolytic-subunit--mitochondrial_Clpp_O88696.tre
folder4
    fas4
        BAct_B-cell-antigen-receptor-complex-associated-protein-alpha-chain_Cd79a_P11911.fas
        MtMx_Glutaredoxin-related-protein-5--mitochondrial_Glrx5_Q80Y14.fas
    tre4
        BAct_B-cell-antigen-receptor-complex-associated-protein-alpha-chain_Cd79a_P11911.tre
        MtMx_Glutaredoxin-related-protein-5--mitochondrial_Glrx5_Q80Y14.tre
folder5
    fas5
        BAct_B-cell-antigen-receptor-complex-associated-protein-alpha-chain_Cd79a_P11911.fas
        MtMx_PNEG_03130_M7NNF5.fas
    tre5
        BAct_B-cell-antigen-receptor-complex-associated-protein-alpha-chain_Cd79a_P11911.tre
        MtMx_PNEG_03130_M7NNF5.tre
folder6
    fas6
        BAct_Tyrosine-protein-kinase-Lyn_Lyn_P25911.fas
        BAct_Tyrosine-protein-kinase-SYK_Syk_P48025.fas
    tre6
        BAct_Tyrosine-protein-kinase-Lyn_Lyn_P25911.tre
```

```
/root/Desktop/SAMPLE2/output2/
er2
    fas2
        BAct_B-cell-antigen-receptor-complex-associated-protein-alpha-chain_Cd79a_P11911.fas
        BAct_Tyrosine-protein-kinase-Lyn_Lyn_P25911.fas
    tre2
        BAct_B-cell-antigen-receptor-complex-associated-protein-alpha-chain_Cd79a_P11911.tre
        BAct_Tyrosine-protein-kinase-Lyn_Lyn_P25911.tre
folder3
    fas3
        BAct_B-cell-antigen-receptor-complex-associated-protein-alpha-chain_Cd79a_P11911.fas
        BAct_Tyrosine-protein-kinase-SYK_Syk_P48025.fas
    tre3
        BAct_B-cell-antigen-receptor-complex-associated-protein-alpha-chain_Cd79a_P11911.tre
        BAct_Tyrosine-protein-kinase-SYK_Syk_P48025.tre
folder4
    fas4
        BAct_B-cell-antigen-receptor-complex-associated-protein-alpha-chain_Cd79a_P11911.fas
        MtMx_ATP-dependent-Clp-protease-proteolytic-subunit--mitochondrial_Clpp_O88696.fas
    tre4
        BAct_B-cell-antigen-receptor-complex-associated-protein-alpha-chain_Cd79a_P11911.tre
        MtMx_ATP-dependent-Clp-protease-proteolytic-subunit--mitochondrial_Clpp_O88696.tre
folder5
    fas5
        BAct_B-cell-antigen-receptor-complex-associated-protein-alpha-chain_Cd79a_P11911.fas
        MtMx_Glutaredoxin-related-protein-5--mitochondrial_Glrx5_Q80Y14.fas
    tre5
        BAct_B-cell-antigen-receptor-complex-associated-protein-alpha-chain_Cd79a_P11911.tre
        MtMx_Glutaredoxin-related-protein-5--mitochondrial_Glrx5_Q80Y14.tre
folder6
    fas6
        BAct_B-cell-antigen-receptor-complex-associated-protein-alpha-chain_Cd79a_P11911.fas
        MtMx_PNEG_03130_M7NNF5.fas
    tre6
```

Figure 10. Files distribution into folders for parallel multicore computing. (A) All possible co-evolving protein pairs from 5 proteins, resulting in 12 pairs. (B) Alternative, all possible pairs to one chosen protein of interest in this case CD79a, resulting in 6 folders.

4.3 Detection of inter-residue inter-protein co-evolution with CAPS

4.3.1 Testing CAPS parameters on TLR 1, 2, 6, CD79a,b proteins

CAPS can be run with set parameters for Alpha (type one error) level, which represents the probability of accepting false positive co-evolution. Default value is set to 0.001. Raising alpha value lowers the stringency of statistical significance of correlation coefficients compared to correlation coefficients of sampled randomly generated sequences. It lowers both the probability of obtaining false positives as well as the probability of finding true positives. When comparing Toll like receptors 1, 2, 6, CD79a and CD79b, the co-evolution pairs of TLR1-6, TLR2-CD79a, TLR2-CD79b were revealed only after increasing alpha parameter from 0.0001 to default value of 0.001 or higher (Figure 11). Unexpectedly the detected co-evolution between TLR2-6 by alpha 0.0001 was lost by higher alpha. Raising the number of random sampling cycles from default 100 to 200 cycles or to converging number of repetitions had no effect, however it doubled the computational time. Gap threshold parameter changes the acceptance tolerance of residue columns with gaps, 0 – full tolerance, 1 – no tolerance (default value is 0.5). Running CAPS with same parameters for pairs of proteins in individual folders for same co-evolving pairs resulted into unexpected additional detected co-evolving pairs of CD79a-CD79b and TLR2-6. This might be due to missing additional Chi-squared test that couldn´t be found from CAPS source code.

											individual folders		
		g	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**1**
		r	**50**	**100**	**200**	**converg**	**100**	**100**	**200**	**100**	**converg**	**100**	**converg**	**converg**
Protein A	Protein B	**a**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.001**	**0.01**	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**
Toll-like-receptor-1_Tlr1_Q9EPQ1	Cd79a_P11911		NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO
Toll-like-receptor-1_Tlr1_Q9EPQ1	Toll-like-receptor-2_Tlr2_Q9QUN7		NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO
Toll-like-receptor-1_Tlr1_Q9EPQ1	Cd79b_P15530		NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO
Toll-like-receptor-1_Tlr1_Q9EPQ1	Toll-like-receptor-6_Tlr6_Q9EPW9		NO	NO	NO	NO	**YES**	**YES**	**YES**	**YES**	**YES**	**YES**	**YES**	**YES**
Cd79a_P11911	Toll-like-receptor-1_Tlr1_Q9EPQ1		NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO
Cd79a_P11911	Toll-like-receptor-2_Tlr2_Q9QUN7		NO	NO	NO	NO	**YES**	**YES**	**YES**	**YES**	**YES**	**YES**	**YES**	**YES**
Cd79a_P11911	Cd79b_P15530		NO	NO	NO	NO	NO	NO	NO	NO	NO	**YES**	**YES**	**YES**
Cd79a_P11911	Toll-like-receptor-6_Tlr6_Q9EPW9		NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO
Toll-like-receptor-2_Tlr2_Q9QUN7	Toll-like-receptor-1_Tlr1_Q9EPQ1		NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO
Toll-like-receptor-2_Tlr2_Q9QUN7	Cd79a_P11911		NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO
Toll-like-receptor-2_Tlr2_Q9QUN7	Cd79b_P15530		NO	NO	NO	NO	**YES**	**YES**	**YES**	**YES**	**YES**	**YES**	**YES**	**YES**
Toll-like-receptor-2_Tlr2_Q9QUN7	Toll-like-receptor-6_Tlr6_Q9EPW9		**YES**	**YES**	**YES**	**YES**	NO	NO	NO	NO	NO	NO	NO	NO
Cd79b_P15530	Toll-like-receptor-1_Tlr1_Q9EPQ1		NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO
Cd79b_P15530	Cd79a_P11911		NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO
Cd79b_P15530	Toll-like-receptor-2_Tlr2_Q9QUN7		NO	NO	NO	NO	**YES**	**YES**	**YES**	**YES**	**YES**	**YES**	**YES**	**YES**
Cd79b_P15530	Toll-like-receptor-6_Tlr6_Q9EPW9		NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO
Toll-like-receptor-6_Tlr6_Q9EPW9	Toll-like-receptor-1_Tlr1_Q9EPQ1		NO	NO	NO	NO	**YES**	**YES**	**YES**	**YES**	**YES**	**YES**	**YES**	**YES**
Toll-like-receptor-6_Tlr6_Q9EPW9	Cd79a_P11911		NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO
Toll-like-receptor-6_Tlr6_Q9EPW9	Toll-like-receptor-2_Tlr2_Q9QUN7		**YES**	**YES**	**YES**	**YES**	NO	NO	NO	NO	NO	**YES**	**YES**	**YES**
Toll-like-receptor-6_Tlr6_Q9EPW9	Cd79b_P15530		NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO

Figure 11. CAPS analysis of inter-protein co-evolution of sample proteins with different alpha (type I error, a) levels, number of random cycles (r), gap threshold level (g), where 0 – full tolerance, 1 – no tolerance. Found co-evolution marked by grey shade.

4.3.2 Testing CAPS on unlikely co-evolving proteins

To test the performance of CAPS on data with clearly expected or unexpected co-evolution, CAPS was run with 3 proteins involved in B cell activation (CD79a, CD79b and Lyn) and 2 proteins located in mitochondrial matrix (ATP-dependent-Clp-protease-proteolytic-subunit - Clpp, Glutaredoxin-related-protein-5 – Glrx5). Inter-protein co-evolution was found for

CD79a-Clpp, CD79a-Glrx5, CD79b-Clpp, Lyn-CD79a, Lyn-Glrx5, Clpp-CD79b (this co-evolution is found in the context of Clpp, compared to all its other pairs), Glrx5-Clpp. This totals in 5 inter-protein co-evolutions between mitochondrial matrix proteins and B cell activatory proteins located by cell plasma membrane compared to only one of co-evolutions within each of these compartments. Raising the gap parameter in range of 0.5, 0.8, 0.9 and 1 did not affect the outcome (Figure 12). The CAPS was run in individual-pairs-folder mode.

Protein A			Protein B								gap threshold			
cell function	name	gene	cell function	gene	n. pairs	total corr.coef.	n.pair/t.cor.c *	aver.(np/tcc) *	av.x tcorc	chi	0.5	0.8	0.9	1
BActpath	B-cell-antigen-receptor-complex-associated-protein-alpha-chain	Cd79a	**BActpath**	Cd79b	9	241296	3.7	4.1	9.9	0.07	NO	NO	NO	NO
BActpath	B-cell-antigen-receptor-complex-associated-protein-alpha-chain	Cd79a	**BActpath**	Lyn	7	439177	1.6	4.1	17.9	6.67	NO	NO	NO	NO
BActpath	B-cell-antigen-receptor-complex-associated-protein-alpha-chain	Cd79a	MtMx	Clpp	10	171375	5.8	4.1	7.0	1.29	YES	YES	YES	YES
BActpath	B-cell-antigen-receptor-complex-associated-protein-alpha-chain	Cd79a	MtMx	Glrx5	8	154466	5.2	4.1	6.3	0.45	YES	YES	YES	YES
BActpath	B-cell-antigen-receptor-complex-associated-protein-beta-chain	Cd79b	**BActpath**	Cd79a	9	241296	3.7	12.7	30.7	15.32	NO	NO	NO	NO
BActpath	B-cell-antigen-receptor-complex-associated-protein-beta-chain	Cd79b	**BActpath**	Lyn	0	507408	0.0	12.7	64.5	64.52	NO	NO	NO	NO
BActpath	B-cell-antigen-receptor-complex-associated-protein-beta-chain	Cd79b	MtMx	Clpp	90	198000	45.5	12.7	25.2	166.88	YES	YES	YES	YES
BActpath	B-cell-antigen-receptor-complex-associated-protein-beta-chain	Cd79b	MtMx	Glrx5	3	178464	1.7	12.7	22.7	17.09	NO	NO	NO	NO
BActpath	Tyrosine-protein-kinase-Lyn	Lyn	**BActpath**	Cd79a	7	439177	1.6	0.8	3.4	3.68	YES	YES	YES	YES
BActpath	Tyrosine-protein-kinase-Lyn	Lyn	**BActpath**	Cd79b	0	507408	0.0	0.8	4.0	3.97	NO	NO	NO	NO
BActpath	Tyrosine-protein-kinase-Lyn	Lyn	MtMx	Clpp	0	360375	0.0	0.8	2.8	2.82	NO	NO	NO	NO
BActpath	Tyrosine-protein-kinase-Lyn	Lyn	MtMx	Glrx5	5	324818	1.5	0.8	2.5	2.37	YES	YES	YES	YES
MtMx	ATP-dependent-Clp-protease-proteolytic-subunit--mitochondrial	Clpp	**BActpath**	Cd79a	10	171375	5.8	17.4	29.7	13.11	NO	NO	NO	NO
MtMx	ATP-dependent-Clp-protease-proteolytic-subunit--mitochondrial	Clpp	**BActpath**	Cd79b	90	198000	45.5	17.4	34.4	90.04	YES	YES	YES	YES
MtMx	ATP-dependent-Clp-protease-proteolytic-subunit--mitochondrial	Clpp	**BActpath**	Lyn	0	360375	0.0	17.4	62.6	62.56	NO	NO	NO	NO
MtMx	ATP-dependent-Clp-protease-proteolytic-subunit--mitochondrial	Clpp	MtMx	Glrx5	23	126750	18.1	17.4	22.0	0.05	NO	NO	NO	NO
MtMx	Glutaredoxin-related-protein-5--mitochondrial	Glrx5	**BActpath**	Cd79a	8	154466	5.2	6.6	10.3	0.49	NO	NO	NO	NO
MtMx	Glutaredoxin-related-protein-5--mitochondrial	Glrx5	**BActpath**	Cd79b	3	178464	1.7	6.6	11.8	6.60	NO	NO	NO	NO
MtMx	Glutaredoxin-related-protein-5--mitochondrial	Glrx5	**BActpath**	Lyn	5	324818	1.5	6.6	21.6	12.72	NO	NO	NO	NO
MtMx	Glutaredoxin-related-protein-5--mitochondrial	Glrx5	MtMx	Clpp	23	126750	18.1	6.6	8.4	25.30	YES	YES	YES	YES

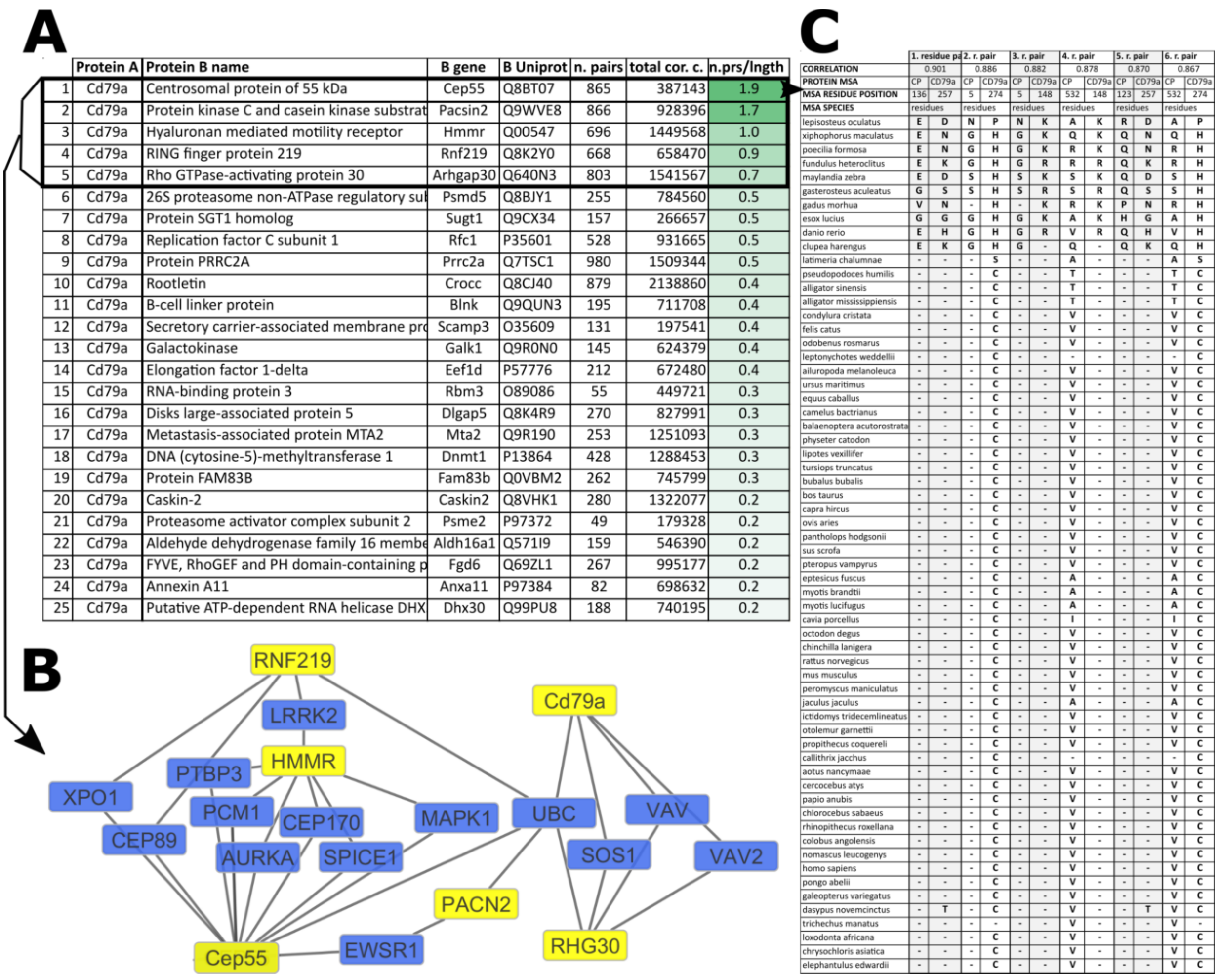
Figure 12. CAPS analysis of inter-protein co-evolution of B cell activation and mitochondrial matrix proteins with parameters used for calculations of significant inter-protein co-evolution from individual pair folders. "n.pairs" – number of co-evolving residue pairs, "total corr. coef" – sum of all correlation coefficients for whole protein pair, "n.pair/t.cor.c" – number of co-evolving residue pairs divided by sum of all correlation coefficients for whole protein pair, "aver.(np/tcc)" – average "n.pair/t.cor.c" within all pairs to the protein A, "av.xtcorc" is the E_i mean number of co-evolving pairs (as in equation 13) and "chi" is the chi squared value (as in equation 12). Found co-evolution marked darker "YES".

*** x10^5**

4.3.3 CAPS on proximity labelling MS hits

Finally, the CAPS was run in parallel in individual paired folder mode as described in the methods (3.2.11. Step 8: Multicore parallel run of CAPS). 538 proteins of non-raft targeted APEX2 proximity labelling MS hits were paired against two known B cell activation proteins CD79a (Figure 13) and Lyn (Figure 14). The co-evolution significance was determined as previously described (equation 12). However, since this categorisation only gives "YES/NO" statement further ranking was used by number of found inter-residue pairs between two proteins, divided by the length of the MSA of the partner protein to CD79a (Figure 13 A) or Lyn (Figure 14 A). The resulting top ranking co-evolving proteins were analysed also in

Cytoscape program. Cytoscape acquires biochemically determined interaction networks from public databases. Of each top five CD79a or Lyn co-evolution partners first neighbour interaction networks were merged to find the overlapping (closest known) sharing interaction partners to CD79a or Lyn. Top four found co-evolving partners to CD79a only interacted with CD79a via UBC - Polyubiquitin-C, which promotes protein recycling, whereas the fifth top co-evolving protein Arhgap30 had additional three shared interacting first neighbour partners to CD79a protein (Figure 13 B). Top co-evolving proteins to Lyn all shared at least five functional interaction first partners with Lyn or each other. The fourth top co-evolving protein to Lyn – Was had direct interaction with Lyn (Figure 14 B). The highest ranked co-evolving protein for CD79a or Lyn was further analysed. Six co-evolving residue pairs between Cep55 and CD79 with highest correlation coefficient from MSA were aligned to each other (as columns in Figure 3 B or C). With the top five of the co-evolving residue pairs the gaps were also included in CAPS correlation coefficient calculations (Figure 13 C). Similarly for the top six co-evolving residue pairs between Lyn and Hmmr alignment, it was observed that gaps contributed to the calculations of CAPS correlation coefficient calculations (Figure 14 C). The analysis was done with gap threshold set at 1, random cycles set to converge and default alpha value of 0.001.

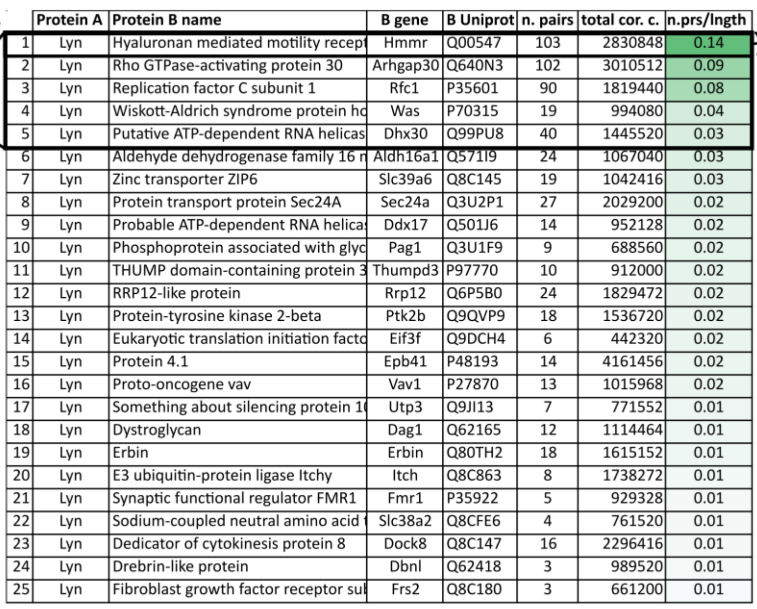


	Protein A	Protein B name	B gene	B Uniprot	n. pairs	total cor. c.	n.prs/lngth
1	Cd79a	Centrosomal protein of 55 kDa	Cep55	Q8BT07	865	387143	1.9
2	Cd79a	Protein kinase C and casein kinase substrat	Pacsin2	Q9WVE8	866	928396	1.7
3	Cd79a	Hyaluronan mediated motility receptor	Hmmr	Q00547	696	1449568	1.0
4	Cd79a	RING finger protein 219	Rnf219	Q8K2Y0	668	658470	0.9
5	Cd79a	Rho GTPase-activating protein 30	Arhgap30	Q640N3	803	1541567	0.7
6	Cd79a	26S proteasome non-ATPase regulatory sul	Psmd5	Q8BJY1	255	784560	0.5
7	Cd79a	Protein SGT1 homolog	Sugt1	Q9CX34	157	266657	0.5
8	Cd79a	Replication factor C subunit 1	Rfc1	P35601	528	931665	0.5
9	Cd79a	Protein PRRC2A	Prrc2a	Q7TSC1	980	1509344	0.5
10	Cd79a	Rootletin	Crocc	Q8CJ40	879	2138860	0.4
11	Cd79a	B-cell linker protein	Blnk	Q9QUN3	195	711708	0.4
12	Cd79a	Secretory carrier-associated membrane pro	Scamp3	O35609	131	197541	0.4
13	Cd79a	Galactokinase	Galk1	Q9R0N0	145	624379	0.4
14	Cd79a	Elongation factor 1-delta	Eef1d	P57776	212	672480	0.4
15	Cd79a	RNA-binding protein 3	Rbm3	O89086	55	449721	0.3
16	Cd79a	Disks large-associated protein 5	Dlgap5	Q8K4R9	270	827991	0.3
17	Cd79a	Metastasis-associated protein MTA2	Mta2	Q9R190	253	1251093	0.3
18	Cd79a	DNA (cytosine-5)-methyltransferase 1	Dnmt1	P13864	428	1288453	0.3
19	Cd79a	Protein FAM83B	Fam83b	Q0VBM2	262	745799	0.3
20	Cd79a	Caskin-2	Caskin2	Q8VHK1	280	1322077	0.2
21	Cd79a	Proteasome activator complex subunit 2	Psme2	P97372	49	179328	0.2
22	Cd79a	Aldehyde dehydrogenase family 16 membe	Aldh16a1	Q571I9	159	546390	0.2
23	Cd79a	FYVE, RhoGEF and PH domain-containing p	Fgd6	Q69ZL1	267	995177	0.2
24	Cd79a	Annexin A11	Anxa11	P97384	82	698632	0.2
25	Cd79a	Putative ATP-dependent RNA helicase DHX	Dhx30	Q99PU8	188	740195	0.2

	1. residue pa		2. r. pair		3. r. pair		4. r. pair		5. r. pair		6. r. pair	
CORRELATION	0.901		0.886		0.882		0.878		0.870		0.867	
PROTEIN MSA	CP	CD79a	CP	CD79a	CP	CD79a	CP	CD79a	CP	CD79a	CP	CD79a
MSA RESIDUE POSITION	136	257	5	274	5	148	532	148	123	257	532	274
MSA SPECIES	residues		residues		residues		residues		residues		residues	
lepisosteus oculatus	E	D	N	P	N	K	A	K	R	D	A	P
xiphophorus maculatus	E	N	G	H	G	K	Q	K	Q	N	Q	H
poecilia formosa	E	N	G	H	G	K	R	K	Q	N	R	H
fundulus heteroclitus	E	K	G	H	G	R	R	R	Q	K	R	H
maylandia zebra	E	D	S	H	S	K	S	K	Q	D	S	H
gasterosteus aculeatus	G	S	S	H	S	R	S	R	Q	S	S	H
gadus morhua	V	N	-	H	-	K	R	K	P	N	R	H
esox lucius	G	G	G	H	G	K	A	K	H	G	A	H
danio rerio	E	H	G	H	G	R	V	R	Q	H	V	H
clupea harengus	E	K	G	H	G	-	Q	-	Q	K	Q	H
latimeria chalumnae	-	-	-	S	-	-	A	-	-	-	A	S
pseudopodoces humilis	-	-	-	C	-	-	T	-	-	-	T	C
alligator sinensis	-	-	-	C	-	-	T	-	-	-	T	C
alligator mississippiensis	-	-	-	C	-	-	T	-	-	-	T	C
condylura cristata	-	-	-	C	-	-	V	-	-	-	V	C
felis catus	-	-	-	C	-	-	V	-	-	-	V	C
odobenus rosmarus	-	-	-	C	-	-	V	-	-	-	V	C
leptonychotes weddellii	-	-	-	C	-	-	-	-	-	-	-	C
ailuropoda melanoleuca	-	-	-	C	-	-	V	-	-	-	V	C
ursus maritimus	-	-	-	C	-	-	V	-	-	-	V	C
equus caballus	-	-	-	C	-	-	V	-	-	-	V	C
camelus bactrianus	-	-	-	C	-	-	V	-	-	-	V	C
balaenoptera acutorostrata	-	-	-	C	-	-	V	-	-	-	V	C
physeter catodon	-	-	-	C	-	-	V	-	-	-	V	C
lipotes vexillifer	-	-	-	C	-	-	V	-	-	-	V	C
tursiops truncatus	-	-	-	C	-	-	V	-	-	-	V	C
bubalus bubalis	-	-	-	C	-	-	V	-	-	-	V	C
bos taurus	-	-	-	C	-	-	V	-	-	-	V	C
capra hircus	-	-	-	C	-	-	V	-	-	-	V	C
ovis aries	-	-	-	C	-	-	V	-	-	-	V	C
pantholops hodgsonii	-	-	-	C	-	-	V	-	-	-	V	C
sus scrofa	-	-	-	C	-	-	V	-	-	-	V	C
pteropus vampyrus	-	-	-	C	-	-	V	-	-	-	V	C
eptesicus fuscus	-	-	-	C	-	-	A	-	-	-	A	C
myotis brandtii	-	-	-	C	-	-	A	-	-	-	A	C
myotis lucifugus	-	-	-	C	-	-	A	-	-	-	A	C
cavia porcellus	-	-	-	C	-	-	I	-	-	-	I	C
octodon degus	-	-	-	C	-	-	V	-	-	-	V	C
chinchilla lanigera	-	-	-	C	-	-	V	-	-	-	V	C
rattus norvegicus	-	-	-	C	-	-	V	-	-	-	V	C
mus musculus	-	-	-	C	-	-	V	-	-	-	V	C
peromyscus maniculatus	-	-	-	C	-	-	V	-	-	-	V	C
jaculus jaculus	-	-	-	C	-	-	A	-	-	-	A	C
ictidomys tridecemlineatus	-	-	-	C	-	-	V	-	-	-	V	C
otolemur garnettii	-	-	-	C	-	-	V	-	-	-	V	C
propithecus coquereli	-	-	-	C	-	-	V	-	-	-	V	C
callithrix jacchus	-	-	-	C	-	-	-	-	-	-	-	C
aotus nancymaae	-	-	-	C	-	-	V	-	-	-	V	C
cercocebus atys	-	-	-	C	-	-	V	-	-	-	V	C
papio anubis	-	-	-	C	-	-	V	-	-	-	V	C
chlorocebus sabaeus	-	-	-	C	-	-	V	-	-	-	V	C
rhinopithecus roxellana	-	-	-	C	-	-	V	-	-	-	V	C
colobus angolensis	-	-	-	C	-	-	V	-	-	-	V	C
nomascus leucogenys	-	-	-	C	-	-	V	-	-	-	V	C
homo sapiens	-	-	-	C	-	-	V	-	-	-	V	C
pongo abelii	-	-	-	C	-	-	V	-	-	-	V	C
galeopterus variegatus	-	-	-	C	-	-	V	-	-	-	V	C
dasypus novemcinctus	-	T	-	C	-	-	V	-	-	T	V	C
trichechus manatus	-	-	-	-	-	-	V	-	-	-	V	-
loxodonta africana	-	-	-	C	-	-	V	-	-	-	V	C
chrysochloris asiatica	-	-	-	C	-	-	V	-	-	-	V	C
elephantulus edwardii	-	-	-	C	-	-	V	-	-	-	V	C

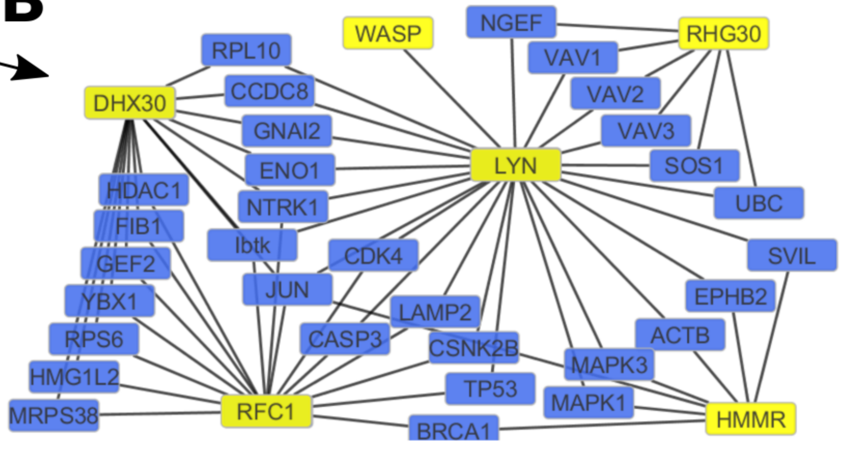
Figure 13. CAPS analysis of inter-protein co-evolution of CD79a versus 538 proteins of non-raft targeted APEX2 proximity labelling MS hits. (A) Ordered by highest score in "n.prs/lngth" - number of found co-evolving residue pairs between two proteins divided by the length of non-CD79a partner protein. (B) For comparison; top five ranked CD79a co-evolving proteins (yellow) in single neighbour interaction distance as biochemically determined in Cytoscape public databases. (C) Top 6 highest correlation coefficient inter-residue co-evolving pairs between Cep55 and CD79a, the residue columns (as in Figure 3 B,C) with co-evolving amino-acid residues.

A

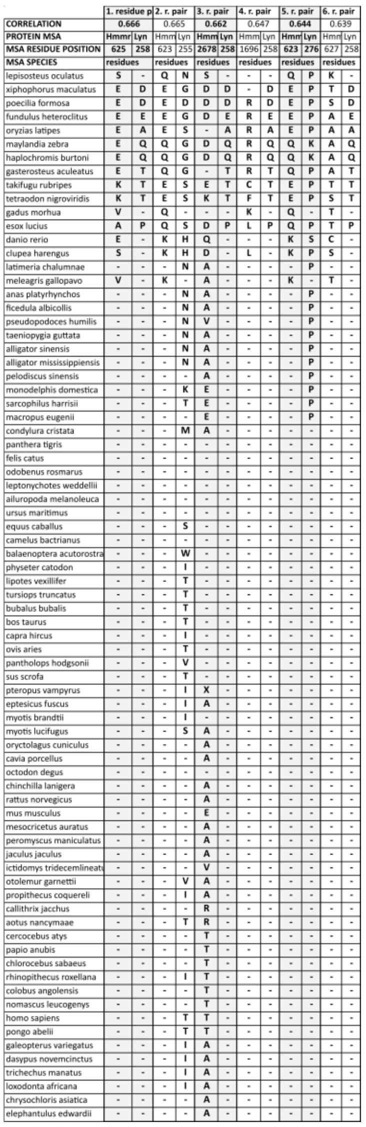


	Protein A	Protein B name	B gene	B Uniprot	n. pairs	total cor. c.	n.prs/lngth
1	Lyn	Hyaluronan mediated motility recept	Hmmr	Q00547	103	2830848	0.14
2	Lyn	Rho GTPase-activating protein 30	Arhgap30	Q640N3	102	3010512	0.09
3	Lyn	Replication factor C subunit 1	Rfc1	P35601	90	1819440	0.08
4	Lyn	Wiskott-Aldrich syndrome protein ho	Was	P70315	19	994080	0.04
5	Lyn	Putative ATP-dependent RNA helicas	Dhx30	Q99PU8	40	1445520	0.03
6	Lyn	Aldehyde dehydrogenase family 16 m	Aldh16a1	Q571I9	24	1067040	0.03
7	Lyn	Zinc transporter ZIP6	Slc39a6	Q8C145	19	1042416	0.03
8	Lyn	Protein transport protein Sec24A	Sec24a	Q3U2P1	27	2029200	0.02
9	Lyn	Probable ATP-dependent RNA helica	Ddx17	Q501J6	14	952128	0.02
10	Lyn	Phosphoprotein associated with glyc	Pag1	Q3U1F9	9	688560	0.02
11	Lyn	THUMP domain-containing protein 3	Thumpd3	P97770	10	912000	0.02
12	Lyn	RRP12-like protein	Rrp12	Q6P5B0	24	1829472	0.02
13	Lyn	Protein-tyrosine kinase 2-beta	Ptk2b	Q9QVP9	18	1536720	0.02
14	Lyn	Eukaryotic translation initiation facto	Eif3f	Q9DCH4	6	442320	0.02
15	Lyn	Protein 4.1	Epb41	P48193	14	4161456	0.02
16	Lyn	Proto-oncogene vav	Vav1	P27870	13	1015968	0.02
17	Lyn	Something about silencing protein 10	Utp3	Q9JI13	7	771552	0.01
18	Lyn	Dystroglycan	Dag1	Q62165	12	1114464	0.01
19	Lyn	Erbin	Erbin	Q80TH2	18	1615152	0.01
20	Lyn	E3 ubiquitin-protein ligase Itchy	Itch	Q8C863	8	1738272	0.01
21	Lyn	Synaptic functional regulator FMR1	Fmr1	P35922	5	929328	0.01
22	Lyn	Sodium-coupled neutral amino acid t	Slc38a2	Q8CFE6	4	761520	0.01
23	Lyn	Dedicator of cytokinesis protein 8	Dock8	Q8C147	16	2296416	0.01
24	Lyn	Drebrin-like protein	Dbnl	Q62418	3	989520	0.01
25	Lyn	Fibroblast growth factor receptor sul	Frs2	Q8C180	3	661200	0.01

B



C



	1. residue p		2. r. pair		3. r. pair		4. r. pair		5. r. pair		6. r. pair	
CORRELATION	0.666		0.665		0.662		0.647		0.644		0.639	
PROTEIN MSA	Hmmr	Lyn	Hmm	Lyn	Hmm	Lyn	Hmm	Lyn	Hmm	Lyn	Hmm	Lyn
MSA RESIDUE POSITION	625	258	623	255	2678	258	1696	258	623	276	627	258
MSA SPECIES	residues		residues		residues		residues		residues		residues	
lepisosteus oculatus	S	-	Q	N	S	-	-	-	Q	P	K	-
xiphophorus maculatus	E	D	E	G	D	D	-	D	E	P	T	D
poecilia formosa	E	D	E	D	D	D	R	D	E	P	S	D
fundulus heteroclitus	E	E	E	G	D	E	R	E	E	P	A	E
oryzias latipes	E	A	E	S	-	A	R	A	E	P	A	A
maylandia zebra	E	Q	Q	G	D	Q	R	Q	Q	K	A	Q
haplochromis burtoni	E	Q	Q	G	D	Q	R	Q	Q	K	A	Q
gasterosteus aculeatus	E	T	Q	G	-	T	R	T	Q	P	A	T
takifugu rubripes	K	T	E	S	E	T	C	T	E	P	T	T
tetraodon nigroviridis	K	T	E	S	K	T	F	T	E	P	S	T
gadus morhua	V	-	Q	-	-	-	K	-	Q	-	T	-
esox lucius	A	P	Q	S	D	P	L	P	Q	P	T	P
danio rerio	E	-	K	H	Q	-	-	-	K	S	C	-
clupea harengus	S	-	K	H	D	-	L	-	K	P	S	-
latimeria chalumnae	-	-	-	N	A	-	-	-	-	P	-	-
meleagris gallopavo	V	-	K	-	A	-	-	-	K	-	T	-
anas platyrhynchos	-	-	-	N	A	-	-	-	-	P	-	-
ficedula albicollis	-	-	-	N	A	-	-	-	-	P	-	-
pseudopodoces humilis	-	-	-	N	V	-	-	-	-	P	-	-
taeniopygia guttata	-	-	-	N	A	-	-	-	-	P	-	-
alligator sinensis	-	-	-	N	A	-	-	-	-	P	-	-
alligator mississippiensis	-	-	-	N	A	-	-	-	-	P	-	-
pelodiscus sinensis	-	-	-	-	A	-	-	-	-	P	-	-
monodelphis domestica	-	-	-	K	E	-	-	-	-	P	-	-
sarcophilus harrisii	-	-	-	T	E	-	-	-	-	P	-	-
macropus eugenii	-	-	-	-	E	-	-	-	-	P	-	-
condylura cristata	-	-	-	M	A	-	-	-	-	-	-	-
panthera tigris	-	-	-	-	-	-	-	-	-	-	-	-
felis catus	-	-	-	-	-	-	-	-	-	-	-	-
odobenus rosmarus	-	-	-	-	-	-	-	-	-	-	-	-
leptonychotes weddellii	-	-	-	-	-	-	-	-	-	-	-	-
ailuropoda melanoleuca	-	-	-	-	-	-	-	-	-	-	-	-
ursus maritimus	-	-	-	-	-	-	-	-	-	-	-	-
equus caballus	-	-	-	S	-	-	-	-	-	-	-	-
camelus bactrianus	-	-	-	-	-	-	-	-	-	-	-	-
balaenoptera acutorostra	-	-	-	W	-	-	-	-	-	-	-	-
physeter catodon	-	-	-	I	-	-	-	-	-	-	-	-
lipotes vexillifer	-	-	-	T	-	-	-	-	-	-	-	-
tursiops truncatus	-	-	-	T	-	-	-	-	-	-	-	-
bubalus bubalis	-	-	-	T	-	-	-	-	-	-	-	-
bos taurus	-	-	-	T	-	-	-	-	-	-	-	-
capra hircus	-	-	-	I	-	-	-	-	-	-	-	-
ovis aries	-	-	-	T	-	-	-	-	-	-	-	-
pantholops hodgsonii	-	-	-	V	-	-	-	-	-	-	-	-
sus scrofa	-	-	-	T	-	-	-	-	-	-	-	-
pteropus vampyrus	-	-	-	I	X	-	-	-	-	-	-	-
eptesicus fuscus	-	-	-	I	A	-	-	-	-	-	-	-
myotis brandtii	-	-	-	I	-	-	-	-	-	-	-	-
myotis lucifugus	-	-	-	S	A	-	-	-	-	-	-	-
oryctolagus cuniculus	-	-	-	-	A	-	-	-	-	-	-	-
cavia porcellus	-	-	-	-	A	-	-	-	-	-	-	-
octodon degus	-	-	-	-	-	-	-	-	-	-	-	-
chinchilla lanigera	-	-	-	-	A	-	-	-	-	-	-	-
rattus norvegicus	-	-	-	-	A	-	-	-	-	-	-	-
mus musculus	-	-	-	-	E	-	-	-	-	-	-	-
mesocricetus auratus	-	-	-	-	A	-	-	-	-	-	-	-
peromyscus maniculatus	-	-	-	-	A	-	-	-	-	-	-	-
jaculus jaculus	-	-	-	-	A	-	-	-	-	-	-	-
ictidomys tridecemlineatu	-	-	-	-	V	-	-	-	-	-	-	-
otolemur garnettii	-	-	-	V	A	-	-	-	-	-	-	-
propithecus coquereli	-	-	-	I	A	-	-	-	-	-	-	-
callithrix jacchus	-	-	-	-	R	-	-	-	-	-	-	-
aotus nancymaae	-	-	-	T	R	-	-	-	-	-	-	-
cercocebus atys	-	-	-	-	T	-	-	-	-	-	-	-
papio anubis	-	-	-	-	T	-	-	-	-	-	-	-
chlorocebus sabaeus	-	-	-	-	T	-	-	-	-	-	-	-
rhinopithecus roxellana	-	-	-	I	T	-	-	-	-	-	-	-
colobus angolensis	-	-	-	-	T	-	-	-	-	-	-	-
nomascus leucogenys	-	-	-	-	T	-	-	-	-	-	-	-
homo sapiens	-	-	-	T	T	-	-	-	-	-	-	-
pongo abelii	-	-	-	T	T	-	-	-	-	-	-	-
galeopterus variegatus	-	-	-	I	A	-	-	-	-	-	-	-
dasypus novemcinctus	-	-	-	I	A	-	-	-	-	-	-	-
trichechus manatus	-	-	-	I	A	-	-	-	-	-	-	-
loxodonta africana	-	-	-	I	A	-	-	-	-	-	-	-
chrysochloris asiatica	-	-	-	-	A	-	-	-	-	-	-	-
elephantulus edwardii	-	-	-	-	A	-	-	-	-	-	-	-

Figure 14. CAPS analysis of inter-protein co-evolution of Lyn versus 538 proteins of non-raft targeted APEX2 proximity labelling MS hits. (A) Ordered by highest score in "n.prs/lngth" - number of found co-evolving residue pairs between two proteins divided by the length of non-Lyn partner protein. (B) For comparison; top five ranked Lyn co-evolving proteins (yellow) in single neighbour interaction distance as biochemicaly determined in Cytoscape public databases. (C) Top 6 highest correlation coefficient inter-residue co-evolving pairs between Hmmr and Lyn, the residue columns (as in Figure 3 B,C) with co-evolving amino-acid residues.

5. Discussion

The aim of this thesis was to create a script that would automatize and scale up the retrieval of orthologous protein sequences, creation of their MSA and TOLs, organise these for multicore parallel run of CAPS and combine CAPS output into comprehensible readout of adaptive co-evolution. These steps were achieved with the usage of Python programming language and public databases.

The retrieval of orthologous sequences was obtained from OrthoDB database via Python REST API client. The positive side of automatic retrieval is speed and avoiding human error. The drawback is that the database server may not always be available and the script should be tailored to other similar servers in this case. Other drawback could be that the data in the database is not of sufficient quality. Some of the orthologous sequences contained x – undefined amino acid residues, while others could have several homologues of which chosen one is not necessarily optimal. This affects the overall quality of MSA which is crucial for quality of orthologues MSA based co-evolution analysis.

The initial TOL for chosen 85 vertebrate species was obtained from Timetree server. Out of 85 species that are found in OrthoDB database there were three cases of species not found in the Timetree database (*Poecilia formosa, Astyanax mexicanus, Oreochromis niloticus*). In these cases the nearest related species of same genus that was found in Timetree was used for building of TOL and renamed to match the OrthoDB retrieved sequences. This accounted to negligible difference in TOL in the overall scale of evolution time distances.

The multiple sequence alignment was performed in two steps with MUSCLE and PRANK. MUSCLE provided a template alignment for PRANK which realigned the sequences taking also phylogenetic information into account. By visual inspection of final PRANK aligned MSAs the alignments of the input sequences were satisfactory. Based on obtained orthologous sequences TOLs of overlapping species to original 85 species TOL was created by pruning. Here special care had to be taken to have exactly matching naming of each sequence in individual protein MSA to the species in corresponding TOL.

The obtained MSAs and TOLs were sorted into individual folders each containing MSAs or TOLs of two proteins. This was done either in all possible pairs or one protein of interest versus potential partner’s combination. The former created exponential number of combinations whereas the latter creates as many folders as there are proteins. The difference in approaches comes into account when considering the calculation times that are far lower with the second approach when analysing hundreds of proteins.

The CAPS was first tested with several different input parameters for a set of five proteins involved in innate and adaptive immune response (TLR 1,2,6 and CD79a,b correspondingly) that may interact with each other. There was an increase in the found co-evolving proteins when lowering the stringency to avoid type one error - raising the alpha value from 0.0001 to default 0.001 or higher where there was no further change in the output. Unexpected, however, was the exclusion of the co-evolving pair (TLR 2 – TLR 6) by higher alpha values previously found by alpha 0.0001. There is no obvious explanation as to why this happened. Increasing the number of random cycles or gap threshold had no effect on the result which is also unexpected and is may be revealing the improper working of CAPS. The comparison of co-evolution determination from individual folders based on post-CAPS calculations showed two more pairs than the ordinary CAPS run on the same proteins in one folder. The reason for this may be in usage of only one chi squared test for which the input values were readily available in the output file. Whereas there might be other additional "filtering" tests that are performed differently by CAPS when only two proteins are in the folder than when there are more.

Next, the accuracy of CAPS co-evolution detection was determined with the usage of likely and unlikely interacting and therefore non/co-evolving proteins as CAPS input. CD79a, b, Lyn found at plasma membrane, all three interacting in B cell activation were used for highly interacting proteins with expected co-evolution, compared to two mitochondrial matrix proteins Clpp and Glrx5. Mitochondrial matrix is separated from plasma membrane with two membranes in a separate organelle. Direct interaction and high co-evolution with plasma membrane residing B cell activation proteins is highly unlikely. However, the CAPS analysis showed far greater co-evolution of B cell activation proteins paired with mitochondrial matrix proteins than within each functional group. This was repeated with several gap thresholds with no difference in the outcome. This unexpected outcome further undermined the likelihood of CAPS' ability to properly detect co-evolution.

Finally, two proteins CD79a and Lyn both involved in B cell activation were analysed with CAPS versus 538 protein hits found by non-raft APEX2 targeted proximity labelling/pull-down mass spectrometry. The top co-evolving protein partners were ranked and analysed for comparison in Cytoscape which allows the usage of tested biochemically obtained interaction information. By CD79a the top four CAPS found co-evolving partners had no true functional interaction as the only common interaction neighbour was Poly – Ubiquitin-C, which promotes protein recycling irrespective of protein functionality. By Lyn the CAPS output was better supported by Cytoscape output. However there are clearly more closely interacting partners to Lyn among the 538 proteins that were not even found to be co-evolving with CAPS. For this reason the residue pairs that are the basis for final "positive" inter-protein co-evolution of the top ranked CAPS co-evolving CD79a or Lyn were examined. It was revealed that CAPS does not filter out the MSA columns containing gaps as set by the input parameter of gap threshold of 1. CAPS does not filter out even the MSA columns containing more than 50% gaps as would be expected if default value was used in case the input parameter was ignored. Furthermore the gaps in the columns are used for calculation of correlation factors. As an example, the residue position column that contains a change in one amino acid to another at one stage in evolution will be highly correlating with the column where at the same time in evolution there is a gap (as seen in Figure 13 C and 14 C).

Unfortunately this issue could not be resolved because of the thesis time constraint and due to the discontinuation of the CAPS support research team, whose members were otherwise very helpful and cooperative.

6. Conclusions

The aims of this thesis were to:

- automatize and scale up the retrieval of orthologous sequences for hundreds of proteins identified from mass spectrometry or elsewhere

- create multiple sequence alignments and phylogenetic tree presentations for the retrieved orthologues of each protein

- prepare these as input files for multicore parallel run of CAPS – coevolution analysis programme on supercomputer

- combine CAPS multiple output files into comprehensible readout of adaptive co-evolution

Each of these steps was achieved. Unfortunately CAPS analysis did not perform as expected, the possible reasons discussed in section 5. The achieved automatic MSA and TOL creation can be used for other than CAPS co-evolutionary approaches which are practically all based on MSA input and may have higher predictive power (as shown in Figure 5). The other approach could be (Bio) Python pre-processing of MSAs by removal of gap-containing columns prior to CAPS analysis.

7. References

Altschul, Stephen F., Gish, Warren, Miller, Webb, Myers, Eugene W., and Lipman, David J (1990). Basic local alignment search tool. *J. Mol. Biol.* **215**, 403-410.

Altschul, Stephen F., Madden, Thomas L., Schaffer, Alejandro A., Zhang, Jinghui, Zhang, Zheng, Miller, Webb, and Lipman, David J (1997). Gapped BLAST+ and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.* **25**(17); 3389-3402.

Batista FD, Harwood NE (2009). The who, how and where of antigen presentation to B cells. *Nat Rev Immunol.* **9**(1):15-27.

Borghesi L, Milcarek C (2006). From B cell to plasma cell: regulation of V(D)J recombination and antibody secretion. *Immunol. Res.* **36** (1–3): 27–32.

Brocchieri L, Karlin S (2005). Protein length in eukaryotic and prokaryotic proteomes. *Nucleic Acids Res.* **33** (10): 3390–3400.

Camacho C, Coulouris G, Avagyan V, Ma N, Papadopoulos J, Bealer K, Madden TL (2009). BLAST+: architecture and applications. *BMC Bioinformatics*. **10**:421.

Carter RH (2006). B cells in health and disease. *Mayo Clin Proc*. **81**(3):377-84.

Casari, G., Sander, C. & Valencia, A (1995). A method to predict functional residues in proteins. *Nature Struct. Biol.* **2**, 171–178.

Chen C, Huang H, Wu CH (2017). Protein Bioinformatics Databases and Resources. *Methods Mol Biol.* **1558**:3-39.

Crick, F.H.C. (1958). "On Protein Synthesis". Symposia of the Society for Experimental Biology, Number XII: The Biological Replication of Macromolecules. Cambridge University Press.

Cock PA, Antao T, Chang JT, Chapman BA, Cox CJ, Dalke A, Friedberg I, Hamelryck T, Kauff F, Wilczynski B, de Hoon MJL (2009) Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*, **25**, 1422-1423

Dal Porto JM, Gauld SB, Merrell KT, Mills D, Pugh-Bernard AE, Cambier J (2004). B cell antigen receptor signaling 101. *Mol Immunol.* **41**(6-7):599-613.

Darwin, CR (1862). On the various contrivances by which British and foreign orchids are fertilised by insects. John Murray, London

De Las Rivas J, Fontanillo C (2010). Protein-protein interactions essentials: key concepts to building and analyzing interactome networks. *PLoS Comp Biol.* **6** (6): e1000807.

del Sol Mesa A, Pazos, F, Valencia, A (2003). Automatic methods for predicting functionally important residues. *J. Mol. Biol.* **326**, 1289–1302.

Edgar RC (2004). MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Res.* **32**(5):1792-7.

Ehrlich PR, Raven PH (1969). Differentiation of populations. *Science.* **165**(3899):1228-32.

Fares MA, McNally D, (2006). CAPS: coevolution analysis using protein sequences. *Bioinformatics*. **22**(22):2821-2.

Fares MA, Travers SA (2006). A novel method for detecting intramolecular coevolution: adding a further dimension to selective constraints analyses. *Genetics.* **173**(1):9-23.

Flajnik MF, Kasahara M (2010). Origin and evolution of the adaptive immune system: genetic events and selective pressures. *Nat Rev Genet.* **11**(1):47-59.

Göbel U, Sander C, Schneider R, Valencia A (1994). Correlated mutations and residue contacts in proteins. *Proteins.* **18**: 309–317.

Graur D, Li WH (2000). Fundamentals of Molecular Evolution (Second ed.). Sunderland, Massachusetts: Sinauer Associates, Inc.

Heap RE, Gant MS, Lamoliatte F, Peltier J, Trost M (2017). Mass spectrometry techniques for studying the ubiquitin system. *Biochem Soc Trans*. **45**(5):1137-1148.

Hedges SB, Marin J, Suleski M, Paymer M, Kumar S (2015). Tree of Life Reveals Clock-Like Speciation and Diversification. *Mol Biol Evol.* **32**: 835-845.

Henikoff S, Henikoff JG (1992). Amino acid substitution matrices from protein blocks, *Proc Natl Acad Sci USA* **89**: 10915-10919.

Huang Y, Temperley ND, Ren L, Smith J, Li N, Burt DW (2011). Molecular evolution of the vertebrate TLR1 gene family--a complex history of gene duplication, gene conversion, positive selection and co-evolution. *BMC Evol Biol.* **11**:149.

Janeway, Charles; Paul Travers; Mark Walport; Mark Shlomchik (2001). Immunobiology; Fifth Edition. Garland Science

Jensen RA (2001). Orthologs and paralogs - we need to get it right. *Genome Biol.* **2**(8):INTERACTIONS1002.

Juan D, Pazos F, Valencia A (2008). High-confidence prediction of global interactomes based on genomewide coevolutionary networks. *Proc Natl Acad Sci.* **105**: 934–939.

Juan D, Pazos F, Valencia A (2013). Emerging methods in protein co-evolution. *Nat Rev Genet*. **14**(4): 249-61.

King RC, Mulligan P, Stansfield W (2013). A Dictionary of Genetics. Oxford University Press

Lichtarge O, Bourne HR, Cohen FE (1996). An evolutionary trace method defines binding surfaces common to protein families. *J Mol Biol.* **257**: 342–358.

Lockless SW, Ranganathan R (1999). Evolutionarily conserved pathways of energetic connectivity in protein families. *Science.* **286**: 295–299.

Löytynoja A, Goldman N (2008). Phylogenyaware gap placement prevents errors in sequence alignment and evolutionary analysis. *Science.* **320**: 1632–1635.

Löytynoja A (2014). Phylogeny-aware alignment with PRANK. *Methods Mol Biol*. **1079**: 155-70.

Madaoui H, Guerois R (2008). Coevolution at protein complex interfaces can be detected by the complementarity trace with important impact for predictive docking. *Proc Natl Acad Sci*. **105**(22):7708-13.

Mateu MG, Fersht AR (1999). Mutually compensatory mutations during evolution of the tetramerization domain of tumor suppressor p53 lead to impaired hetero-oligomerization. *Proc Natl Acad Sci*. **96**: 3595–3599.

Mihalek I, Res I, Lichtarge O (2004). A family of evolutionentropy hybrid methods for ranking protein residues by importance. *J Mol. Biol*. **336**: 1265–1282.

Nirenberg MW, Matthaei JH (1961). The dependence of cell-free protein synthesis in E. coli upon naturally occurring or synthetic polyribonucleotides. *Proc Natl Acad Sci*. **47** (10): 1588–602.

Notredame C, Higgins DG, Heringa J (2000). T-Coffee: A novel method for fast and

accurate multiple sequence alignment. *J Mol Biol*. **302**(1):205-17.

Pazos F, Helmer-Citterich M, Ausiello G, Valencia A (1997) Correlated mutations contain information about protein–protein interaction. *J Mol Biol.* **271**: 511–523

Pazos F, Valencia A (2001). Similarity of phylogenetic trees as indicator of protein–protein interaction. *Protein Eng.* **14**: 609–614.

Pazos F, Valencia A (2002). In silico two-hybrid system for the selection of physically interacting protein pairs. *Proteins* **47**: 219–227.

Pazos F, Ranea JAG, Juan D, Sternberg MJE (2005). Assessing protein co-evolution in the context of the tree of life assists in the prediction of the interactome. *J Mol Biol*. **352**: 1002–1015.

Pazos F, Valencia A (2008). Protein co-evolution, co-adaptation and interactions. *EMBO J*. **27**(20):2648-55.

Pellegrini M, Marcotte EM, Thompson MJ, Eisenberg D, Yeates TO (1999). Assigning protein functions by comparative genome analysis: protein phylogenetic profiles. *Proc Natl Acad Sci.* **96**: 4285–4288.

Rhee HW, Zou P, Udeshi ND, Martell JD, Mootha VK, Carr SA, Ting AY (2013). Proteomic mapping of mitochondria in living cells via spatially restricted enzymatic tagging. *Science* **339**(6125):1328-1331.

Rausell A, Juan D, Pazos F, Valencia A (2010). Protein interactions and ligand binding: from protein subfamilies to functional specificity. *Proc Natl Acad Sci*. **107**: 1995–2000.

Reva B, Antipin Y, Sander C (2007). Determinants of protein function revealed by combinatorial entropy optimization. *Genome Biol*. **8**: R232

Sievers F, Wilm A, Dineen D, Gibson TJ, Karplus K, Li W, Lopez R, McWilliam H, Remmert M, Söding J, Thompson JD, Higgins DG (2011). Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. *Mol Syst Biol*. **7**:539.

Slupsky JR (2015). Enhancing BCR signals at the cell membrane. Blood. **125**(4):586-7.

Shannon P, Markiel A, Ozier O, Baliga NS, Wang JT, Ramage D, Amin N, Schwikowski B, Ideker T (2003). Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Res*. **13**(11):2498-504.

Shim Choi S, Li W, Lahn BT (2005). Robust signals of coevolution of interacting residues in mammalian proteomes identified by phylogeny-aided structural analysis. *Nat Genet* **37**: 1367–1371.

Tillier ERM, Charlebois RL (2009). The human protein coevolution network. *Genome Res*. **19**: 1861–1871.

van Rossum G (1995). Python tutorial, Technical Report CS-R9526, Centrum voor Wiskunde en Informatica

Watson JD, Crick FHC (1953). A structure for deoxyribose nucleic acid. *Nature*. **171**: 737–738.

Weigt M, White RA, Szurmant H, Hoch JA, Hwa T (2009). Identification of direct residue contacts in protein–protein interaction by message passing. *Proc Natl Acad Sci*. **106**: 67–72.

Wysocki VH, Resing KA, Zhang Q, Cheng G (2005). Mass spectrometry of peptides and proteins. *Methods*. **35**(3):211-22.

Zdobnov EM, Tegenfeldt F, Kuznetsov D, Waterhouse RM, Simão FA, Ioannidis P, Seppey M, Loetscher A, Kriventseva EV (2017). OrthoDB v9.1: cataloging evolutionary and functional annotations for animal, fungal, plant, archaeal, bacterial and viral orthologs. *Nucleic Acids Res.* **45**(D1):D744-D749.

8. Appendices

FIRST PYTHON SCRIPT CONTAINING STEPS 1-2-3-4-5-6-B:

```
import csv #for writing CSV with columns of species for each protein
from itertools import islice #for working with files, geting lines of choice-to extract the second line for species fasta
import itertools
import re #for string manipulation
import ast #for work with dictionaries
import urllib2
from lxml import etree
import os
from Bio import SeqIO #biopython to change OrthoDBFAST headers
import time
start_time = time.clock()
import sys
import urllib
import urllib2
import json
import time
import re #for strings
import os#for file path
import Bio
from Bio import AlignIO
from Bio import SeqIO
from Bio.SeqIO.FastaIO import SimpleFastaParser
from Bio.Align.Applications import ClustalwCommandline
from Bio import pairwise2
from Bio.SubsMat import MatrixInfo as matlist
matrix = matlist.blosum62
import subprocess #to manage supprocess
import sys
def launchWithoutConsole(command, args):
   """Launches 'command' windowless and waits until finished"""
   startupinfo = subprocess.STARTUPINFO()
   startupinfo.dwFlags |= subprocess.STARTF_USESHOWWINDOW
   return subprocess.Popen([command] + args, startupinfo=startupinfo).wait()
import requests
import time
def bordered(text):
   lines = text.splitlines()
   width = max(len(s) for s in lines)
   res = [unichr(0x2588) + unichr(0x2588)* width + unichr(0x2588)]
   for s in lines:
      res.append(unichr(0x2588) + (s + ' ' * width)[:width] + unichr(0x2588))
   res.append(unichr(0x2588) + unichr(0x2588) * width + unichr(0x2588))
   return '\n'.join(res)

start_time = time.clock()
seqfast=""
listofprotsspecies=[]###to make species list
singlespecieslist=[]
outype=".xml"#'.txt'#could be ".fasta" or ".tab"
############################################################################ STEP 1:UNIPROT IDENTIFIERS ######################
listofUPids = ["P11911","Q99KK9",                    "Q80V62", "P16277",             "P29351"] #### IMPORTANT PUT HERE THE LIST OF UNIPROT IDENTIFIERS, PROTEINS OF INTEREST!
filelistfasta=[]###########file list from which folders are made and files moved
filelistdnd=[]
from ete3 import Tree
def diff(first, second): #function for checking if the list of species in trees changes
      second = set(second)
      return [item for item in first if item not in second]
##################################################
def MusclePrank (symbol, RecName, GeneName):  (function containing step 5, 6 and B is declared before step 2)
   ################################################################# STEP 5: first part making MUSCLE MSA alignement and tree
   flname=symbol
   from Bio.Align.Applications import MuscleCommandline
   muscle_exe = r"E:\Program Files\muscle\muscle3.8.31_i86win32.exe" ######IMPORTANT CHANGE APROPRIATELY
   muscle_cline = MuscleCommandline(muscle_exe, input=flname+".fs",  tree1="tree.dnd")
   stdout, stderr = muscle_cline()
   from StringIO import StringIO
   from Bio import AlignIO
   startupinfo = subprocess.STARTUPINFO() ####to avoid command line window pop-up!!!
   startupinfo.dwFlags |= subprocess.STARTF_USESHOWWINDOW
   handle=StringIO(stdout)
   align = AlignIO.read(handle, "fasta")
   AlignIO.write(align, 'aligned.fasta', "fasta")
   print"phyloDraw"
   handle.close()

   ################################################################# STEP 6:second part doing prank MSA based on prealigned muscle FASTA and muscle phylogenetic
Newick tree
   from Bio.Align.Applications import PrankCommandline
   #I had to modify to include partaligned parameter!!! \Python27\Lib\site-packages\Bio\Align\Applications\_prank.py
   entryname=RecName+"_"+GeneName+"_"+symbol
   prank_cline = PrankCommandline(d="aligned.fasta",o=entryname, # prefix only!
                        f=8, # FASTA output
                        t="treetest.dnd",# noxml=True,
                        showtree=entryname,
                        iterate=15,partaligned=True,
                        prunetree=True, prunedata=True)#prunedata
   from StringIO import StringIO
   from Bio import AlignIO
   stdout, stderr = prank_cline()
   print "stdout",stdout
   stdout, stderr = prank_cline()
   print "stdout",stdout
   from Bio import Phylo
   tree = Phylo.read(entryname+".dnd", "newick")
   print bordered("PRINTING ASCII TREE: Phylo.draw_ascii(tree)")
   print bordered("UniprotACC: "+UniprotAcc)
   Phylo.draw_ascii(tree)
   filelistfasta.append(entryname+".fas")#
   filelistdnd.append(entryname+".dnd")#

   print bordered("UniprotACC: "+UniprotAcc)
   ########################################################## STEP B: preparation of TOL pruning to match the PRANK MSA output
   treewhole = Tree("treetest.dnd", format=1) #reading the original tree file
   tree = Tree(entryname+".dnd", format=1) #reading the tree file from prank, with less species
   trwhlnames=[leaf.name for leaf in treewhole] #making a list of species from original tree file with all species
   print "len(ntrwhlnames)",len(trwhlnames)
   print trwhlnames
   trnames=[leaf.name for leaf in tree]#making a list of species from prank tree file with all species
   print "len(trnames)",len(trnames)
   import copy
   newtree = copy.deepcopy(treewhole) #copy original tree file so it can be modified
   newtree.prune(trnames) #PRUNING THE ORIGINAL TREE WITH PRANK FILE, SO ONLY THE SPECIES FROM PRANK FILE REMAIN
   newtree.write(format=1, outfile=entryname+".tre") #writing the tree file with prank species
   ntrnames=[leaf.name for leaf in newtree] #making a list of species from prank-pruned original tree file with all species
   print "len(ntrnames)",len(ntrnames)
   print "diff(trwhlnames,ntrnames)",diff(trwhlnames,ntrnames) #checking that new prank-pruned original tree file really contains less species than original
   print bordered("UniprotACC: "+UniprotAcc)
for UniprotAcc in listofUPids:########FOR LOOP TO GO THROUGH ALL UNIPROT
IDENTIFIERS##############################################################################################
   try:
      print bordered("UniprotACC: "+UniprotAcc)
      singlespecieslist=[]#restarting single species list
      singlespecieslist.append(UniprotAcc)
      ##############################################################################STEP 2 - Online retrieval of protein sequence at Uniprot
      urlStr = 'http://www.uniprot.org/uniprot/'+UniprotAcc+outype
      response = requests.get(urlStr)
      with open('feed.xml', 'wb') as file:
          file.write(response.content)
      seqfast
      tree = etree.parse("feed.xml")
      root = tree.getroot()
      print root.tag
      print root.attrib

      for entry in root.findall('entry', root.nsmap):
          if entry.find('protein/recommendedName/fullName', root.nsmap) is None:  ####in case uniprot accession xml has missing information on this:
              RecName=""
          else:
              RecName = entry.find('protein/recommendedName/fullName', root.nsmap).text
          singlespecieslist.append(RecName)#second element uniprot protein name
          print unichr(0x2588),"Recommended Name: ",RecName
          if entry.find('gene/name', root.nsmap) is None:  ####in case uniprot accession xml has missing information on this:
              GeneName=""
          else:
              GeneName = entry.find('gene/name', root.nsmap).text
          print unichr(0x2588),"GeneName: ",GeneName
          singlespecieslist.append(GeneName)#third element
          for altName in entry.findall('protein/alternativeName/fullName', root.nsmap):
              print unichr(0x2588),"Alternative Name: ",altName.text
          for subcelloc in entry.findall("comment/subcellularLocation/location", root.nsmap):
              print unichr(0x2588),"Subcell Loc: ",subcelloc.text
          sequence = entry.find('sequence', root.nsmap).text
          seqfast=">"+UniprotAcc+sequence
          print "Sequence Length: ",len(sequence)
          singlespecieslist.append(str(len(sequence)))
          print seqfast
          for fulnm3 in root.iter('fullName'):
              print fulnm3.text
      with open('seqfast.fasta', 'wb') as file:
          file.write(seqfast)
      #############part in case OrthoDB would update the matching of UniprotID to OrthoDB
identifiers#########################################################################GETTING DATA FROM ODB SITE
      #OrDBid="Q9HCC0" #works also directly from uniprot!
      OrDBid="-"#UniprotAcc #testing false input
      #the output is {"status": "ok", "message": "no clusters found", "data": [], "count": 0}
      #OrDBid="1433B_MOUSE"#this should be read from uniprot TAB or BLASTED
      urlStr = "http://www.orthodb.org/fasta?query="+OrDBid+"&level=7742&species=&universal=&singlecopy="
      response = requests.get(urlStr)
      print response.content
      with open('original.fs', 'w') as file:
          file.write(response.content)
      #print file
      #add species names:
      #orthoDB species IDS D:\MASTERS\PYTHON\OrthoDB\odb9v1_species.tab
      urlStr = "http://www.orthodb.org/tab?query="+OrDBid+"&level=7742&species=&universal=&singlecopy="
      response = requests.get(urlStr)
      print response.content
      with open('original.tab', 'w') as file:
          file.write(response.content)
      #print file    ####IF RESPONSE FROM ODB EMPTY THEN I SEARCH WITH BLAST FOR CLOSEST OTHER UNIPROT OR ODB PROTEIN
      print response.content
      print
      if ast.literal_eval(response.content)["count"]==0: #### with ast.literal_eval turning string that looks like dictionary into dictionary
          print "EMPTY"

      ############BLASTING UNIPROT DATABASE, because sometimes Uniprot can't find its own identifiers!!!
          from Bio.Blast.Applications import NcbiblastpCommandline  #for BLAST

          db1 = "uniprot-ODB.fasta"  #Uniprot mouse PROTEIN FASTA DATABASE, that has also ODB identifiers
          ########IMPORTANT CHANGE THE PATH TO THE UNIPROT SEQUENCE BELLOW!
          blast_cline = NcbiblastpCommandline(query="D:\MASTERS\PYTHON\OrthoDB\seqfast.fasta", db=db1, evalue=0.001, outfmt=5,
out="D:\MASTERS\PYTHON\OrthoDB\LocUniprot_BLASToutput.xml")
          stdout, stderr = blast_cline()
          print(stdout,stderr)
          print "\n"
          print bordered("UNIPROT BLAST")
          print bordered("%.2f" %(time.clock() - start_time)+" seconds") #time it needs for the whole script!
          from Bio.Blast import NCBIXML  #using Biopython library
          result_handle = open("LocUniprot_BLASToutput.xml")
          blast_record = NCBIXML.read(result_handle)  #using Biopython library

          first = True ####geting just the first top alignement
          for alignment in blast_record.alignments: #using Biopython library
              for hsp in alignment.hsps: #using Biopython library
                  if first:
                      #if hsp.expect < E_VALUE_THRESH:
                      print('****Alignment****')
                      print('sequence altitle:', alignment.title)
                      #print alignment.title  #string containing ensembl IDs
                      print('length:', alignment.length)
                      print('e value:', hsp.expect)
                      print(hsp.query[0:75] + '...')
                      print(hsp.match[0:75] + '...')
                      print(hsp.sbjct[0:75] + '...')
                      ALTTL=alignment.title
                      UNIPRidx=ALTTL.find("|", 17, 23)+1#len(ALTTL))+3
                      UNIPRidx2=ALTTL.find("|",  23, 35)#UNIPRidx, len(ALTTL))
                      UNIPRacc=ALTTL[UNIPRidx:UNIPRidx2]
                      print bordered("BLASTED UNIPROTACCESSION"+UNIPRacc)

                      print "orig UNIPR sq",(len(sequence))  #comparing uniprot input seq length and ensembl hit sequence length
                      print "alt UNIPR sq",int(alignment.length)
                      uniprlnght=int(alignment.length)
                      print unichr(0x2588),"% length difference of orig/alt sq:","%.2f" % ((len(sequence)/float(alignment.length)-len(sequence)/(alignment.length))*100), "%"
                      Uniprotsq=hsp.sbjct
                      Sqnobrks=sequence.replace('\n', '').replace('\r', '')
                      print "Sqnobrks",Sqnobrks
                      first = False #ending to make sure only first - most similar hit is taken
```

##**STEP 3**: **LOCAL BLASTING for ORTHODB IDs**
##

```
        from Bio.Blast.Applications import NcbiblastpCommandline  #for BLAST

        db2 = "MusMusculusODB.fs"  #Uniprot mouse PROTEIN FASTA DATABASE, that has also ODB identifiers
        ##change the path as needed below!
        blast_cline = NcbiblastpCommandline(query="D:\MASTERS\PYTHON\OrthoDB\seqfast.fasta", db=db2, evalue=0.001, outfmt=5,
out="D:\MASTERS\PYTHON\OrthoDB\LocODB_BLASToutput.xml")
        stdout, stderr = blast_cline()
        print(stdout,stderr)

        ###parse the LocENSMBL_BLASToutput.xml

        print bordered("UniprotACC: "+UniprotAcc)
        print "\n"

        print bordered("ODB BLAST")
        print bordered("%.2f" %(time.clock() - start_time)+" seconds")  #time it needs for the whole script!
        from Bio.Blast import NCBIXML  #using Biopython library
        result_handle = open("LocODB_BLASToutput.xml")
        blast_record = NCBIXML.read(result_handle)  #using Biopython library

        first = True ####geting just the first top alignement
        for alignment in blast_record.alignments: #using Biopython library
            for hsp in alignment.hsps: #using Biopython library
                if first:
                    #if hsp.expect < E_VALUE_THRESH:
                    print('****Alignment****')
                    print('sequence:', alignment.title)
                    print alignment.title  #string containing ensembl IDs
                    print('length:', alignment.length)
                    print('e value:', hsp.expect)
                    print(hsp.query[0:75] + '...')
                    print(hsp.match[0:75] + '...')
                    print(hsp.sbjct[0:75] + '...')

                    hsp.sbjct


                    ALTTL=alignment.title
                    ODBidx=ALTTL.find("10090:", 0, len(ALTTL)) ##we are looking through mouse proteins with IDs that start with Mus musculus TAXid 10090
```

```
                        ODBacc=ALTTL[ODBidx:len(ALTTL)]
                        print ODBacc
                        print bordered("BLASTED ODBACCESSION"+ODBacc)
                        print "orig UNIPR sq",(len(sequence))  #comparing uniprot input seq length and ensembl hit sequence length
                        print "ODB sq",int(alignment.length)
                        print unichr(0x2588),"% length difference of origUnipr/ODB sq:","%.2f" % ((len(sequence)/float(alignment.length)-len(sequence)/(alignment.length))*100), "%"
                        ODBsq=hsp.sbjct
                        #ODBscore=pairwise2.align.globaldx(Sqnobrks, ODBsq, matrix, score_only=1)####PAIRWISE SCORING OF SIMILARITY, NO GAP PENALTY CURRENTLY
                        #print "Pairwise alignement score is: ",ODBscore
                        print bordered("comparing uniprt ODB BLAST:")
                        print unichr(0x2588),"% length difference of blast Unipr/ODB sq:","%.2f" % (( uniprlnght/float(alignment.length)- uniprlnght/(alignment.length))*100), "%"

                        #ODBUniprtscore=pairwise2.align.globaldx(Uniprotsq, ODBsq, matrix, score_only=1)####PAIRWISE SCORING OF SIMILARITY, NO GAP PENALTY CURRENTLY
                        #print "Pairwise alignement score is: ",ODBUniprtscore

                        #COULD SET SOME THRESHOLD HOW HIGH SHOULD THE SCORE OR LENGHT SIMILARITY BE IN ORDER TO PROCEED
                        first = False #ending to make sure only first - most similar hit is taken

            ##############################################################################STEP 4: RETRIEVAL OF OrthoDB Orthologous sequences, parsing and filtering them
for homologs
            #OrDBid="Q9HCC0" #works also directly from uniprot!
            OrDBid=ODBacc#OR ODBacc #EITHER FROM UNIPROT OR FROM ODB BLAST? #THINK ABOUT IT, WHICH COULD BE BETTER, ARE THERE SOME EXCLUSIVE CASES ETC...
            #the output is {"status": "ok", "message": "no clusters found", "data": [], "count": 0}

            #OrDBid="1433B_MOUSE"#this should be read from uniprot TAB or BLASTED
            urlStrfasta = "http://www.orthodb.org/fasta?query="+OrDBid+"&level=7742&species=&universal=&singlecopy="

            responsefasta = requests.get(urlStrfasta)
            print responsefasta.content
            urlStrtab = "http://www.orthodb.org/tab?query="+OrDBid+"&level=7742&species=&universal=&singlecopy="

            responsetab = requests.get(urlStrtab)
            print responsetab.content
            with open('original.fs', 'w') as original, open('original.tab', 'w') as originaltab:#, open('corrected.fs', 'w') as corrected:
                original.write(responsefasta.content)
                original.close
                originaltab.write(responsetab.content)
                originaltab.close
            print bordered("UniprotACC: "+UniprotAcc)

        oldinstance=""#for comparison of homologs
        maxspecscore=0#for pairwise scoring of sequences
        maxrecord=0
        previousrecord=0

            #SOME SPECIES HAVE SEVERAL HOMOLOGS, SO WE NEED TO FIND WHICH THOSE ARE AND COMPARE THEM TO FIND THE HOMOLOG, THAT IS CLOSEST TO THE MOUSE ORIGINAL
SEQUENCE
        with open('original.tab', 'r') as originaltab:  ####just to get first instance of species
            for firstline in islice(originaltab, 1, 2): #taking second line, because the first one is the header
                oldinstance=(re.split(r'\t+', firstline)[4].replace(" ", "_")).lower()
                print "#######SLICENOW",oldinstance
            originaltab.close
        first = 1
        with open('original.fs', 'r') as original:  ####just to get first instance of species
            records = SeqIO.parse(original, 'fasta')
            for record in records:#, 0, 1): #taking first line record
                if first:
                    print "RECORD: ",record
                    print "RECORDid: ",record.id
                    print "RECORDseq: ",record.seq
                    print "RECORDname: ",record.name
                    print "RECORDdesc: ",record.description
                    record.description=""
                    record.id=oldinstance#+" "+record.id
                    previousrecord=record
                    maxrecord=record
                    first = False
            original.close
        print bordered("UniprotACC: "+UniprotAcc)
        print "#########CHECKING after file closing########: ",maxrecord

        print bordered("HEAVY COMPUTING OF PAIRWISE ALIGNEMENTS, BE PATIENT!")
        with open('original.fs', 'r') as original, open('original.tab', 'r') as originaltab, open(UniprotAcc+'.fs', 'w') as corrected:

            records = SeqIO.parse(original, 'fasta')
            next(originaltab)#skipping header row by tab file
            next(originaltab)
            next(records)
            i=0
            for record, line in zip(records, originaltab):
                i=i+1
                newinstance=(re.split(r'\t+', line)[4].replace(" ", "_")).lower()
                #print record.id, "\n", i,". "
                #print re.split(r'\t+', line)[4].replace(" ", "_")#splits original line by tabs, takes 4th element -species name and replaces space with underscore
                if newinstance!=oldinstance:#checking if the current species isnt the same as species in the previous instance
                    record.id=newinstance#+" "+record.id
                    record.description=""
                    SeqIO.write(previousrecord, corrected, 'fasta')#if it is not the same, now it will save the previous species (max record) into fasta
                    singlespecieslist.append(oldinstance)

                    #oldinstance=newinstance #updating species instance
                    maxrecord=record #since this is new species maxrecord is restarted
                    previousrecord=record #as well as previous record is updated
                    maxspecscore=(( len(Sqnobrks)/float(len(record.seq))- len(Sqnobrks)/len(record.seq))*100)#len#pairwise2.align.globaldx(Sqnobrks, record.seq, matrix, score_only=1)
                    #because the pairwise alignment is way to slow I simply used the best length match

                    #since this is new species maxscore needs to be restarted
                        ###at this point I could make a special list for a file with just species names in columns

                else: #in case current species is the same as previous
                    newspecscore=(( len(Sqnobrks)/float(len(record.seq))- len(Sqnobrks)/len(record.seq))*100)#pairwise2.align.globaldx(Sqnobrks, record.seq, matrix, score_only=1) #comparing
the pairwise score of current sequence with uniprot
                    if newspecscore<maxspecscore:#if the new pairwise score is lower than max score
                        previousrecord=maxrecord#then "previous record" will be the record with so far max score (maxrecord), and the current record will be skipped
                    else: #if the new pairwis score is not lower (but higher) than the previous max score
                        maxspecscore=newspecscore #then new score becomes maxspecscore
                        record.id=newinstance#+" "+record.id
                        record.description=""
                        maxrecord=record #the current record becomes maxrecord
                        previousrecord=record #and also "previousrecord" is updated
                oldinstance=newinstance #so next round in for loop, next line will have the previous line to compare to...
            original.close
            originaltab.close
            corrected.close
        MusclePrank(UniprotAcc, RecName, GeneName)
        listofprotsspecies.append(singlespecieslist)#adding individual proteins specieslist
    except Exception as e: ####PRINTOUT OF ALL THE ERRORS FOR EACH UNIPROT IDENTIFIER THAT COULDNT BE PROCESSED
        try:
            with open('errors.txt', 'a') as errorsf:
                errorsf.write("UniprotACC: "+UniprotAcc+"\n")
                errorsf.write("RecName: "+RecName+"\n")
                errorsf.write("GeneName: "+GeneName+"\n")
                print str(e)
                errorsf.write(str(e)+"\n")
                errorsf.write("\n")
                errorsf.close
        except:
            pass
        pass
export_data = itertools.izip_longest(*listofprotsspecies, fillvalue='-')

print export_data
with open('allprotsspecieslist.txt', 'w') as myfile:
     wr = csv.writer(myfile)
     wr.writerows(export_data)
myfile.close()

print bordered("THE END: "+ "%.2f" %(time.clock() - start_time)+ " seconds") #time it needs for the whole script!

print 5*unichr(9989)
```

SECOND PYTHON SCRIPT CONTAINING STEP 7

```
#######################################################################################STEP 7: COMBINING THE FILES FOR ALL POSSIBLE PAIRS OF PROTEINS


mypathfas=r"D:\MASTERS\TEST2\FAS" #IMPORTANT copy here the path to your tre files
mypathtre=r"D:\MASTERS\TEST2\TRE" #IMPORTANT copy here the path to your fas files
#make sure names and number of files in each folder match

size=2  #IMPORTANT  put in the number of files you want per folder
#if set to 2 it will make all possible pairs so each protein has a pair with another one from the source folder
#if you want all the files to be combined just with one repeating protein set to 1, but specify the file to be added bellow (un/comment the command)

import os.path
import itertools
import os
dir_path = os.path.dirname(os.path.realpath(__file__))#current path where this .py is could be changed
#put all the files into this dir
from os import listdir
from os.path import isfile, join
import shutil
def srtngfls(mypath, size):
    onlyfiles = [f for f in listdir(mypath) if isfile(join(mypath, f))]
    parentfolder=os.path.abspath(os.path.join(mypath, os.pardir))
    dirsorting=1

    filename, file_extension = os.path.splitext(onlyfiles[0])
    file_extension = file_extension.translate(None, '.')
    if dirsorting==1:
        print onlyfiles
        print file_extension
        #### REDISTRIBUTING THE FILES INTO PAIRED
FOLDERS#########################################################################################################
            ###first creating folders and copying FASTA files
            iterator=0
            for pair in itertools.combinations(onlyfiles, size):
                print pair
                iterator=iterator+1
                if not os.path.exists(parentfolder+"\\folder"+str(iterator)+"\\"+file_extension+str(iterator)):
                    os.makedirs(parentfolder+"\\folder"+str(iterator)+"\\"+file_extension+str(iterator))
                #shutil.copy2('/src/file.ext', '/dst/dir')
                x=0
                for x in range(0,size):
                    shutil.copy2(mypath+"\\"+pair[x], parentfolder+"\\folder"+str(iterator)+"\\"+file_extension+str(iterator))
                    #shutil.copy2(mypath+"\\"+pair[1], parentfolder+"\\folder"+str(iterator)+"\\"+file_extension+str(iterator))
                    #shutil.copy2(mypath+"\\"+pair[2], parentfolder+"\\folder"+str(iterator)+"\\"+file_extension+str(iterator))

                #fas and tre files need to be set into parent directory
                #UNCOMMENT AND SPECIFY THE FILENAME AND PATH TO THE FILE YOU WANT TO BE ADDED TO EACH FOLDER
                #shutil.copy2(parentfolder+"\\"+"lidIgAIntrct_B-cell antigen receptor complex-associated protein alpha chain_Cd79a_P11911."+file_extension,
parentfolder+"\\folder"+str(iterator)+"\\"+file_extension+str(iterator))
                #shutil.copy2(parentfolder+"\\"+"Sarcolipin_Sln_Q9CQD6."+file_extension, parentfolder+"\\folder"+str(iterator)+"\\"+file_extension+str(iterator))

srtngfls(mypathtre, size) #calling of function for tre
srtngfls(mypathfas, size) #and fas files
```

COMMAND LIST FOR STEP 8 CAPS PARALLEL MULTICORE RUNNING IN LINUX ENVIRONMENT ON TAITO CSC SUPERCOMPUTER

```
time (cd /wrk/sustarvi/TEST/folder128759 && /homeappl/home/sustarvi/appl_taito/caps2/caps -F fas128759/ --inter -T tre128759/ -g 0.6 -H mus_musculus -c)
time (cd /wrk/sustarvi/TEST/folder128760 && /homeappl/home/sustarvi/appl_taito/caps2/caps -F fas128760/ --inter -T tre128760/ -g 0.6 -H mus_musculus -c)
time (cd /wrk/sustarvi/TEST/folder128761 && /homeappl/home/sustarvi/appl_taito/caps2/caps -F fas128761/ --inter -T tre128761/ -g 0.6 -H mus_musculus -c)
time (cd /wrk/sustarvi/TEST/folder128762 && /homeappl/home/sustarvi/appl_taito/caps2/caps -F fas128762/ --inter -T tre128762/ -g 0.6 -H mus_musculus -c)
time (cd /wrk/sustarvi/TEST/folder128763 && /homeappl/home/sustarvi/appl_taito/caps2/caps -F fas128763/ --inter -T tre128763/ -g 0.6 -H mus_musculus -c)
time (cd /wrk/sustarvi/TEST/folder128764 && /homeappl/home/sustarvi/appl_taito/caps2/caps -F fas128764/ --inter -T tre128764/ -g 0.6 -H mus_musculus -c)
........
```

BASH SCRIPT FOR STEP 8 IF ABOVE COMMAND LIST CONTAINS >700 COMMANS (LINES) IT NEEDS TO BE SPLIT AND EACH SUB-COMMAND LIST NEEDS TO BE RUN SUBSEQUENTLY, THIS BASH SCRIPT AUTOMATIZES THE AUTOMATIC RUN THROUGH ALL SUB-COMMAND LISTS

```
#!/bin/bash
sbatch_commandlist -commands xaa
sbatch_commandlist -commands xab
sbatch_commandlist -commands xac
sbatch_commandlist -commands xad
sbatch_commandlist -commands xae
sbatch_commandlist -commands xaf
sbatch_commandlist -commands xag
sbatch_commandlist -commands xah
sbatch_commandlist -commands xai
sbatch_commandlist -commands xaj
sbatch_commandlist -commands xak
...
```

COMMAND LIST FOR STEP 8 MOVEMENT AND SIMULTANEOUS RENAMING OF CAPS OUTPUT FROM EACH INDIVIDUAL SUBFOLDER INTO ONE COMMON FOLDER FOR ARCHIVING AND DOWNLOAD FROM SUPERCOMPUTER

```
mv /wrk/sustarvi/TEST/folder128759/coev_inter.csv /wrk/sustarvi/TEST/OUTPUT/foldercoev_inter128759.csv
mv /wrk/sustarvi/TEST/folder128760/coev_inter.csv /wrk/sustarvi/TEST/OUTPUT/foldercoev_inter128760.csv
mv /wrk/sustarvi/TEST/folder128761/coev_inter.csv /wrk/sustarvi/TEST/OUTPUT/foldercoev_inter128761.csv
mv /wrk/sustarvi/TEST/folder128762/coev_inter.csv /wrk/sustarvi/TEST/OUTPUT/foldercoev_inter128762.csv
mv /wrk/sustarvi/TEST/folder128763/coev_inter.csv /wrk/sustarvi/TEST/OUTPUT/foldercoev_inter128763.csv
mv /wrk/sustarvi/TEST/folder128764/coev_inter.csv /wrk/sustarvi/TEST/OUTPUT/foldercoev_inter128764.csv
mv /wrk/sustarvi/TEST/folder128765/coev_inter.csv /wrk/sustarvi/TEST/OUTPUT/foldercoev_inter128765.csv
...
```

THIRD PYTHON SCRIPT CONTAINING STEP 9

```
##########################################################################################STEP 9: EXTRACTION OF TOP ROWS IN CAPS OUTPUT COEV_INTER FILES

mypath=r"D:\MASTERS\TEST2\IgA12720_AllFiles\foldercoev" #put here the path to the folder with CAPS output files

import re
import csv
import itertools
import os
import os.path
from os import listdir
from os.path import isfile, join
import shutil
onlyfiles = [f for f in listdir(mypath) if (isfile(join(mypath, f)) & f.endswith('.csv'))]
for file in onlyfiles:
    print file
    with open(mypath+"\\"+file, 'rb') as csvfile:
        spamreader = csv.reader(csvfile, delimiter='\t')#, quotechar='|')
        templist=[]
        indtemplist=[str(csvfile)]
        for grow in spamreader:
            templist.append(grow)
        if len(templist)>2:
            for element in templist[1]:#[row]:
                print "ELEMENT", element," type: ",type(element)
                element = re.sub("[']", "", element)
                tempsbel=element.split("_")
                for sbelement in tempsbel:
                    print sbelement
                    indtemplist.append(sbelement)
                    savestr='\t'.join(indtemplist)
                    print "savestr", savestr
            with open(mypath+"\\"+'extractedCAPSoutput.txt', 'a') as pfile:
                pfile.write(savestr+'\n')
                pfile.close
```

LIST OF MS HIT PROTEINS:
Cep55,Pacsin2,Hmmr,Rnf219,Arhgap30,Psmd5,Sugt1,Rfc1,Prrc2a,Crocc,Blnk,Scamp3,Galk1,Eef1d,Rbm3,Dlgap5,
Mta2,Dnmt1,Fam83b,Caskin2,Psme2,Aldh16a1,Fgd6,Anxa11,Dhx30,Slc39a6,Pdlim2,Atxn2l,Eif3f,Anxa7,Scrib,Slc3
9a10,Snap29,Prdx5,Pag1,Eif4g1,Utp3,Nsdhl,Ybx1,Lyar,Nfkb1,Tax1bp1,Stx7,Kars,Il16,Abcf1,Ubap2,Msh6,Nup214,
Tufm,Lasp1,Atp5o,Acsl4,Lonp1,Pfas,U2surp,Eif4h,Hcfc1,Swap70,Was,Lrrc40,Smc4,Eif2a,Ranbp2,Pdcd6,Ncstn,Khs
rp,Edc4,Afdn,Nvl,Lrrc59,Stip1,Sri,Slamf6,Sec24a,Nup155,Mtmr14,Ikzf1,Pdcd6ip,Eif4g3,Nsun2,Slc38a2,Abcb7,Thu
mpd3,Elp1,Purb,Pdlim5,Ppp1r12a,Uba1,Eif3g,Snx27,Ap3b1,Prmt5,Gart,Nedd4,Ptk2b,Chd4,Nampt,Lig1,Asns,Igh
m,Csde1,G3bp2,Carm1,Fam98a,Usp6nl,Csrp1,Eps15,Arfgap2,Rangap1,Eif3b,Hadha,Cap1,Plaa,Parp1,Ints7,Dnaja1
,Eif3c,Adgrl2,Eif3h,Pacsin1,Mars,Erap1,Kif23,Trip13,Ablim1,Hsd17b12,Apbb1ip,Kif11,Dhx36,Sh3gl2,Rrp12,Cdv3,S
mc1a,Ehd4,Map2k1,Sars,Dag1,Zdhhc5,Ptpra,Ddx1,Gtpbp1,Sqstm1,Psma5,Hnrnpd,Rars,Ruvbl2,Snap23,Frs2,Nap1
l1,Wars,Gnas,Lcp1,Clic4,Inpp5d,Npepps,Cdc42,Smc2,Ddx19a,Pfkl,Itch,Kif5b,Bzw1,Rock1,Rbbp7,Slc4a7,Dock8,Pip
4k2b,Msh2,Mtss1,Rrm2,Tardbp,Zc3h15,Mtrex,Snrnp200,Sec31a,Erbin,Ipo9,Psmd11,Gnaz,Rab14,Myh9,Sf3b1,Cn
n2,Prkaca,Llgl1,Xpnpep1,Cdc37,Hspd1,Limd1,Top2a,Flna,Ptprc,Dhx15,Dnajc5,Pafah1b1,Dlg1,Rplp0,Rnpep,Hnrnp
f,Clasp2,Vav1,Elmo1,Rtcb,Abi1,Ddx17,Fcgr2,Pi4ka,Dbnl,Snd1,Slc16a1,Mcm4,Fyn,Polr2b,Kif15,Aldh18a1,Gspt1,M
ap3k7,Rab1b,Rala,Nxf1,Nedd9,Ywhaq,Myo1g,Plcg2,Wasf2,Usp9x,Picalm,Flnb,Lars,Timm50,H2D1,Ipo5,Ddx27,Can
d1,Mcm6,Lbr,Psmc2,Psd4,Supt16h,Slc3a2,Slc7a1,Tpp2,Dpysl2,Sec23b,Sptbn1,Osbp,Cd22,Akap12,Prpf6,Igsf3,Hdl
bp,Tln1,Dhx9,Ptpn23,Smarca4,Dock11,Eftud2,Ywhae,Ywhag,Psmd12,Psmd2,Psmd6,Psmc6,Psmc1,Psmc3,Psmc4,
Psmc5,Mpst,Pdpk1,Rps12,Rps13,Rps17,Rps19,Prkaa1,Mpp1,Rpl10a,Rpl30,Rpl35a,Rpl38,Rpl9,Abi3,Aacs,Acta1,Ac
tr3,Ahsa1,Adsl,Adss,Arf4,Arf6,Nudt5,Aars,Akr1b1,Actn4,Aimp1,Aimp2,Ano6,Ap1g1,Ap2b1,Bax,Nars,Atl3,Pfkp,Ata
d3,Aurkaip1,Btla,Cd79b,Cd72,Cd19,Grk2,Blvra,Tmco1,Cacybp,Prkar1a,Csnk1a1,Cavin1,Cnot1,Cd2ap,Cd82,Cnbp,C
lic1,Cpsf3,Copa,Copb1,Arcn1,Cc2d1b,Ccdc6,Ncaph,FAM120A,Cpne3,Coro1a,Coro1b,Coro1c,Ctps1,Chordc1,Dync
1h1,Cyfip2,Acot7,Dock2,Drg2,Dip2b,Dcbld1,Prim1,Polr1c,Dnaja2,Stt3a,Cbl,Ehd1,Ehd3,Eef1b,Elp3,Emb,Sh3glb1,E
ps15l1,Eif4a3,Etf1,Eif2s2,Eif2s3x,Eif3a,Eif4b,Eif5a,Eif5,Dis3,Xpo1,Cse1l,Capza1,Capza2,Capzb,Fcrl1,Aldoa,Gphn,G
csam,Gfpt1,Glrx3,Gstp2,Gstt3,Pygb,Gmps,Grb2,Rheb,Gnai2,Gnai3,Gna13,Hspa4,Hsph1,Hcls1,Hgs,Hnrnpl,Hip1r,H
prt1,Igbp1,Kpna3,Kpnb1,Impdh2,Isyna1,Iars,Klc1,Kifc1,Larp1,Lmnb1,Lrch1,Lrch4,Cd37,Lpxn,Hells,Lsp1,Mif,Mpp6,
Immt,Map1s,Mink1,Tomm40,Mapk1,Map4k1,Bub3,Mad2l1,Msn,Mthfd1l,Abcb1a,Trmt112,Naa15,Notch2,Nmral
1,Nsfl1c,Nup107,Nup160,Nup93,Nop58,Npm1,Fkbp4,Pes1,Farsa,Pebp1,Pip5k1a,Pgk1,Plscr3,Prpsap2,Atp2b1,Pls3
,Plekha2,Prpf8,Ddx20,Ddx6,Phb,Adrm1,Psme1,Prmt1,Pdia3,Fam49b,Mgea5,Sec61a1,Rp2,Rel,Glyr1,Racgap1,Rasa
2,Rasa3,G3bp1,Rasgrp2,Rac1,Rras2,Rab11a,Rab35,Rab5c,Rab7a,Rab8a,Rap1b,Upf1,Rfc5,Rpa1,Arhgdia,Rock2,Rr
m1,Rps6ka3,Nob1,Ruvbl1,Ahcyl1,Mat2a,Mat2a,Scfd1,Strap,Shcbp1,Spcs2,Stat1,Stat3,Scimp,Slc12a2,Slc12a6,Spt
an1,Srm,Ddx39b,Smc3,Fmr1,Stxbp5,Sdcbp,Ly9,Tbc1d15,Txnl1,Rela,Trim28,Rhoa,Tagln2,Tkt,Tpt1,Tm9sf3,Tnpo1,T
pi1,Tpm3,Tuba4a,Tubb5,Cdkn2a,Tsg101,Btk,Csk,Lyn,Ptpn1,Ptpn6,Usp10,Usp14,Otub1,Uba6,Cmpk1,Myo1c,Myo
1e,Vta1,Vapa,Nsf,Vdac3,Wdr91,Wbp2,Ythdf2,Zc3hav1,Znf622.