Ortholog detection using the reciprocal smallest distance algorithm

Running head: Finding orthologs with RSD

Dennis P. Wall and Todd DeLuca

The Computational Biology Initiative, Department of Systems Biology, Harvard Medical School

Corresponding author:

Dennis P. Wall, PhD

Director, Computational Biology Initiative

Department of Systems Biology

200 Longwood Ave, WA 536

Harvard Medical School

Boston, MA 02115

Email : dpwall@hms.harvard.edu

Phone : 617-432-3894

Fax : 617-432-5012

ABSTRACT

All protein coding genes have a phylogenetic history that when understood can lead to deep insights into the diversification or conservation of function, the evolution of developmental complexity, and the molecular basis of disease. One important part to reconstructing the relationships among genes in different organisms is an accurate method to find orthologs as well as an accurate measure of evolutionary diversification. The present chapter details such a method, called the reciprocal smallest distance algorithm (RSD). This approach improves upon the common procedure of taking reciprocal best blast hits (RBH) in the identification of orthologs by using global sequence alignment and maximum likelihood estimation of evolutionary distances to detect orthologs between two genomes. RSD finds many putative orthologs missed by RBH because it is less likely to be misled by the presence of close paralogs in genomes. The package offers a tremendous amount of flexibility in investigating parameter settings allowing the user to search for increasingly distant orthologs between highly divergent species, among other advantages. The flexibility of this tool makes it a unique and powerful addition to other available approaches for ortholog detection.

Key Words: orthologs, orthology, reciprocal smallest distance, reciprocal blast hit, maximum likelihood, molecular phylogenetics, phylogeny.

1. Introduction

The number of fully sequenced genomes is growing at an unprecedented rate. Presently there are 339 completed genomes and 1867 in various stages of construction (588 of these are Eukaryotic genomes) ***(1)***. The sheer number of newly sequenced genomes brings exciting opportunities and challenges to the field of comparative genomics. Paramount to this field is the

ability to accurately detect orthologs across genomes. Orthologs are used to answer, partially or completely, many important questions in a vast majority of biological fields ***(2)***. Aside from the important task of providing functional annotation to protein coding genes ***(2-4)***, orthologs have been used in understanding the evolution of Eukaryotes ***(5-8)***, predicting novel protein interactions ***(9-12)***, finding disease genes in 'model' organisms ***(13)***, determining the composition and of gene families ***(14)***, understanding coevolution of interacting proteins ***(15)***, comprehending the degree of compensation for deleterious mutations ***(16, 17)***, predicting deleterious alleles in humans ***(18, 19)***, as well as in measuring the effects of functional genomic variables on protein evolutionary rate in order to converge on a general model of protein evolution ***(20-29)***, and other areas of evolutionary genomics ***(30)***.

In all cases, being able to differentiate between orthologs (sequences that diverged from each other at a time of the species divergence) and paralogs (sequences that diverged at another time) is critical. For example, when comparing the evolutionary rates of proteins in the absence of a normalizing molecular clock, such as the rate of synonymous substitutions, estimates of evolutionary rate must be based upon comparisons between sequences that are strictly orthologous. Only if all sequence comparisons share the same time of divergence are protein evolutionary distances expected to be proportional to relative evolutionary rates.

The importance of this distinction between orthologs and paralogs is illustrated in Figure 1. The gene duplication event denoted by a black circle significantly predates the speciation event and results in two paralogous gene lineages. Treating as orthologs genes that are the result of this split, namely comparing either D or E with A, or comparing C with B, would yield inflated measures of evolutionary divergence between the two species and cause misinterpretation of functional similarity. Only the orthologous comparisons (A compared with C and B compared with D or E) would yield evolutionary distances indicative of the relative rates of protein evolution and an appropriate interpretation of functional equivalency.

A common procedure for identifying proteins that are putatively orthologous involves the identification of reciprocal best BLAST hits (RBH) (Hirsh and Fraser 2001; Jordan *et al.* 2002). Protein i in genome I is a reciprocal best hit of protein j in genome J if a forward search of genome J with protein i yields as the top hit protein j, and a reciprocal query of genome I with protein j yields as the top hit protein i. However, BLAST searches often return as the highest scoring hit a protein that is not the nearest phylogenetic neighbor of the query sequence ***(31)***. If the forward BLAST yields a paralogous best hit, regardless of whether the reciprocal BLAST corrects the error by recovering an actual ortholog, both pairs will be excluded. Thus, while RBH will rightfully prevent admission of the paralogous pair to the set of proteins for which relative evolutionary rates are estimated, it will also wrongly exclude an authentically orthologous pair from consideration. Despite this potential pitfall, perhaps the most commonly used orthology resources – Clusters of Orthologous Genes (C/KOGs) ***(14, 32)***, INPARANOID ***(33, 34)***, and the National Center For Biotechnology Information's (NCBI) resource, HomoloGene ***(35)*** -- are based on reciprocal best BLAST hits, and are thus likely to be incomplete or possibly erroneous.

In an effort to correct for this source of error, Wall et al. ***(36)*** developed a novel algorithm that preserves the safeguard of reciprocal genome queries, but is less susceptible to excluding an ortholog if a paralog is returned as the top hit in either the forward or reverse steps of a reciprocal BLAST query. This approach, termed the reciprocal smallest distance algorithm (*RSD*), has been shown to provide more comprehensive lists of orthologs than other methods that are based on BLAST alone. Furthermore, it is likely to be more accurate for identifying orthologs because it uses a phylogenetically-grounded measurement of similarity that matches our assumptions about how orthologs in different species have evolved.

RSD (Figure 2), which is the focus of this chapter, employs BLAST ***(37)*** as a first step, starting with a subject genome, *J*, and a protein query sequence, *i*, belonging to genome *I*. A set of hits,

H, exceeding a predefined significance threshold (e.g., $E < 10^{-20}$, though this is adjustable) is obtained. Then, using clustalW ***(38)***, each protein sequence in *H* is aligned separately with the original query sequence *i*. If the alignable region of the two sequences exceeds a threshold fraction of the alignment's total length (0.8 was the working cutoff in the original paper ***(36)***), the program PAML ***(39)*** (specifically, the package codeml) is used to obtain a maximum likelihood estimate of the number of amino acid substitutions separating the two protein sequences, given an empirical amino acid substitution rate matrix ***(40)***. The model under which a maximum likelihood estimate is obtained in *RSD* may include variation in evolutionary rate among protein sites, by assuming a gamma distribution of rate across sites and by setting the shape parameter of this distribution, α, to a level appropriate for the phylogenetic distance of the species being compared ***(41)***. (This parameter, α, may be altered to accommodate different degrees of phylogenetic distance.) Of all sequences in *H* for which an evolutionary distance is estimated, only *j*, the sequence yielding the shortest distance, is retained. This sequence *j* is then used for a reciprocal BLAST against genome *I*, retrieving a set of high scoring hits, *L*. If any hit from *L* is the original query sequence, *i*, the distance between *i* and *j* is retrieved from the set of smallest distances calculated previously. The remaining hits from *L* are then separately aligned with *j* and maximum likelihood distance estimates are calculated for these pairs as described above. If the protein sequence from *L* producing the shortest distance to *j* is the original query sequence, *i*, it is assumed that a true orthologous pair has been found and their evolutionary distance is retained (Figure 2).

RSD has been shown to find a substantially larger list of orthologs than reciprocal best BLAST ***(36)***. Thus, relative to BLAST alone, BLAST followed by evolutionary distance estimation based on pairwise global alignment is more likely to identify the same pair of sequences in both directions of a reciprocal query. This is presumably because BLAST can often return a paralog in at least one direction of a reciprocal query. When this occurs, the orthologous sequence is

likely to be among the sequences that are high-scoring, but not best, BLAST hits, an issue expected to be even more common in genomes that have high rates of gene duplication or gene conversion. In such cases, the global alignment and evolutionary distance estimation in *RSD* can recover the putative ortholog, revealing that it is in fact the nearest evolutionary neighbor of the query, even if not the best BLAST hit ***(36)***.

The original RSD algorithm has already been used in a variety of contexts to increase our understanding of both the nature of genome organization and to identify phylogenetic trends among different species ***(42-46)***. To enable its efficient usage, we have packaged the RSD algorithm into a user-friendly software suite, the operation of which is described in the sections below. RSD is available as a local standalone package, written primarily in Python and suitable for any Unix-based operating system. This program provides a tremendous amount of flexibility to navigate through parameter space, allowing alteration to the number of hits that are returned from the forward and reverse BLAST steps of the algorithm, changes to the BLAST E value threshold for the hits returned, the divergence parameter, and the shape parameter of the gamma distribution, α. Together these parameters can be used to fine-tune the accuracy of the distance calculations and ortholog assignments, an aspect that differentiates it from other ortholog detection applications. For these and other reasons, we expect RSD to make a solid addition to the arsenal of tools available for ortholog identification.

2. Materials

This section outlines the software download and installation steps required to run the RSD package on your local computer, as well as the software dependencies of RSD. RSD may be used only on Unix-based operating systems (e.g., Linux, Mac OS X), thus its installation and execution assumes a rudimentary understanding of Unix on the part of the user.

1. Download whole genome sequences in amino acid format from reliable public repositories. Specifically, download the complete set of peptides from at least two species in fastA-formatted files, one file per species *(see* **Notes 1 & 2**).
2. Download and install the Washington University BLAST 2.0 executables *(see* **Note 3**). The programs blastp (the protein based blast tool), xdget, and xdformat are required to run RSD.
3. Download and install the multiple sequence alignment program clustalW version 1.83 for Unix ***(47)*** *(see* **Notes 4 & 5**). The clustalW program is required to run RSD.
4. Download and install the phylogenetic analysis package PAML version 3.14 ***(39)*** *(see* **Note 6**). The codeml program is required to run RSD.
5. Download and install Python 2.4, including the bsddb module. *(see* **Note 7**).
6. Download RSD from http://cbi.med.harvard.edu/RSD.tar.gz *(see* **Note 8**)
7. Unpack the RSD archive to the directory you want it to be installed in. (**Note 8**)
8. After unpacking RSD, you should see a folder called "RSD_standalone"
9. Change directory to RSD_standalone and list its contents. You should see the following files and two directories (the README file contains similar instructions to what you see here).

BioUtilities.py	ReadFasta.py	examples
Blast_compute.py	Utility.py	execute.py
README	clustal2phylip	***formatter.py***
RSD.py	codeml.ctl_cp	jones.dat
RSD_common.py		

10. Three files listed above – RSD.py, Blast_compute.py, and formatter.py shown in ***bold and italics*** – are executed by the user. The remaining files are subsidiary components

of the programs that should not be altered. The directory called examples contains example data described later in this tutorial.

11. Blast_compute.py precomputes all-against-all reciprocal BLASTs building a forward blast hits file (FBH) and a reverse blast hits file (RBH) *(see* **Note 9**). At your Unix prompt type "python Blast_compute.py –-help" to see a list of options *(see* also Table 1 for a description of these options)
12. RSD.py is the central program of the RSD package. At your Unix prompt Type "python RSD.py –-help" to see a list of those options *(see* Table 1 for descriptions).
13. Formatter.py can be used to format a fastA-formatted proteome for use with RSD.py and Blast_compute.py. *(see* **Note 10**)

3. Methods

This section details the remaining formatting steps and Unix commands required to run the RSD package. We have streamlined RSD to require as few steps as possible for proper execution, distilling it down to two straightforward pieces – properly formatting your genomic databases and manipulating RSD's command-line options.

3.1 Preparing to run RSD from your local computer

1. Confirm that python, blastp, xdget, xdformat, clustalW, and codeml are functional and are in your path. *(see* **Note 11**)
2. Preformat your genomes by removing everything except for alphanumeric characters from the name lines of each sequence. Characters such as "|,?,$,%.#,^, (,)" can cause the program to quit unexpectedly. Name lines must also be truncated to no more than 22 characters. See the examples/Blast/blast_dbs subdirectory for examples of properly formatted databases. (also see **Note 10**)

3. Format your genomes by using the xdformat program from the Washington University BLAST 2.0 executables. The command is "xdformat –p –I name_of_your_fastA_formatted_database." This will create the indices used by the program **blastp**. (also see **Note 10**)

3.2 Running RSD

RSD has 4 chief modes of operation: (1) Running BLAST within the algorithm (hereafter referred to as "on the fly" computation mode) for an all-against-all search between two genomes (2) Running an all-against-all query using precomputed BLAST databases (thereby removing BLAST steps from the algorithm and speeding up the computation of reciprocal smallest distance significantly), (3) On the fly using user input sequences *(see* **Note 13**) to detect orthologs, and (4) Using user input sequences together with precomputed BLAST databases. Precomputed blast databases are generated via Blast_compute.py.

1. The RSD package comes with example files to test the installation and to give you an idea of how to set program options. The following steps use these example files to demonstrate the setup of common argument strings for running RSD.
2. Test computing BLAST hits between two genomes by typing: "python Blast_compute.py -q examples/Blast/blast_dbs/AGRO1_format.faa -s examples/Blast/blast_dbs/SINO1_format.faa -o example_AGRO1_vs_SINO1_blast_hits" *(see* **Note 14**)
3. Test operation mode 1 by typing: "python RSD.py --thresh=1e-10 --div=0.5 -q examples/Blast/blast_dbs/SINO1_format.faa -s examples/Blast/blast_dbs/AGRO1_format.faa -o examples/Orthologs/example_output_complete" *(see* **Note 14**)

4. Test operation mode 2 by typing: "python RSD.py --thresh=1e-10 --div=0.5 -q examples/Blast/blast_dbs/SINO1_format.faa -s examples/Blast/blast_dbs/AGRO1_format.faa -o examples/Orthologs/example_output_complete --fbh= examples/Blast/blast_results/example_blast_resultsF --rbh= examples/Blast/blast_results/example_blast_resultsR". *(see* **Note 14**)
5. Test operation mode 3 by typing: "python RSD.py --thresh=1e-10 --div=0.5 -u examples/userseqs_example -q examples/Blast/blast_dbs/SINO1_format.faa -s examples/Blast/blast_dbs/AGRO1_format.faa -o examples/Orthologs/example_output_userseqs" *(see* **Note 14**)
6. Test operation mode 4 by typing: "python RSD.py --thresh=1e-10 --div=0.5 -u examples/userseqs_example -q examples/Blast/blast_dbs/SINO1_format.faa -s examples/Blast/blast_dbs/AGRO1_format.faa -o examples/Orthologs/example_output --fbh= examples/Blast/blast_results/example_blast_resultsF --rbh= examples/Blast/blast_results/example_blast_resultsR" *(see* **Note 14**)
7. In all of the test runs above, the adjustable parameters, threshold (thresh) and divergence (div) are left unchanged. Note also that the other adjustable parameters nhits, fixalpha, and alpha are not included in the command line argument at all, and are thus left at their default values (Table 1). Nevertheless, these parameters can have a large effect on the size and content of the list of orthologs identified between two genomes and on the estimated evolutionary distance between proteins. Try altering these parameters variously to see how the lists of orthologs change. *(see* **Note 15&16**)

4. Notes

1. A sequence in fastA format begins with a single-line description, followed by lines of sequence data. The description line is distinguished from the sequence data by a greater-than (">") symbol in the first column. An example of an amino acid sequence in fastA format is:

>gi|4502175|ref|NP_001640.1| apical protein of Xenopus-like; APX homolog of Xenopus; apical protein, Xenopus laevis-like [Homo sapiens]
MEGAEPRARPERLAEAETRAADGGRLVEVQLSGGAPWGFTLKGGREHGEPLVITKIEEGSKAAAVDKLL
AGDEIVGINDIGLSGFRQEAICLVKGSHKTLKLVVKRRSELGWRPHSWHATKFSDSHPELAASPFTSTSG
CPSWSGRHHASSSSHDLSSSWEQTNLQRTLDHFSSLGSVDSLDHPSSRLSVAKSNSSIDHLGSHSKRD
SAYGSFSTSSSTPDHTLSKADTSSAENILYTVGLWEAPRQGGRQAQAAGDPQGSEEKLSCFPPRVPGD
SGKGPRPEYNAEPKLAAPGRSNFGPVWYVPDKKKAPSSPPPPPPPLRSDSFAATKSHEKAQGPVFSEA
AAAQHFTALAQAQPRGDRRPELTDRPWRSAHPGSLGKGSGGPGCPQEAHADGSWPPSKDGASSRLQ
ASLSSSDVRFPQSPHSGRHPPLYSDHSPLCADSLGQEPGAASFQNDSPPQVRGLSSCDQKLGSGWQG

Note that Blank lines are not allowed in the middle of fastA input.

2. We use the word genome in this chapter to refer to an organism's complete set of protein coding genes encoded into amino acids. Complete genomes are available from many on-line resources including:

http://www.ensembl.org/info/data/download.html ***(48)***

ftp://ftp.ncbi.nih.gov/genomes/

http://www.genomesonline.org/ ***(49)***

3. The Washington University BLAST 2.0 executables require that a license agreement be established at the following website http://blast.wustl.edu/licensing before downloading.

4. clustalW version 1.83 is available for downloadable at ftp://ftp.ebi.ac.uk/pub/software/unix/clustalw (the relevant file is "clustalw1.83.UNIX.tar.gz"). After downloading, you will need to gunzip, extract, and compile the program. See the README included with the download for details. Also see ***(38)***.

5. In principle, other alignment programs could be used so long as they can produce output in clustalW format.

6. PAML version 3.15 is available for download at http://abacus.gene.ucl.ac.uk/software/paml.html. After downloading you will need to compile the source code. The binary file used by RSD is codeml. Once compiled this binary executable may be moved to a directory of your choice (e.g. /usr/bin or /usr/local/bin).

7. Python version 2.4 can be downloaded from http://www.python.org/download/, except for Mac OS X users. See http://www.python.org/download/download_mac.html for details on downloading and installing python, including a working bsddb module.

8. After downloading RSD.tar.gz you will need to use tar and gunzip to decompress and extract the archive. Note that wget, gunzip, and tar are common Unix commands that you might need to use when downloading genomes as well as the other software packages.

9. The Blast_compute.py will conduct all-against-all queries between your two genomes building a forward blats hits file and a reverse blast hits file. These results contain all hits irrespective of the E value or score for each hit. Thus, you are free to explore the effects of different E value thresholds when using these precomputed blast databases in RSD processes. Using RSD in a precompute mode significantly speeds the search time, especially when testing the effects of

multiple, different parameter combinations on the identification of orthologs between two genomes.

10. formatter.py is a simple python script that can format whole genomic databases downloaded from NCBI. To run this program simply type "python formatter.py --infile name_of_database_to_format –outdir dir_to_place_formatted_db_and_indices". The script will eliminate any offending characters and create a new, properly formatted database for use with Washington University BLAST 2.0. If you have databases in formats other than NCBI's standard format that also have offending characters, you may need to alter the formatter.py script.

11. Test the functionality of the external programs blastp, xdget, xdformat, clustalW, and codeml by typing the names of the packages at your Unix command prompt. If the commands are recognized, the packages were installed correctly and are in your path.

12. The jones matrix is one of many possible matrices that can be used by the package codeml. PAML ships with several including dayhoff.dat, wag.dat, and mtmam.dat. You may also want to create your own amino acid substitution matrix ***(50, 51)***.

13. A user may create a fastA formatted file that contains select sequences from a query genome for which they would like to find orthologs in a particular subject genome. For example, a user may be interested in orthologous relationships for a set of proteins that belong to a particular biological pathway. The computation time for this calculation is significantly less than an all-against-all search.

14. Please note that the text in this command is wrapped (i.e. no hard returns).

15. The orthologous pair found by RSD is frequently not the top blast hit; 16% of the time it is the 2nd hit on the list and 7% of the time it is the third hit (Wall and DeLuca, *unpublished*). This highlights the differences between BLAST-based approaches and the RSD method for discovery of orthologs and stresses the importance of considering hits within a BLAST report deeper than the first or even the third best hit.

16. α, the shape parameter of the gamma distribution can make a significant impact on the estimates of divergence between two sequences, and in some cases can alter the composition of the list of orthologous pairs. For example, consider the following lists of orthologs between two species of fungus, Schizosaccharomyces pombe (S. pombe) and Saccharomyces cerevisiae (S. cerev.):

S.pombe	S.cerev.	D (α = 0.2)
19075528	6320000	2.668
19075302	6320003	0.9366
19112374	6320006	19.0
19113007	6320008	19.0
19075193	6320010	3.0685
19114542	6320011	19.0
19112732	6320013	0.6942
19114337	6320016	0.9877

S.pombe	S.cerev.	D_(α = 1.5)
19075528	6320000	0.7918
19075302	6320003	0.4386
19112374	6320006	2.6379
19113007	6320008	2.1115
19075193	6320010	0.8305
19114542	6320011	2.1817
19112732	6320013	0.3818
19114337	6320016	0.4624

Because these species last shared a common ancestor more that 330 million years ago and are quite distantly related, choosing a value of 1.5 for the shape parameter of the gamma

distribution will yield more reliable results ***(41)*** than the value of 0.2, which produces spurious distance estimates (the distance value of 19 is an upper limit of the codeml package and should not be trusted).

Acknowledgements

Many thanks to past and present members of the Computational Biology Initiative who provided advice and expertise, I-Hsien Wu, Tom Monaghan, Jian Pu, Saurav Singh, and Leon Peshkin.

References

1. Bernal, A., Ear, U., et al. (2001) Genomes OnLine Database (GOLD): a monitor of genome projects world-wide. Nucleic Acids Res **29**, 126-127.

2. Turchin, A. and Kohane, I. S. (2002) Gene homology resources on the World Wide Web. Physiol Genomics **11**, 165-177.

3. Pellegrini, M., Marcotte, E. M., et al. (1999) Assigning protein functions by comparative genome analysis: protein phylogenetic profiles. Proc Natl Acad Sci U S A **96**, 4285-4288.

4. Marcotte, E. M., Pellegrini, M., et al. (1999) Detecting protein function and protein-protein interactions from genome sequences. Science **285**, 751-753.

5. Dacks, J. B. and Doolittle, W. F. (2001) Reconstructing/deconstructing the earliest eukaryotes: how comparative genomics can help. Cell **107**, 419-425.

6. Lang, B. F., Seif, E., et al. (1999) A comparative genomics approach to the evolution of eukaryotes and their mitochondria. J Eukaryot Microbiol **46**, 320-326.

7. Rubin, G. M., Yandell, M. D., et al. (2000) Comparative genomics of the eukaryotes. Science **287**, 2204-2215.

8. Ureta-Vidal, A., Ettwiller, L., et al. (2003) Comparative genomics: genome-wide analysis in metazoan eukaryotes. Nat Rev Genet **4**, 251-262.

9. Espadaler, J., Aragues, R., et al. (2005) Detecting remotely related proteins by their interactions and sequence similarity. Proc Natl Acad Sci U S A **102**, 7151-7156.

10. Espadaler, J., Romero-Isart, O., et al. (2005) Prediction of protein-protein interactions using distant conservation of sequence patterns and structure relationships. Bioinformatics **21**, 3360-3368.

11. Matthews, L. R., Vaglio, P., et al. (2001) Identification of potential interaction networks using sequence-based searches for conserved protein-protein interactions or "interologs". Genome Res **11**, 2120-2126.

12. Kim, W. K., Bolser, D. M., et al. (2004) Large-scale co-evolution analysis of protein structural interlogues using the global protein structural interactome map (PSIMAP). Bioinformatics **20**, 1138-1150.

13. O'Brien, K. P., Westerlund, I., et al. (2004) OrthoDisease: a database of human disease orthologs. Hum Mutat **24**, 112-119.

14. Tatusov, R. L., Koonin, E. V., et al. (1997) A genomic perspective on protein families. Science **278**, 631-637.

15. Fraser, H. B., Hirsh, A. E., et al. (2004) Coevolution of gene expression among interacting proteins. Proc Natl Acad Sci U S A **101**, 9033-9038.

16. Kulathinal, R. J., Bettencourt, B. R., et al. (2004) Compensated deleterious mutations in insect genomes. Science **306**, 1553-1554.

17. Kondrashov, A. S., Sunyaev, S., et al. (2002) Dobzhansky-Muller incompatibilities in protein evolution. Proc Natl Acad Sci U S A **99**, 14878-14883.

18. Sunyaev, S., Kondrashov, F. A., et al. (2003) Impact of selection, mutation rate and genetic drift on human genetic variation. Hum Mol Genet **12**, 3325-3330.

19. Sunyaev, S., Ramensky, V., et al. (2001) Prediction of deleterious human alleles. Hum Mol Genet **10**, 591-597.

20. Fraser, H. B., Wall, D. P., et al. (2003) A simple dependence between protein evolution rate and the number of protein-protein interactions. BMC Evol Biol **3**, 11.

21. Herbeck, J. T. and Wall, D. P. (2005) Converging on a general model of protein evolution. Trends Biotechnol

22. Wall, D. P., Hirsh, A. E., et al. (2005) Functional genomic analysis of the rates of protein evolution. Proc Natl Acad Sci U S A **102**, 5483-5488.

23. Hirsh, A.E. and Fraser, H.B. (2001) Protein dispensability and rate of evolution. Nature **411**, 1046-1049.

24. Hurst, L. D. and Smith, N. G. (1999) Do essential genes evolve slowly? Curr Biol **9**, 747-750.

25. Jordan, I. K., Rogozin, I. B., et al. (2002) Essential genes are more evolutionarily conserved than are nonessential genes in bacteria. Genome Res. **12**, 962-968.

26. Krylov, D. M., Wolf, Y. I., et al. (2003) Gene loss, protein sequence divergence, gene dispensability, expression level, and interactivity are correlated in eukaryotic evolution. Genome Res. **13**, 2229-2235.

27. Yang, Jing, Gu, Zhenglong, et al. (2003) Rate of protein evolution versus fitness effect of gene deletion. Mol. Bol. Evol. **20**, 772-774.

28. Zhang, J. and He, X. (2005) Significant impact of protein dispensability on the instantaneous rate of protein evolution. Mol Biol Evol **22**, 1147-1155.

29. Rocha, E. P. and Danchin, A. (2004) An analysis of determinants of amino acids substitution rates in bacterial proteins. Mol. Biol. Evol. **21**, 108-116.

30. Koonin, E. V., Aravind, L., et al. (2000) The impact of comparative genomics on our understanding of evolution. Cell **101**, 573-576.

31. Koski, L. B. and Golding, G. B. (2001) The closest BLAST hit is often not the nearest neighbor. J Mol Evol **52**, 540-542.

32. Tatusov, R. L., Galperin, M. Y., et al. (2000) The COG database: a tool for genome-scale analysis of protein functions and evolution. Nucleic Acids Res **28**, 33-36.

33. O'Brien, K. P., Remm, M., et al. (2005) Inparanoid: a comprehensive database of eukaryotic orthologs. Nucleic Acids Res **33**, D476-480.

34. Remm, M., Storm, C. E., et al. (2001) Automatic clustering of orthologs and in-paralogs from pairwise species comparisons. J Mol Biol **314**, 1041-1052.

35. Wheeler, D. L., Barrett, T., et al. (2005) Database resources of the National Center for Biotechnology Information. Nucleic Acids Res **33**, D39-45.

36. Wall, D. P., Fraser, H. B., et al. (2003) Detecting putative orthologs. Bioinformatics **19**, 1710-1711.

37. Altschul, S. F., Gish, W., et al. (1990) Basic local alignment search tool. J Mol Biol **215**, 403-410.

38. Chenna, R., Sugawara, H., et al. (2003) Multiple sequence alignment with the Clustal series of programs. Nucleic Acids Res **31**, 3497-3500.

39. Yang, Z. (1997) PAML: a program package for phylogenetic analysis by maximum likelihood. Comput Appl Biosci **13**, 555-556.

40. Jones, D. T., Taylor, W. R., et al. (1992) The rapid generation of mutation data matrices from protein sequences. Comput Appl Biosci **8**, 275-282.

41. Nei, M., Xu, P., et al. (2001) Estimation of divergence times from multiprotein sequences for a few mammalian species and several distantly related organisms. Proc Natl Acad Sci U S A **98**, 2497-2502.

42. Degnan, P. H., Lazarus, A. B., et al. (2005) Genome sequence of Blochmannia pennsylvanicus indicates parallel evolutionary trends among bacterial mutualists of insects. Genome Res **15**, 1023-1033.

43. Gasch, A. P., Moses, A. M., et al. (2004) Conservation and evolution of cis-regulatory systems in ascomycete fungi. PLoS Biol **2**, e398.

44. Nayak, S., Goree, J., et al. (2005) fog-2 and the evolution of self-fertile hermaphroditism in Caenorhabditis. PLoS Biol **3**, e6.

45. Wu, H., Su, Z., et al. (2005) Prediction of functional modules based on comparative genome analysis and Gene Ontology application. Nucleic Acids Res **33**, 2822-2837.

46. Wuchty, S. (2004) Evolution and topology in the yeast protein interaction network. Genome Res **14**, 1310-1314.

47. Thompson, J. D., Higgins, D. G., et al. (1994) CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. Nucleic Acids Res **22**, 4673-4680.

48. Birney, E., Andrews, D., et al. (2006) Ensembl 2006. Nucleic Acids Res **34**, D556-561.

49. Liolios, K., Tavernarakis, N., et al. (2006) The Genomes On Line Database (GOLD) v.2: a monitor of genome projects worldwide. Nucleic Acids Res **34**, D332-334.

50. Bastien, O., Roy, S., et al. (2005) Construction of non-symmetric substitution matrices derived from proteomes with biased amino acid distributions. C R Biol **328**, 445-453.

51. Olsen, R. and Loomis, W. F. (2005) A collection of amino acid replacement matrices derived from clusters of orthologs. J Mol Evol **61**, 659-665.

52. Fitch, W. M. (1970) Distinguishing homologous from analogous proteins. Syst Zool **19**, 99-113.

TABLE 1: The two user-operated RSD programs and their associated options.

Program	Options	Description
Blast_compute.py	-h, --help	Shows the parameter options for the program
	-q QUERYDB, --querydb=QUERYDB	Path to the query database (fastA format)
	-s SUBJECTDB, --subjectdb=SUBJECTDB	Path to the subject db (fastA format)
	-o OUTFILE, --outfile=OUTFILE	Path to the blast db output file (any name you choose)
RSD.py	-h, --help	Shows the parameter options for the program
	--thresh=THRESH	Required. E-value threshold for blast subcomponent (e.g. 0.01, 1e-10)
	--div=DIV	Required. Sequence divergence for alignment subcomponent. (e.g. 0.2, 0.5, 0.8)
	-u USERSEQS, --userseqs=USERSEQS	Optional. Path to file containing FastA formatted sequences from the query database. Useful if you are interested in only a small subset of genes as opposed to the entire list of orthologs between two genomes.
	-q QUERYDB, --querydb=QUERYDB	Required. Path to the query database (fastaA format) and, if forward blast hits option is not given, wublast formatted index files with same base name
	-s SUBJECTDB, --subjectdb=SUBJECTDB	Required. Path to the subject database (fastA format) and, if reverse blast hits option not given, wublast formatted index files with same

		base name
	-o OUTFILE, --outfile=OUTFILE	Required. Path to RSD output file (will be overwritten)
	--fbh=FBH	Optional. Path to forward blast hits file, a file (generated by Blast_compute) which caches the results from the BLASTing every sequence in the query db against the subject db and vice versa. Used to avoid repeated blasting when RSD run multiple times with the same query db and subject but different threshold and divergence parameters.
	--rbh=RBH	Optional. Path to reverse blast hits file, otherwise instructions the same as above.
	-n NHITS, --nhits=NHITS	Optional. Default = 3. The number of top blast hits you would like to investigate as possible orthologs.
	--fa=FIXALPHA	Optional. Default = 1 (alpha fixed at specified or default value). If set to 0, alpha will be estimated by the program codeml.
	-a ALPHA, --alpha=ALPHA	Optional. Default=1.53. α = value for shape parameter of the gamma distribution.

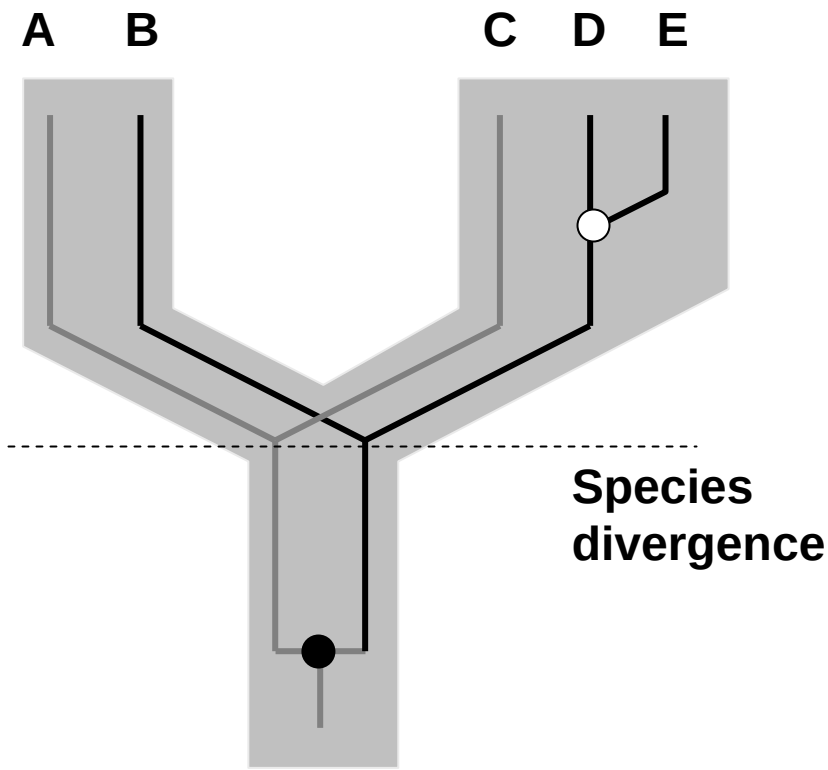
Figures

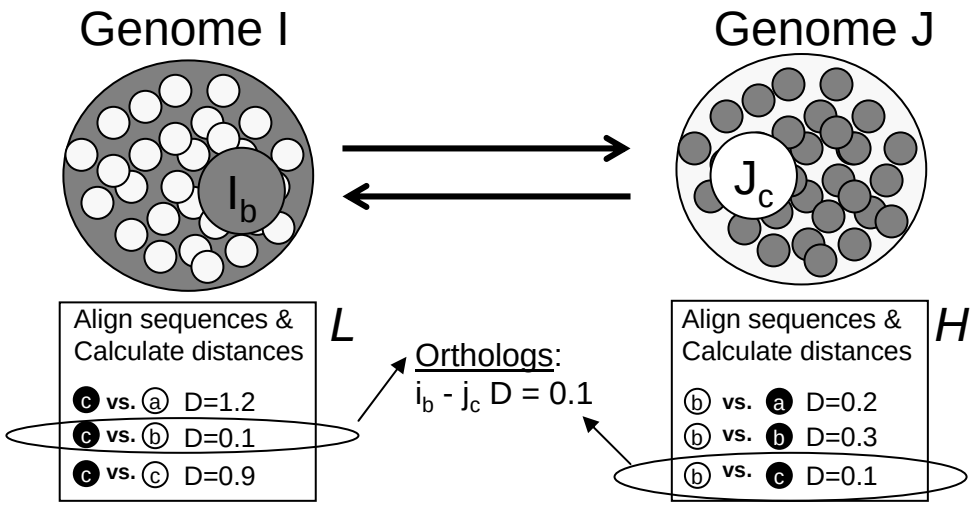
Figure 1. Conflating paralogs and orthologs.

An ortholog is defined as a gene in two species that has evolved directly from a single gene in their last common ancestor ***(52)***, that is, that arose at the evolutionary split between two species, and that conveys the same function in both species. Conversely paralogs are sequences that diverged from each other at some other time, and that either evolve new function or lose function entirely via effects of genetic drift. Here A and B are orthologs to D, but paralogs to E. Distinguishing between paralogs and orthologs is important to many areas of study, including functional annotation of genes in newly sequenced genomes and to accurately estimating the rates of protein diversification in genomes. Thick lines represent the organism tree, thin lines represent the gene tree.

Figure 2: The reciprocal smallest distance algorithm (*RSD*).

Arrows denote bidirectional BLAST runs. After each run, hits are paired with the query to calculate evolutionary distances. If the same pair produces the smallest distance in both search directions, it is assumed to be orthologous. The specifics of the algorithm are provided in the Introduction.


A
B
C
D
E
Species
divergence


Genome I
Genome J
I_b
J_c
Align sequences &
Calculate distances
c vs. a D=1.2
c vs. b D=0.1
c vs. c D=0.9
L
Orthologs:
i_b - j_c D = 0.1
Align sequences &
Calculate distances
b vs. a D=0.2
b vs. b D=0.3
b vs. c D=0.1
H